Automated Theorem Proving in Higher-Order Logics

Christoph E. Benzmüller

Special thanks to: Chad E. Brown

http://www.ags.uni-sb.de/~chris/

ATPHOL’06

SS06, Lecture Course at TU Darmstadt, Germany

UNIVERSITÄT DES SAARLANDES

Introduction Meeting

Outline for Today

- Notion of higher-order logic and focus of this lecture

Outline for Today

- Notion of higher-order logic and focus of this lecture
- ’Who am I?’ and ’Who are You?’

Outline for Today

- Notion of higher-order logic and focus of this lecture
- 'Who am I?' and 'Who are You?'
- Organizational issues

Outline for Today

- Notion of higher-order logic and focus of this lecture
- ’Who am I?’ and ’Who are You?’
- Organizational issues
- Overview on my current research projects (if time permits)

Outline for Today

- Notion of higher-order logic and focus of this lecture
- ’Who am I?’ and ’Who are You?’
- Organizational issues
- Overview on my current research projects (if time permits)
- (Actual lecture starts on Monday!)

Notion of Higher-Order Logic

We are interested in expressive logic language(s):

- quantification over functions and predicate variables

Notion of Higher-Order Logic

We are interested in expressive logic language(s):

- quantification over functions and predicate variables
  - $\forall x_{nat} \forall y_{nat} . x + y = y + x$ versus
    $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat} . x \circ y = y \circ x$

Notion of Higher-Order Logic

We are interested in expressive logic language(s):

- quantification over functions and predicate variables
  - $\forall x_{nat} \forall y_{nat}. x + y = y + x$ versus
    $\exists \circ_{nat \times nat \to nat} \forall x_{nat} \forall y_{nat}. x \circ y = y \circ x$
  - $\lambda x_{nat}. x \in A \wedge x \in B$ is a witness for $\exists p_{nat \to o}. A \cap B \subseteq p$

Notion of Higher-Order Logic

We are interested in expressive logic language(s):

- quantification over functions and predicate variables
  - $\forall x_{nat} \forall y_{nat}. x + y = y + x$ versus
    $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat}. x \circ y = y \circ x$
  - $\lambda x_{nat}. x \in A \wedge x \in B$ is a witness for $\exists p_{nat \rightarrow o}. A \cap B \subseteq p$
- λ-terms to denote/represent (unnamed) functions, predicates and sets

Notion of Higher-Order Logic

We are interested in expressive logic language(s):

- quantification over functions and predicate variables
  - $\forall x_{nat} \forall y_{nat}. x + y = y + x$ versus
    $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat}. x \circ y = y \circ x$
  - $\lambda x_{nat}. x \in A \wedge x \in B$ is a witness for $\exists p_{nat \rightarrow o}. A \cap B \subseteq p$
- λ-terms to denote/represent (unnamed) functions, predicates and sets
  - a function f on naturals with $x \rightarrow x + 1$ can be represented by the unnamed λ-term $\lambda x_{nat}. x + 1$

Notion of Higher-Order Logic

We are interested in expressive logic language(s):

- quantification over functions and predicate variables
  - $\forall x_{nat} \forall y_{nat}. x + y = y + x$ versus
    $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat}. x \circ y = y \circ x$
  - $\lambda x_{nat}. x \in A \wedge x \in B$ is a witness for $\exists p_{nat \rightarrow o}. A \cap B \subseteq p$
- λ-terms to denote/represent (unnamed) functions, predicates and sets
  - a function f on naturals with $x \rightarrow x + 1$ can be represented by the unnamed λ-term $\lambda x_{nat}. x + 1$
  - the set of all even naturals can be represented by the unnamed λ-term $\lambda x_{nat}. \mathrm{even}\ x$

Notion of Higher-Order Logic

- ...

Notion of Higher-Order Logic

- ...
- predicates and functions over other predicates and functions

Notion of Higher-Order Logic

- ...
- predicates and functions over other predicates and functions
  - $\exists \circ_{nat \times nat \to nat} \forall x_{nat} \forall y_{nat} \forall z_{nat}.\mathsf{commutative}(\circ) \wedge (x \circ y) \circ z = x \circ (y \circ z)$

Notion of Higher-Order Logic

- ...
- predicates and functions over other predicates and functions
  - $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat} \forall z_{nat}.\mathsf{commutative}(\circ) \wedge (x \circ y) \circ z = x \circ (y \circ z)$
- types to avoid logical paradoxes and inconsistencies

Notion of Higher-Order Logic

- ...
- predicates and functions over other predicates and functions
  - $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat} \forall z_{nat}.\mathsf{commutative}(\circ) \wedge (x \circ y) \circ z = x \circ (y \circ z)$
- types to avoid logical paradoxes and inconsistencies
  - “the set of all sets that do not contain themselves”

Notion of Higher-Order Logic

- ...
- predicates and functions over other predicates and functions
  - $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat} \forall z_{nat}. commutative(\circ) \wedge (x \circ y) \circ z = x \circ (y \circ z)$
- types to avoid logical paradoxes and inconsistencies
  - "the set of all sets that do not contain themselves"
  - our choice is: Alonzo Church's Simple Type Theory / Classical Higher-Order Logic

Notion of Higher-Order Logic

- ...
- predicates and functions over other predicates and functions
  - $\exists \circ_{nat \times nat \rightarrow nat} \forall x_{nat} \forall y_{nat} \forall z_{nat}.\mathsf{commutative}(\circ) \wedge (x \circ y) \circ z = x \circ (y \circ z)$
- types to avoid logical paradoxes and inconsistencies
  - "the set of all sets that do not contain themselves"
  - our choice is: Alonzo Church's Simple Type Theory / Classical Higher-Order Logic
  - we will not study rich type systems supporting polymorphism or dependent types

Focus of the Lecture

$$\exists p.\, p \supseteq A \cap B$$

p={x|x in A and x in B}

Focus of the Lecture

- Syntax: Simply typed λ-calculus and Church's Simple Type Theory

Focus of the Lecture

- Syntax: Simply typed λ-calculus and Church's Simple Type Theory
- Semantics: applicative structures, different model classes (including Henkin semantics), extensionality and equality

Focus of the Lecture

- Syntax: Simply typed λ-calculus and Church's Simple Type Theory
- Semantics: applicative structures, different model classes (including Henkin semantics), extensionality and equality
- Abstract consistency, Hintikka sets, and model existence

Focus of the Lecture

- Syntax: Simply typed λ-calculus and Church's Simple Type Theory
- Semantics: applicative structures, different model classes (including Henkin semantics), extensionality and equality
- Abstract consistency, Hintikka sets, and model existence
- Calculi: natural deduction, sequent calculus, unification, extensional resolution, soundness and completeness, cut-elimination and cut-simulation

Focus of the Lecture

- Syntax: Simply typed λ-calculus and Church's Simple Type Theory
- Semantics: applicative structures, different model classes (including Henkin semantics), extensionality and equality
- Abstract consistency, Hintikka sets, and model existence
- Calculi: natural deduction, sequent calculus, unification, extensional resolution, soundness and completeness, cut-elimination and cut-simulation
- The theorem prover LEO

Relevance and Applications

Applications of Higher-Order Theorem Proving

- Formal Methods

Relevance and Applications

Applications of Higher-Order Theorem Proving

- Formal Methods
- Mathematics

Relevance and Applications

Applications of Higher-Order Theorem Proving

- Formal Methods
- Mathematics
- Functional and Logical Programming

Relevance and Applications

Applications of Higher-Order Theorem Proving

- Formal Methods
- Mathematics
- Functional and Logical Programming
- AI

Relevance and Applications

Applications of Higher-Order Theorem Proving

- Formal Methods
- Mathematics
- Functional and Logical Programming
- AI
  - CYC

Relevance and Applications

Applications of Higher-Order Theorem Proving

- Formal Methods
- Mathematics
- Functional and Logical Programming
- AI
  - CYC
  - ...

Relevance and Applications

Applications of Higher-Order Theorem Proving

- Formal Methods
- Mathematics
- Functional and Logical Programming
- AI
  - CYC
  - ...
- Computational Linguistics

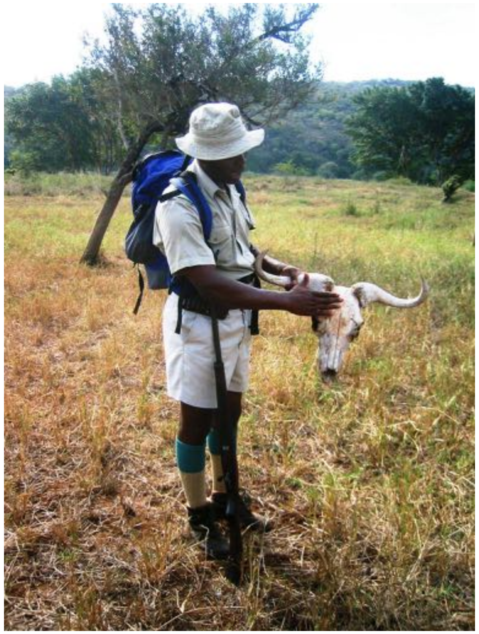
Relevance and Applications

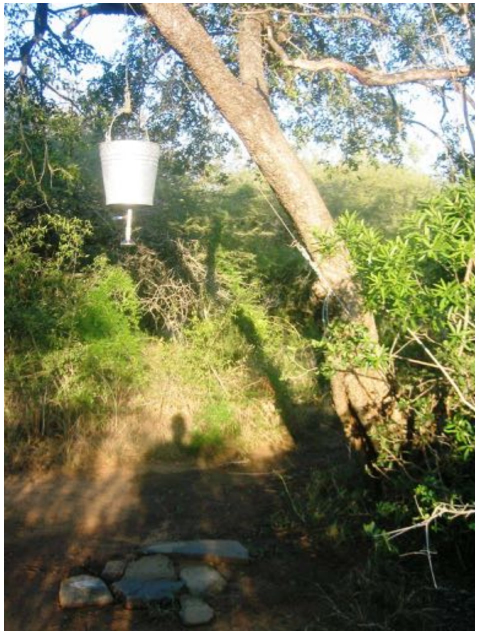
Applications of Higher-Order Theorem Proving

- Formal Methods
- Mathematics
- Functional and Logical Programming
- AI
  - CYC
  - ...
- Computational Linguistics
- ...

’Who am I?’ and ’Who are You?’

Who am I?





UNIVERSITÄT DES SAARLANDES

Who am I?

- Grown-up in Saarburg and Trier

Who am I?

- Grown-up in Saarburg and Trier

  - schools in Saarburg and Trier
  - freelancer of daily newspaper in Trier

Who am I?

- Grown-up in Saarburg and Trier

  - schools in Saarburg and Trier
  - freelancer of daily newspaper in Trier

- Sport: Middle- and Long-distance Running

Who am I?

- Grown-up in Saarburg and Trier

  - schools in Saarburg and Trier
  - freelancer of daily newspaper in Trier

- Sport: Middle- and Long-distance Running

  - German champion
  - Olympic Center (OSP) Saarbrücken

Who am I?

- Grown-up in Saarburg and Trier

  - schools in Saarburg and Trier
  - freelancer of daily newspaper in Trier

- Sport: Middle- and Long-distance Running

  - German champion
  - Olympic Center (OSP) Saarbrücken

- Computer Science Study in Saarbrücken (89-95):

Who am I?

- Grown-up in Saarburg and Trier

  - schools in Saarburg and Trier
  - freelancer of daily newspaper in Trier

- Sport: Middle- and Long-distance Running

  - German champion
  - Olympic Center (OSP) Saarbrücken

- Computer Science Study in Saarbrücken (89-95):

  - aside of professional training at OSP
  - 'caught' by Jörg Siekmann's AI lecture
  - focus: formal methods, algebraic specification, automated reasoning

UNIVERSITÄT DES SAARLANDES

Who am I?

- PhD at Saarbrücken and CMU, Pittsburgh (95-99)

Who am I?

- PhD at Saarbrücken and CMU, Pittsburgh (95-99)

  - mechanization/automation of higher-order logic
  - higher-order theorem prover LEO
  - semantics of higher-order logic
  - Stipend of the Deutsche Studienstiftung
  - J. Siekmann, M. Kohlhase, F. Pfenning, P. Andrews

UNIVERSITÄT DES SAARLANDES

Who am I?

- PhD at Saarbrücken and CMU, Pittsburgh (95-99)

  - mechanization/automation of higher-order logic
  - higher-order theorem prover LEO
  - semantics of higher-order logic
  - Stipend of the Deutsche Studienstiftung
  - J. Siekmann, M. Kohlhase, F. Pfenning, P. Andrews

- PostDoc in UK (1999/2000)

Who am I?

- PhD at Saarbrücken and CMU, Pittsburgh (95-99)

  - mechanization/automation of higher-order logic
  - higher-order theorem prover LEO
  - semantics of higher-order logic
  - Stipend of the Deutsche Studienstiftung
  - J. Siekmann, M. Kohlhase, F. Pfenning, P. Andrews

- PostDoc in UK (1999/2000)

  - Univ. of Birmingham and Univ. of Edinburgh
  - EPSRC grant
  - focus on agent-oriented theorem proving

Who am I?

- Head of OMEGA group / Dozent at Saarland University (since 2001)

Who am I?

- Head of OMEGA group / Dozent at Saarland University (since 2001)
  - focus on assistance systems for formal methods and mathematics
  - group has approx. 10 full time researchers
  - principal investigator of 2 projects in the
    SFB 378: Resource-adaptive Cognitive Processes

Who am I?

- Head of OMEGA group / Dozent at Saarland University (since 2001)
  - focus on assistance systems for formal methods and mathematics
  - group has approx. 10 full time researchers
  - principal investigator of 2 projects in the

    SFB 378: Resource-adaptive Cognitive Processes
- SFB Project OMEGA:

Who am I?

- Head of OMEGA group / Dozent at Saarland University (since 2001)
  - focus on assistance systems for formal methods and mathematics
  - group has approx. 10 full time researchers
  - principal investigator of 2 projects in the

    SFB 378: Resource-adaptive Cognitive Processes
- SFB Project OMEGA:

  - integrated environment of maths reasoning tools
  - application of artificial intelligence techniques

Who am I?

- Head of OMEGA group / Dozent at Saarland University (since 2001)
  - focus on assistance systems for formal methods and mathematics
  - group has approx. 10 full time researchers
  - principal investigator of 2 projects in the

    SFB 378: Resource-adaptive Cognitive Processes
- SFB Project OMEGA:
  - integrated environment of maths reasoning tools
  - application of artificial intelligence techniques

- SFB Project DIALOG:

Who am I?

- Head of OMEGA group / Dozent at Saarland University (since 2001)
  - focus on assistance systems for formal methods and mathematics
  - group has approx. 10 full time researchers
  - principal investigator of 2 projects in the
    SFB 378: Resource-adaptive Cognitive Processes
- SFB Project OMEGA:

  - integrated environment of maths reasoning tools
  - application of artificial intelligence techniques
- SFB Project DIALOG:

  - natural tutorial dialog: student ↔ maths assistance system
  - collaboration with Computational Linguistics

Who are You?

Organizational Issues

Before we start . . .

We need to discuss and fix:

- Miscellaneous

Before we start . . .

We need to discuss and fix:

- Miscellaneous
- Lectures

Before we start . . .

We need to discuss and fix:

- Miscellaneous
- Lectures
- Exercises and Tutorials

Before we start . . .

We need to discuss and fix:

- Miscellaneous
- Lectures
- Exercises and Tutorials
- Examination

Miscellaneous

- Lecture Webpage:
  http://www.ags.uni-sb.de/ chris/ATPHOL-06/
  This webpage will contain important information such as:
  dates, announcements, link to slides and exercises, literature, etc.

Miscellaneous

- Lecture Webpage:
  http://www.ags.uni-sb.de/ chris/ATPHOL-06/
  This webpage will contain important information such as: dates, announcements, link to slides and exercises, literature, etc.
- I will handout a registration questionnaire: Please write your name, immatriculation number, semester, etc.

Miscellaneous

- Lecture Webpage:
  http://www.ags.uni-sb.de/ chris/ATPHOL-06/
  This webpage will contain important information such as: dates, announcements, link to slides and exercises, literature, etc.
- I will handout a registration questionnaire: Please write your name, immatriculation number, semester, etc.
- Except for the lecturing dates and times, I will be in Saarbrücken. To discuss problems etc., please write an e-mail

Miscellaneous

- Lecture Webpage:
  http://www.ags.uni-sb.de/ chris/ATPHOL-06/
  This webpage will contain important information such as: dates, announcements, link to slides and exercises, literature, etc.
- I will handout a registration questionnaire: Please write your name, immatriculation number, semester, etc.
- Except for the lecturing dates and times, I will be in Saarbrücken. To discuss problems etc., please write an e-mail
- Organizational support: Markus Aderhold and Prof. Christoph Walther

Miscellaneous

- Lecture Webpage:
  http://www.ags.uni-sb.de/ chris/ATPHOL-06/
  This webpage will contain important information such as: dates, announcements, link to slides and exercises, literature, etc.
- I will handout a registration questionnaire: Please write your name, immatriculation number, semester, etc.
- Except for the lecturing dates and times, I will be in Saarbrücken. To discuss problems etc., please write an e-mail
- Organizational support: Markus Aderhold and Prof. Christoph Walther
- Please note: I have no team that supports this lecture and I am preparing the lectures in my spare time.

Lectures

- Discussion of the exact start end end time of the lectures; do we want a break?

Lectures

- Discussion of the exact start end end time of the lectures; do we want a break?
- Difficulty: I want to prerequire that you are familiar with propositional and first order logic syntax and semantics (and probably mechanization). If this is not the case please let's discuss this issue now.

Lectures

- Discussion of the exact start end end time of the lectures; do we want a break?
- Difficulty: I want to prerequire that you are familiar with propositional and first order logic syntax and semantics (and probably mechanization). If this is not the case please let's discuss this issue now.
- Lectures will be a mixture between beamer presentation and traditional blackboard lecture.

Lectures

- Discussion of the exact start end end time of the lectures; do we want a break?
- Difficulty: I want to prerequire that you are familiar with propositional and first order logic syntax and semantics (and probably mechanization). If this is not the case please let's discuss this issue now.
- Lectures will be a mixture between beamer presentation and traditional blackboard lecture.

Exercises and Tutorials

- Theoretical exercises will be announced on the slides.

Exercises and Tutorials

- Theoretical exercises will be announced on the slides.
- They will be discussed in the tutorials; the tutorials will be held by myself.

Exercises and Tutorials

- Theoretical exercises will be announced on the slides.
- They will be discussed in the tutorials; the tutorials will be held by myself.
- The exercises will not be corrected/graded, however, I expect that every student regularly presents a solution of the theoretical exercises on the blackboard; these regular presentations are required for entering the final examination.

Exercises and Tutorials

- Theoretical exercises will be announced on the slides.
- They will be discussed in the tutorials; the tutorials will be held by myself.
- The exercises will not be corrected/graded, however, I expect that every student regularly presents a solution of the theoretical exercises on the blackboard; these regular presentations are required for entering the final examination.
- I am also planning a practical exercise parallel to the lecture: we will form teams of students and each team will implement an algorithm related to the lecture (most likely higher-order (pre-)unification); at the end of the lecture these implementations will be evaluated in a competition.

Examination

- Written or oral exam as part of a masters study or as part of a diploma exam in a diploma study (e.g. with Prof. Walther)

Examination

- Written or oral exam as part of a masters study or as part of a diploma exam in a diploma study (e.g. with Prof. Walther)
- The entry requirements for the final exam are

Examination

- Written or oral exam as part of a masters study or as part of a diploma exam in a diploma study (e.g. with Prof. Walther)
- The entry requirements for the final exam are
  - regular presentations in tutorials

Examination

- Written or oral exam as part of a masters study or as part of a diploma exam in a diploma study (e.g. with Prof. Walther)
- The entry requirements for the final exam are
  - regular presentations in tutorials
  - passed evaluation of the practical exercise

Examination

- Written or oral exam as part of a masters study or as part of a diploma exam in a diploma study (e.g. with Prof. Walther)
- The entry requirements for the final exam are
  - regular presentations in tutorials
  - passed evaluation of the practical exercise
- Do we have to fix dates for the evaluation of the practical exercise evaluation/competition and for the final exam?

Examination

- Written or oral exam as part of a masters study or as part of a diploma exam in a diploma study (e.g. with Prof. Walther)
- The entry requirements for the final exam are
  - regular presentations in tutorials
  - passed evaluation of the practical exercise
- Do we have to fix dates for the evaluation of the practical exercise evaluation/competition and for the final exam?
- Any questions?

Examination

- Written or oral exam as part of a masters study or as part of a diploma exam in a diploma study (e.g. with Prof. Walther)
- The entry requirements for the final exam are
  - regular presentations in tutorials
  - passed evaluation of the practical exercise
- Do we have to fix dates for the evaluation of the practical exercise evaluation/competition and for the final exam?
- Any questions?

Overview on my current research projects ($\longrightarrow$ other slides)

Some Historical Remarks

History

- Ancient Greek's (mostly Aristoteles):
  laws of human thought; theory of well-chosen axioms and rules (syllogism)

  Examples:

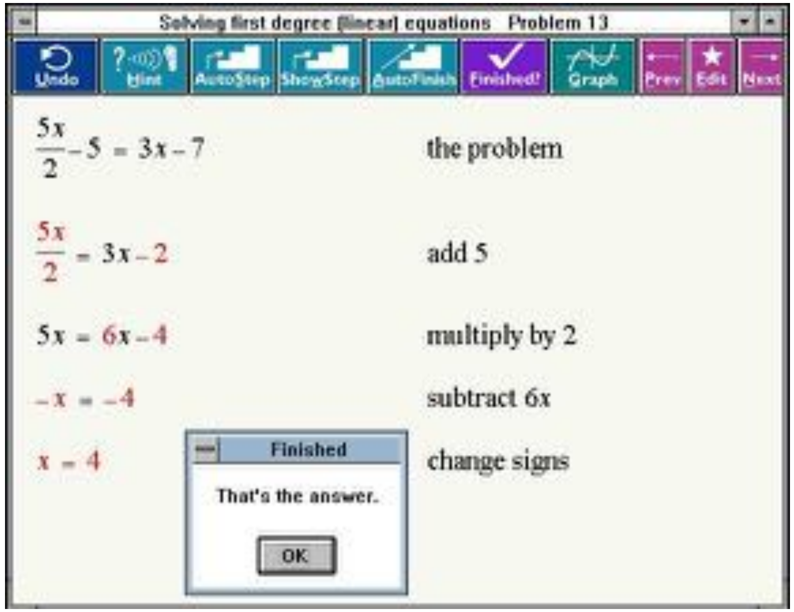
  - Modus Ponens: $$\frac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}\ \mathsf{mp}$$

  - Modus Tollens: $$\frac{\mathsf{A} \Rightarrow \mathbf{B} \quad \neg \mathsf{B}}{\neg \mathsf{A}}\ \mathsf{mt}$$

History (Cont'd)

- Computer algebra systems
  - Roots: Abacus (approx. 500 a.D.)
  - mechanical calculators built from 15. century on
  - modern computer algebra systems are today widely used

W. Schickard's mechanical calculator (1592 - 1635)



MathPert system (Michael Beeson)

History (Cont'd)

- Gottfried Wilhelm Leibniz (1646-1716)
  - Dream of formalizing and mechanizing mathematical reasoning (lingua characteristica and calculus ratiocinator)
  - Mechanization of simple arithmetical operations, e.g., mechanical calculator capable of multiplication

History (Cont'd)

- It took until the end of the 19. century that *modern mathematical logic* was (re-)born:
  - George Boole (1815-1864),
  - Gottlob Frege (1848-1925),
  - Bertrand Russel (1872-1970),
  - and many others

  stimulated the new and deep interest of many researchers in the field of mathematical logic.

History (Cont'd)

- *Frege's Begriffsschrift* is described by Davis [Davis83] *not only as the direct ancestor of contemporary systems of mathematical logic but also as the ancestor of all formal languages, including computer programming languages*.
- First-order logic:

$$\forall x, y, z.(x + (y + z)) = ((x + y) + z)$$

History (Cont'd)

- *Hilbert's program* at the very beginning of this century [Hilbert04,Hilbert27] aimed at the complete development of modern mathematics in a formal system.

History (Cont'd)

- In the early 30's results came fast:
  - Kurt Gödel, Jacques Herbrand and Thoralf Skolem proved the *completeness of the (first-order) predicate calculus* in 1930 [Goedel30,Herbrand30,Skolem28]: every valid formula in the language of the predicate calculus is derivable from its axioms.
  - However, Gödel showed in his famous *incompleteness theorems* [Goedel31] that it is impossible to develop a generally complete calculus that mechanizes mathematical reasoning. More precisely, Gödel showed that *as soon as a system is rich enough to encode Peano arithmetic, one can construct sentences that are valid in Peano arithmetic but which are not derivable in the system itself.*

History (Cont'd)

- Gerhard Gentzen [Gentzen35]: Natural Deduction Calculus (ND)

ND-Rules (examples)

$$\frac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}\ \mathsf{mp}$$

$$\frac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}}\ \wedge I \qquad \frac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}}\ \wedge E_l \qquad \frac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}}\ \wedge E_r$$

$$\frac{\begin{array}{c}[\mathbf{A}]_1 \\ \vdots \\ \mathbf{B}\end{array}}{\mathbf{A} \Rightarrow \mathbf{B}}\ \Rightarrow I^1$$

...etc. ...

ND-Proof for $(\mathbf{A} \wedge \mathbf{B}) \Rightarrow (\mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A}))$

$$\dfrac{\dfrac{\dfrac{[\mathbf{A} \wedge \mathbf{B}]_1}{\mathbf{B}}\ \wedge E_r \qquad \dfrac{\dfrac{[\mathbf{A} \wedge \mathbf{B}]_1}{\mathbf{A}}\ \wedge E_l}{\mathbf{C} \vee \mathbf{A}}\ \vee I_r}{\mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A})}\ \wedge I}{(\mathbf{A} \wedge \mathbf{B}) \Rightarrow (\mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A}))}\ \Rightarrow I^1$$

- Introduced Sequent Calculus initially as tool for investigating cut elimination; however, many interactive theorem provers today employ sequent calculus and not natural deduction calculus.

History (Cont'd)

- Development of electronic computers in the 40's and 50's: disappointment gradually gave away to attempts of developing and implementing proof procedures in practice.
- J.A. Robinson: achieved important break-through (in first-order theorem proving) with his resolution approach in 1965 [Robinson65]. The most important improvement of this approach compared to former ones is that in order to prove a theorem it tries to *refute the negated theorem in a goal directed way*, thereby *employing first-order unification as a powerful filter* instead of simply enumerating the Herbrand universe like most earlier methods.
- Remark: *This lecture (amongst others) investigates the problems of applying the resolution idea in higher-order logic.*

History (Cont'd)

- Robinson's ideas are still employed in many state of the art theorem provers such as OTTER, EQP (which solved the Robbins Problem), or the superposition based prover SPASS.
- Even tableaux based provers like PROTEIN are rather closely related to the resolution approach and unification became an essential (filtering) tool for the whole field.

History HOL

- Higher-order logic: any simply typed logical system that allows quantification over function and predicate variables.
- It was Bertrand Russel [Russell02,Russell03] who first pointed out in 1902 that in connection with the comprehension principles (these principles assure the existence of certain functions) this may allow for *paradoxes*: most prominent example is the *set of all non-self-containing sets*.
- As a possible solution Russel suggested a few years later in [Russell08] a theory of types as a basis for the formalization of mathematics that differentiates between objects and sets (or functions) consisting of these kinds of objects.

History HOL (Cont'd)

- Idea was also taken up by Alonzo Church in 1940, who invented the *simply typed λ-calculus* [Church40] in order to prevent such paradoxes in the untyped λ-calculus, which he developed with Schönfinkel and Curry ten years earlier.
- Typed and untyped λ-calculi play an important or even central role in many research fields of modern computer science.
- The avoidance of paradoxes like Russel's paradox is also a main reason why we employ a logic based on Church's simply typed λ-calculus [Church40] — i.e., *classical type theory / classical higher-order logic* — in this lecture.

History HOL (Cont'd)

- Relatively few researchers concentrated on the mechanization of higher-order logic.
- Robinson presents in [Robinson68,Robinson69] a higher-order proof procedure based on the tableaux idea that itself employs many ideas from the calculi given in [Schuette60] and [Takeuti53].
- The most important works to be mentioned are Peter Andrews' investigation of higher-order resolution [Andrews71], Jensen and Pietrowski's approach [JePi72] and especially Gerard Huet's constrained resolution approach [Huet72,Huet73].

History (Cont'd)

- Big challenge for the mechanization of higher-order logic is the undecidability of higher-order unification [Lucchesi72,Huet73,Goldfarb81].
- Andrews' resolution approach still avoids unification (and instead employs an enumeration of the universe)
- Huet's constrained resolution approach [Huet72] solves the problem by encoding the particular unification problems as unification constraints and by delaying the application of higher-order unification until the end of a refutation.
- Huet's approach additionally gains from the higher-order pre-unification algorithm [Huet75] which avoids the guessing aspects of full higher-order unification

Introduction

λ-Calculus: Motivation

λ-Calculus: Motivation

Consider the following arithmetical computations

$$
\begin{aligned}
(-1)^2 - 1 &= 0 \\
(1)^2 - 1 &= 0 \\
(2)^2 - 1 &= 3 \\
&\dots
\end{aligned}
$$

λ-Calculus: Motivation

Consider the following arithmetical computations

$$(-1)^2 - 1 = 0$$
$$(1)^2 - 1 = 0$$
$$(2)^2 - 1 = 3$$
$$\ldots$$

A more general arithmetic expression for the LHS:

$$x^2 - 1$$

λ-Calculus: Motivation

Consider the 0's (Nullstellen) of this function; we can express the existence of two 0's in first-order logic as follows

$$\exists n, m. n^2 - 1 = 0 \wedge m^2 - 1 = 0 \wedge n \neq m$$

λ-Calculus: Motivation

Consider the 0's (Nullstellen) of this function; we can express the existence of two 0's in first-order logic as follows

$$\exists n, m. n^2 - 1 = 0 \land m^2 - 1 = 0 \land n \neq m$$

Now we may want to talk about the existence of a function f with two 0's:

$$(1) \quad \exists f. \exists n, m. f(n) = 0 \land f(m) = 0 \land n \neq m$$

λ-Calculus: Motivation

Consider the 0's (Nullstellen) of this function; we can express the existence of two 0's in first-order logic as follows

$$\exists n, m. n^2 - 1 = 0 \wedge m^2 - 1 = 0 \wedge n \neq m$$

Now we may want to talk about the existence of a function f with two 0's:

$$(1) \quad \exists f. \exists n, m. f(n) = 0 \wedge f(m) = 0 \wedge n \neq m$$

This expression is not a first-order statement; however we want to be able to express such statements. We also want to prove such statements and in a constructive proof we would like to provide witnesses for f and n, m. In first-order logic we can describe f by the following equation

$$f(x) = x^2 - 1$$

λ-Calculus: λ-terms

In λ-calculus the specified function f can be described (without giving it a name) by the witnessing λ-term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

and the witnesses for n and m are -1 and 1.

λ-Calculus: Set of λ-expressions

Given a countably infinite set of identifiers, say $a, b, c, \ldots, x, y, z, x1, x2, \ldots$. The set of all λ-expressions can then be described by the following context-free grammar in BNF:

1. <expr> ::= <identifier>

λ-Calculus: Set of λ-expressions

Given a countably infinite set of identifiers, say $a, b, c, \ldots, x, y, z, x1, x2, \ldots$. The set of all λ-expressions can then be described by the following context-free grammar in BNF:

1. <expr> ::= <identifier>
2. <expr> ::= [λ <identifier> . <expr>] abstraction

λ-Calculus: Set of λ-expressions

Given a countably infinite set of identifiers, say $a, b, c, ..., x, y, z, x1, x2, ...$ The set of all λ-expressions can then be described by the following context-free grammar in BNF:

1. <expr> ::= <identifier>
2. <expr> ::= [λ <identifier> . <expr>] — abstraction
3. <expr> ::= [<expr> <expr>] — application

λ-Calculus: Conventions

We often omit brackets with the following conventions:

- $[F\,A\,B]$ means $[[F\,A]\,B]$. (Application associates to the left.)

λ-Calculus: Conventions

We often omit brackets with the following conventions:

- $[\mathrm{F\,A\,B}]$ means $[[\mathrm{F\,A}]\,\mathrm{B}]$. (Application associates to the left.)
- $[\lambda \mathrm{x}.\lambda \mathrm{y}.\,\mathrm{B}]$ means $[\lambda \mathrm{x}.[\lambda \mathrm{y}.\,\mathrm{B}]]$.

λ-Calculus: Conventions

We often omit brackets with the following conventions:

- $[F\ A\ B]$ means $[[F\ A]\ B]$. (Application associates to the left.)
- $[\lambda x. \lambda y.\ B]$ means $[\lambda x. [\lambda y.\ B]]$.
- A dot (except possibly after λ <identifier>) stands for a left bracket whose mate is as far to the right as possible without changing the existing bracketing.

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\exists \mathsf{f}.\exists \mathsf{n}, \mathsf{m}.\mathsf{f}(\mathsf{n}) = 0 \wedge \mathsf{f}(\mathsf{m}) = 0 \wedge \mathsf{n} \neq \mathsf{m}$$

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\exists f.\exists n, m.f(n) = 0 \land f(m) = 0 \land n \neq m$$

$$\stackrel{f}{\longrightarrow} \exists n, m.[[\lambda x.x^2 - 1]\, n] = 0 \land [[\lambda x.x^2 - 1]\, m] = 0 \land n \neq m$$

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\exists f.\exists n, m.f(n) = 0 \land f(m) = 0 \land n \neq m$$
$$\xrightarrow{f} \exists n, m.[[\lambda x.x^2 - 1]\, n] = 0 \land [[\lambda x.x^2 - 1]\, m] = 0 \land n \neq m$$
$$\xrightarrow{n,m} [[\lambda x.x^2 - 1]\ -1] = 0 \land [[\lambda x.x^2 - 1]\, 1] = 0 \land -1 \neq 1$$

λ-Calculus: β-reduction

Consider now the instantiation of (1) with these witness terms

$$\begin{aligned}
&\exists f.\exists n, m.f(n) = 0 \wedge f(m) = 0 \wedge n \neq m \\
\stackrel{f}{\longrightarrow}\ &\exists n, m.[[\lambda x.x^2 - 1]\, n] = 0 \wedge [[\lambda x.x^2 - 1]\, m] = 0 \wedge n \neq m \\
\stackrel{n,m}{\longrightarrow}\ &[[\lambda x.x^2 - 1]\ -1] = 0 \wedge [[\lambda x.x^2 - 1]\, 1] = 0 \wedge -1 \neq 1
\end{aligned}$$

Finally we can 'evaluate' function applications by so called β-reduction

$$((-1)^2 - 1) = 0 \wedge (1^2 - 1) = 0 \wedge -1 \neq 1$$

λ-Calculus: β-reduction

The β-reduction rule expresses the idea of function application as motivated on the previous slide. Formally it states that

$$[[\lambda \mathsf{x}.\, \mathsf{A}]\, \mathsf{B}] \longrightarrow_{\beta} \mathsf{A}[\mathsf{x}/\mathsf{B}]$$

if all free occurrences in B remain free in $\mathsf{A}[\mathsf{x}/\mathsf{B}]$. Here, $\mathsf{A}[\mathsf{x}/\mathsf{B}]$ means the expression E with every free occurrence of x in A replaced with B.

λ-Calculus: Currying

A function of two variables is expressed in lambda calculus as a function of one argument which returns a function of one argument. For instance, the function

$$f(x, y) = x^2 - y$$

is encoded as

$$[\lambda x.\lambda y.x^2 - y]$$

λ-Calculus: α-conversion

The names of the bound variables are unimportant:

$$\lambda \mathrm{x}.\mathrm{x}^2 - 1 \text{ and } \lambda \mathrm{y}.\mathrm{y}^2 - 1$$

denote the same function.

λ-Calculus: α-conversion

The names of the bound variables are unimportant:

$$\lambda x.x^2 - 1 \text{ and } \lambda y.y^2 - 1$$

denote the same function.
Formally, the α-conversion rule states that if x and y are variables and A is a λ-expression then

$$[\lambda x.A] \longleftrightarrow_{\alpha} [\lambda y.A[x/y]]$$

if y does not appear freely in A and y is not bound by a λ in A whenever it replaces a x.

λ-Calculus: η-reduction

η-reduction expresses the idea of (functional) extensionality, which in this context is that two functions are the same iff they give the same result for all arguments:

$$[\lambda \mathsf{x}.\mathsf{F}\mathsf{x}] \longrightarrow_{\eta} \mathsf{F}$$

whenever x does not appear free in F.

λ-Calculus: $\beta\eta$-equivalence

- We define $\longleftrightarrow^*_{\alpha\beta\eta}$ as the smallest equivalence relation closed under the reduction rules $\longrightarrow_\beta$ and $\longrightarrow_\eta$ and α-conversion. (Similarly we may define $\longleftrightarrow^*_{\mathrm{M}}$ for M $\subset \{\alpha, \beta, \eta\}$)

λ-Calculus: $\beta\eta$-equivalence

- We define $\longleftrightarrow^*_{\alpha\beta\eta}$ as the smallest equivalence relation closed under the reduction rules $\longrightarrow_\beta$ and $\longrightarrow_\eta$ and α-conversion. (Similarly we may define $\longleftrightarrow^*_{\mathsf{M}}$ for M $\subset \{\alpha, \beta, \eta\}$)

- We call two λ-terms E and T $\alpha\beta\eta$-equivalent (or short equivalent) if

$$\mathsf{E} \longleftrightarrow^*_{\alpha\beta\eta} \mathsf{T}$$

λ-Calculus: $\beta\eta$-equivalence

- We define $\longleftrightarrow^*_{\alpha\beta\eta}$ as the smallest equivalence relation closed under the reduction rules $\longrightarrow_\beta$ and $\longrightarrow_\eta$ and α-conversion. (Similarly we may define $\longleftrightarrow^*_{\mathsf{M}}$ for M $\subset \{\alpha, \beta, \eta\}$)

- We call two λ-terms E and T $\alpha\beta\eta$-equivalent (or short equivalent) if

$$\mathsf{E} \longleftrightarrow^*_{\alpha\beta\eta} \mathsf{T}$$

(Similarly we may define M-equivalence for M $\subset \{\alpha, \beta, \eta\}$)

λ-Calculus: Normalforms

- A λ-expression is called a β-normal form if it does not allow any β-reduction, i.e., has no subexpression of the form

$$[[\lambda \mathsf{x} \,.\, \mathsf{A}]\, \mathsf{B}]$$

λ-Calculus: Normalforms

- A λ-expression is called a β-normal form if it does not allow any β-reduction, i.e., has no subexpression of the form

$$[[\lambda x\ .\ A]\ B]$$

- A λ-expression is called a η-normal form if it does not allow any η-reduction, i.e., has no subexpression of the form (where x does not occur free in E)

$$[\lambda x. E\ x]$$

λ-Calculus: Normalforms

- A λ-expression is called a β-normal form if it does not allow any β-reduction, i.e., has no subexpression of the form

$$[[\lambda x \,.\, A]\, B]$$

- A λ-expression is called a η-normal form if it does not allow any η-reduction, i.e., has no subexpression of the form (where x does not occur free in E)

$$[\lambda x. E\ x]$$

- A λ-expression is called a $\beta\eta$-normal form if it satisfies both conditions.

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)
- The Church-Rosser theorem(s) state that if $A \longrightarrow^* B$ and $A \longrightarrow^* C$, then there is some D such that $B \longrightarrow^* D$ and $C \longrightarrow^* D$.

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)
- The Church-Rosser theorem(s) state that if $A \longrightarrow^* B$ and $A \longrightarrow^* C$, then there is some D such that $B \longrightarrow^* D$ and $C \longrightarrow^* D$.

$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B \arrow[dr] & & C \arrow[dl] \\ & D & \end{tikzcd}$$

λ-Calculus: Normalforms

- Not every λ-expression is equivalent to a ?-normal form (where $? \in \{\beta, \beta\eta\}$)
- The Church-Rosser theorem(s) state that if $A \longrightarrow^* B$ and $A \longrightarrow^* C$, then there is some D such that $B \longrightarrow^* D$ and $C \longrightarrow^* D$.

$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B \arrow[dr] & & C \arrow[dl] \\ & D & \end{tikzcd}$$

- From Church-Rosser it follows that every term has at most one $*$-normal form (up to α-conversion).

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$[[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]]$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$\begin{aligned} &[[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]] \\ \longrightarrow_\beta\ &[\lambda y.[\lambda x.x^2 - 1][[\lambda x.x^2 - 1]y]] \end{aligned}$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$\begin{aligned} & [[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]] \\ \longrightarrow_\beta\ & [\lambda y.[\lambda x.x^2 - 1][[\lambda x.x^2 - 1]y] \\ \longrightarrow_\beta\ & \lambda y.[\lambda x.x^2 - 1]\ [y^2 - 1] \end{aligned}$$

λ-Calculus: Iteration

Consider twofold iteration of function $f := [\lambda x.x^2 - 1]$

$$f(f(x)) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

The following λ-term expresses twofold iteration of a function

$$[\lambda g.\lambda y.g\ [g\ y]]$$

Let us apply this λ-term now to our function f

$$\begin{aligned}
&[[\lambda g.\lambda y.g\ [g\ y]]\ [\lambda x.x^2 - 1]] \\
\longrightarrow_\beta\ &[\lambda y.[\lambda x.x^2 - 1][[\lambda x.x^2 - 1]y]] \\
\longrightarrow_\beta\ &\lambda y.[\lambda x.x^2 - 1]\ [y^2 - 1] \\
\longrightarrow_\beta\ &[\lambda y.[y^2 - 1]^2 - 1] = \lambda y.y^4 - 2y^2
\end{aligned}$$

λ-Calculus: Church Numerals

We employ iteration to define natural numbers as Church numerals:

$$\overline{0} = [\lambda f.\lambda x.x], \qquad \overline{1} = [\lambda f.\lambda x.fx], \qquad \overline{2} = [\lambda f.\lambda x.f(fx)], \qquad \ldots$$

λ-Calculus: Church Numerals

We employ iteration to define natural numbers as Church numerals:

$$\overline{0} = [\lambda f.\lambda x.x], \qquad \overline{1} = [\lambda f.\lambda x.fx], \qquad \overline{2} = [\lambda f.\lambda x.f(fx)], \qquad \ldots$$

Generally a natural number n is encoded as the Church numeral

$$\overline{n} = [\lambda f.\lambda y.f^n\, y]$$

where f^n is an abbreviation for $\underbrace{[f\ [f\ [f \ldots [f}_{n-\text{times}}\ y]]]$.

λ-Calculus: Church Numerals

We employ iteration to define natural numbers as Church numerals:

$$\overline{0} = [\lambda f.\lambda x.x], \qquad \overline{1} = [\lambda f.\lambda x.fx], \qquad \overline{2} = [\lambda f.\lambda x.f(fx)], \qquad \ldots$$

Generally a natural number n is encoded as the Church numeral

$$\overline{n} = [\lambda f.\lambda y.f^n\, y]$$

where f^n is an abbreviation for $\underbrace{[f\ [f\ [f \ldots [f}_{n-\text{times}}\ y]]]$.

Intuitively, the number n in lambda calculus is a function that takes a function f as argument and returns the n-th iterate of f.

λ-Calculus: Church Numerals

We can now define a successor function $\overline{\mathsf{SUCC}}$, which takes a number $\overline{\mathsf{n}}$ and returns $\overline{\mathsf{n}+1}$:

$$\overline{\mathsf{SUCC}} = [\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{f}[\mathsf{nfx}]]$$

λ-Calculus: Church Numerals

We can now define a successor function $\overline{\mathsf{SUCC}}$, which takes a number $\overline{\mathsf{n}}$ and returns $\overline{\mathsf{n}+1}$:

$$\overline{\mathsf{SUCC}} = [\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{f}[\mathsf{nfx}]]$$

Addition is the defined as follows:

$$\overline{\mathsf{PLUS}} = [\lambda \mathsf{m}.\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{mf}[\mathsf{nfx}]]$$

λ-Calculus: Church Numerals

We can now define a successor function $\overline{\mathsf{SUCC}}$, which takes a number $\overline{\mathsf{n}}$ and returns $\overline{\mathsf{n}+1}$:

$$\overline{\mathsf{SUCC}} = [\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{f}[\mathsf{nfx}]]$$

Addition is the defined as follows:

$$\overline{\mathsf{PLUS}} = [\lambda \mathsf{m}.\lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{mf}[\mathsf{nfx}]]$$

Multiplication can then be defined as

$$\overline{\mathsf{MULT}} = \lambda \mathsf{m}.\lambda \mathsf{n}.\mathsf{m}[\overline{\mathsf{PLUS}}\ \mathsf{n}]\overline{0},$$

the idea being that multiplying m and n is the same as adding n to 0 m times.

λ-Calculus: Church Numerals

The predecessesor function is more difficult:

$$\overline{\mathsf{PRED}} = \lambda n.\lambda f.\lambda x.n[\lambda g.\lambda h.h\ [g\ f]]\ [\lambda u.x]\ [\lambda u.u]$$

λ-Calculus: Church Numerals

The predecessesor function is more difficult:

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{h}.\mathsf{h}\ [\mathsf{g}\ \mathsf{f}]]\ [\lambda \mathsf{u}.\mathsf{x}]\ [\lambda \mathsf{u}.\mathsf{u}]$$

or alternatively

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{k}.[\mathsf{g}\ \overline{1}]\ [\lambda \mathsf{u}.\overline{\mathsf{PLUS}}\ [\mathsf{g}\ \mathsf{k}]\ \overline{1}]\ \mathsf{k}]\ [\lambda \mathsf{l}.\ \overline{0}]\ \overline{0}$$

λ-Calculus: Church Numerals

The predecessesor function is more difficult:

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{h}.\mathsf{h}\ [\mathsf{g}\ \mathsf{f}]]\ [\lambda \mathsf{u}.\mathsf{x}]\ [\lambda \mathsf{u}.\mathsf{u}]$$

or alternatively

$$\overline{\mathsf{PRED}} = \lambda \mathsf{n}.\mathsf{n}[\lambda \mathsf{g}.\lambda \mathsf{k}.[\mathsf{g}\ \overline{1}]\ [\lambda \mathsf{u}.\overline{\mathsf{PLUS}}\ [\mathsf{g}\ \mathsf{k}]\ \overline{1}]\ \mathsf{k}]\ [\lambda \mathsf{l}.\ \overline{0}]\ \overline{0}$$

Note the trick $[\mathsf{g}\overline{1}][\lambda \mathsf{u}.\overline{\mathsf{PLUS}}[\mathsf{g}\ \mathsf{k}]\ \overline{1}]\mathsf{k}$ which evaluates to k if $[\mathsf{g}\ \overline{1}]$ is $\overline{0}$ and to $[\mathsf{g}\ \mathsf{k}] + \overline{1}$ otherwise.

λ-Calculus: Sets

$$\{x|x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

λ-Calculus: Sets

$$\{x|x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A.\exists m, n.m \in A \wedge n \in A \wedge m \neq n$$

λ-Calculus: Sets

$$\{x | x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A. \exists m, n. m \in A \land n \in A \land m \neq n$$

In first-order, A can be 'defined' by:

$$[x \in A] \equiv [x^2 - 1 = 0]$$

λ-Calculus: Sets

$$\{x|x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A.\exists m, n.m \in A \wedge n \in A \wedge m \neq n$$

In first-order, A can be 'defined' by:

$$[x \in A] \equiv [x^2 - 1 = 0]$$

In this expression we talk about 'membership'

λ-Calculus: Sets

$$\{x|x^2 - 1 = 0\}$$

$$(\{-1, 1\})$$

The set A has two elements:

$$\exists A.\exists m, n.m \in A \wedge n \in A \wedge m \neq n$$

In first-order, A can be 'defined' by:

$$[x \in A] \equiv [x^2 - 1 = 0]$$

In this expression we talk about 'membership'

Alternatively, we can express the characteristic function of A by the λ-term

$$[\lambda x.[x^2 - 1 = 0]]$$

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

The idea is as follows

$$[[\lambda x.x^2 - 1 = 0]\, a] \text{ evaluates to } a^2 - 1 = 0$$

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

The idea is as follows

$[[\lambda x.x^2 - 1 = 0]\, a]$ evaluates to $a^2 - 1 = 0$

The expression $a^2 - 1 = 0$ is $\top$ ($\top$ denotes Truth) if a is -1 or 1.

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

The idea is as follows

$$[[\lambda x.x^2 - 1 = 0]\, a] \text{ evaluates to } a^2 - 1 = 0$$

The expression $a^2 - 1 = 0$ is $\top$ ($\top$ denotes Truth) if a is -1 or 1. Otherwise, $a^2 - 1 = 0$ is $\bot$ ($\bot$ denotes Falsehood)

λ-Calculus: Sets

$$[\lambda x.x^2 - 1 = 0]$$

The idea is as follows

$$[[\lambda x.x^2 - 1 = 0]\, a] \text{ evaluates to } a^2 - 1 = 0$$

The expression $a^2 - 1 = 0$ is $\top$ ($\top$ denotes Truth) if a is -1 or 1.
Otherwise, $a^2 - 1 = 0$ is $\bot$ ($\bot$ denotes Falsehood)
The characteristic function $[\lambda x.x^2 - 1 = 0]$ provides a witness for

$$\exists P.\exists m, n.\, [P\, m] \land [P\, n] \land m \neq n$$

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{\mathsf{n}} = \lambda \mathsf{f}.\lambda \mathsf{y}.[\mathsf{f}^{\mathsf{n}}\,\mathsf{y}]$$

We can also define the *set* $\overline{\mathsf{N}}$ of all Church numerals

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{\mathsf{n}} = \lambda \mathsf{f}.\lambda \mathsf{y}.[\mathsf{f}^{\mathsf{n}}\,\mathsf{y}]$$

We can also define the *set* $\overline{\mathsf{N}}$ of all Church numerals

$\overline{\mathsf{N}}$ must satisfy three properties:

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{\mathsf{n}} = \lambda \mathsf{f}.\lambda \mathsf{y}.[\mathsf{f}^{\mathsf{n}}\,\mathsf{y}]$$

We can also define the *set* $\overline{\mathsf{N}}$ of all Church numerals

$\overline{\mathsf{N}}$ must satisfy three properties:

1. $[\overline{\mathsf{N}}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

We can also define the *set* $\overline{N}$ of all Church numerals

$\overline{N}$ must satisfy three properties:

1. $[\overline{N}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"
2. $\forall x.[\overline{N}\, x] \supset [\overline{N}[\overline{SUCC}\, x]]$ "$\overline{N}$ is closed under successor"

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

We can also define the *set* $\overline{N}$ of all Church numerals

$\overline{N}$ must satisfy three properties:

1. $[\overline{N}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"
2. $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]]$ "$\overline{N}$ is closed under successor"
3. $\forall P.[P\,\overline{0}] \wedge [\forall x.[P\,x] \supset [P\,[\overline{SUCC}\,x]]] \supset [\overline{N} \subseteq P]$
   "$\overline{N}$ is the least such set"

λ-Calculus: Sets

For each natural number n there is a Church numeral:

$$\overline{n} = \lambda f.\lambda y.[f^n\, y]$$

We can also define the *set* $\overline{N}$ of all Church numerals
$\overline{N}$ must satisfy three properties:

1. $[\overline{N}\,\overline{0}]$ "$\overline{0}$ is a Church numeral"
2. $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]]$ "$\overline{N}$ is closed under successor"
3. $\forall P.[P\,\overline{0}] \wedge [\forall x.[P\,x] \supset [P\,[\overline{SUCC}\,x]]] \supset [\overline{N} \subseteq P]$
   "$\overline{N}$ is the least such set"

Define $\overline{N}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

- $[\overline{\mathsf{N}}\,\overline{0}]$ since $[P\,\overline{0}]$ implies $[P\,\overline{0}]$

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{\mathsf{SUCC}}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

- $[\overline{\mathsf{N}}\,\overline{0}]$ since $[P\,\overline{0}]$ implies $[P\,\overline{0}]$
- $\forall x.[\overline{\mathsf{N}}\,x] \supset [\overline{\mathsf{N}}[\overline{\mathsf{SUCC}}\,x]$ since if $P\,x$ and P is closed under successor, then $P\,[\overline{\mathsf{SUCC}}p]]$

λ-Calculus: Sets

Define $\overline{N}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \land [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

- $[\overline{N}\,\overline{0}]$ since $[P\,\overline{0}]$ implies $[P\,\overline{0}]$
- $\forall x.[\overline{N}\,x] \supset [\overline{N}[\overline{SUCC}\,x]$ since if $P\,x$ and P is closed under successor, then $P\,[\overline{SUCC}p]]$
- $\forall P.[P\,\overline{0}] \land [\forall x.[P\,x] \supset [P\,[\overline{SUCC}\,x]]] \supset [\overline{N} \subseteq P]$
  $\overline{N}$ is the least such set as the intersection of all such sets P

λ-Calculus: Sets

Define $\overline{\mathsf{N}}$ to be:

$$\lambda z.\forall P.[[P\,\overline{0}] \wedge [\forall x.\,[P\,x] \supset [P\,.\,\overline{SUCC}\,x]]] \supset [P\,z]$$

This satisfies the three requirements.

We have used quantification over sets (characteristic functions – the variable P) to define $\overline{\mathsf{N}}$.

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.

Actually it is so powerful that it is inconsistent!

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.

Actually it is so powerful that it is inconsistent!

Russell's paradox:

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

λ-Calculus: Russell's Paradox

Our representation framework is very powerful.
Actually it is so powerful that it is inconsistent!
Russell's paradox:
Consider the term R:

$$[\lambda x. \neg[x\,x]]$$

As a characteristic function, R represents the set of all sets which do not contain themselves:

$$\{x | x \notin x\}$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x. \neg [x\, x]]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$$[[\lambda x.\neg.x\ x]\ R]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$[[\lambda x.\neg.x\ x]\ R]$ evaluates to

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$$[[\lambda x.\neg.x\ x]\ R] \qquad \text{evaluates to} \qquad \neg[R\,R]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$[[\lambda x.\neg.x\ x]\ R]$ evaluates to $\neg[R\,R]$

And we evaluate $\neg[R\,R]$

$$\neg[[\lambda x.\neg.x\,x]\ R]$$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$[[\lambda x.\neg.x\ x]\ R]$ evaluates to $\neg[R\,R]$

And we evaluate $\neg[R\,R]$

$\neg[[\lambda x.\neg.x\,x]\ R]$ evaluates to

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$[[\lambda x.\neg.x\;x]\;R]$ evaluates to $\neg[R\,R]$

And we evaluate $\neg[R\,R]$

$\neg[[\lambda x.\neg.x\,x]\;R]$ evaluates to $\neg\neg[R\,R]$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $E := [R\,R]$

$$[[\lambda x.\neg.x\;x]\;R] \qquad \text{evaluates to} \qquad \neg[R\,R]$$

And we evaluate $\neg[R\,R]$

$$\neg[[\lambda x.\neg.x\,x]\;R] \qquad \text{evaluates to} \qquad \neg\neg[R\,R]$$

which is equivalent to $[R\,R]$

λ-Calculus: Russell's Paradox

Consider the term R:

$$[\lambda x.\neg[x\,x]]$$

Now we evaluate the expression $\mathsf{E} := [\mathsf{R}\,\mathsf{R}]$

$$[[\lambda x.\neg.x\;x]\;\mathsf{R}] \qquad \text{evaluates to} \qquad \neg[\mathsf{R}\,\mathsf{R}]$$

And we evaluate $\neg[\mathsf{R}\,\mathsf{R}]$

$$\neg[[\lambda x.\neg.x\,x]\;\mathsf{R}] \qquad \text{evaluates to} \qquad \neg\neg[\mathsf{R}\,\mathsf{R}]$$

which is equivalent to $[\mathsf{R}\,\mathsf{R}]$

Thus if E holds we can infer $\neg\mathsf{E}$ and vice versa. This is Russell's paradox.

λ-Calculus: Nontermination

Note that the term $[\lambda x.\neg.x\,x]$ (just as the standard example $[\lambda x.x\,x]$) does not terminate with respect to β-reduction:

$$[R\,R] \longrightarrow_\beta \neg[R\,R] \longrightarrow_\beta \neg\neg[R\,R] \longrightarrow_\beta \ldots$$

Typed λ-Calculus

We can avoid Russell's paradox using simple types.

Typed λ-Calculus

We can avoid Russell's paradox using simple types.

Simple Types:

- o Base type of propositions

Typed λ-Calculus

We can avoid Russell's paradox using simple types.

Simple Types:

- o Base type of propositions
- ι Base type of individuals

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

One may include arbitrarily many base types $\iota^1, \ldots, \iota^n, \ldots$.

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \rightarrow \alpha)$) Type of functions from β to α

We often omit parenthesis in types. $(\alpha\beta\gamma)$ means $((\alpha\beta)\gamma)$

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \rightarrow \alpha)$) Type of functions from β to α

We often omit parenthesis in types. $(\alpha\beta\gamma)$ means $((\alpha\beta)\gamma)$
Likewise $(\gamma \rightarrow \beta \rightarrow \alpha)$ means $(\gamma \rightarrow (\beta \rightarrow \alpha))$

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

We often omit parenthesis in types. $(\alpha\beta\gamma)$ means $((\alpha\beta)\gamma)$
Likewise $(\gamma \to \beta \to \alpha)$ means $(\gamma \to (\beta \to \alpha))$
Note that the type $(\alpha\beta\gamma)$ (or $(\gamma \to \beta \to \alpha)$) is the type of a (Curried) function of two arguments which returns a value of type α.

Typed λ-Calculus: Typed Terms

- Typed Variables x_α

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[\mathsf{F}_{\alpha\beta}\mathsf{B}_\beta]_\alpha$ – or $[\mathsf{F}_{\beta\rightarrow\alpha}\mathsf{B}_\beta]_\alpha$

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[\mathsf{F}_{\alpha\beta}\mathsf{B}_\beta]_\alpha$ – or $[\mathsf{F}_{\beta\to\alpha}\mathsf{B}_\beta]_\alpha$
- λ-abstraction $[\lambda \mathsf{y}_\beta.\, \mathsf{A}_\alpha]_{\alpha\beta}$ – or $[\lambda \mathsf{y}_\beta.\, \mathsf{A}_\alpha]_{\beta\to\alpha}$

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[F_{\alpha\beta} B_\beta]_\alpha$ – or $[F_{\beta\rightarrow\alpha} B_\beta]_\alpha$
- λ-abstraction $[\lambda y_\beta . A_\alpha]_{\alpha\beta}$ – or $[\lambda y_\beta . A_\alpha]_{\beta\rightarrow\alpha}$

Examples:

- $[\lambda x_\alpha . x_\alpha]$ term of type $(\alpha\alpha)$ – identity on type α

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[F_{\alpha\beta} B_\beta]_\alpha$ – or $[F_{\beta\rightarrow\alpha} B_\beta]_\alpha$
- λ-abstraction $[\lambda y_\beta.\, A_\alpha]_{\alpha\beta}$ – or $[\lambda y_\beta.\, A_\alpha]_{\beta\rightarrow\alpha}$

Examples:

- $[\lambda x_\alpha.\, x_\alpha]$ term of type $(\alpha\alpha)$ – identity on type α
- $[\lambda y_\beta.\, x_\alpha]$ term of type $(\alpha\beta)$ – constant x-valued function

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\ [\mathsf{MINUS}\ [\mathsf{SQUARE}\, x]\ 1]]$$

where MINUS, SQUARE and 1 are constants.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\ [\mathsf{MINUS}\ [\mathsf{SQUARE}\ x]\ 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\ [\mathsf{MINUS}\ [\mathsf{SQUARE}\ x]\ 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

This is shorthand for

$$[\lambda \mathsf{x}.\ [\mathsf{MINUS}\ [\mathsf{SQUARE}\ \mathsf{x}]\ 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

- x and 1 should be real numbers (type ι)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

This is shorthand for

$$[\lambda \mathsf{x}.\,[\mathsf{MINUS}\,[\mathsf{SQUARE}\,\mathsf{x}]\,1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

- x and 1 should be real numbers (type ι)
- SQUARE should take a real number to a real number (type $(\iota\iota)$)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

This is shorthand for

$$[\lambda \mathsf{x}.\,[\mathsf{MINUS}\,[\mathsf{SQUARE}\,\mathsf{x}]\,1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

- x and 1 should be real numbers (type ι)
- SQUARE should take a real number to a real number (type $(\iota\iota)$)
- MINUS should take two real numbers to a real number (type $(\iota\iota\iota)$)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\,[\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

Typed Term:

$$[\lambda x_\iota.\,[\mathsf{MINUS}_{\iota\iota\iota}\,[\mathsf{SQUARE}_{\iota\iota}\,x_\iota]\,1_\iota]]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\,[\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

Typed Term:

$$[\lambda x_\iota.\,[\mathsf{MINUS}_{\iota\iota\iota}\,[\mathsf{SQUARE}_{\iota\iota}\,x_\iota]\,1_\iota]]$$

This term has type $(\iota\iota)$.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\, [x^2 - 1 = 0]]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x. [x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x . [= \ [\mathsf{MINUS}\, [\mathsf{SQUARE}\, x]\, 1]\, 0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\, [x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[= \; [\mathsf{MINUS}\, [\mathsf{SQUARE}\, x]\, 1]\, 0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

- Already know types of MINUS, SQUARE and 1.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\,[x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

- Already know types of MINUS, SQUARE and 1.
- 0 should be a real number (type ι)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\,[x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

- Already know types of MINUS, SQUARE and 1.
- 0 should be a real number (type ι)
- $=$ takes two real numbers and returns a truth value (type $(o\iota\iota)$)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\,[x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathrm{MINUS}\,[\mathrm{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.
Typed Term:

$$[\lambda x_\iota.\,[=_{o\iota\iota}\ [\mathrm{MINUS}_{\iota\iota\iota}\,[\mathrm{SQUARE}_{\iota\iota}\,x_\iota]\,1_\iota]\,0_\iota]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\,[x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[= \;[\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

Typed Term:

$$[\lambda x_\iota.\,[=_{o\iota\iota}\;[\mathsf{MINUS}_{\iota\iota\iota}\,[\mathsf{SQUARE}_{\iota\iota}\,x_\iota]\,1_\iota]\,0_\iota]$$

This term has type $(o\iota)$.

Typed λ-Calculus: Assigning Types

General algorithm for assigning types to terms (when this is possible) – see Hindley97.

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{C : \alpha \in \Gamma \quad C \text{ variable, parameter or constant}}{\Gamma \vdash_{TA} C : \alpha}\ \text{Hyp}$$

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{\mathsf{C} : \alpha \in \Gamma \quad \mathsf{C}\ \text{variable, parameter or constant}}{\Gamma \vdash_{\mathsf{TA}} \mathsf{C} : \alpha}\ \mathsf{Hyp}$$

$$\frac{\Gamma, \mathsf{y} : \beta \vdash_{\mathsf{TA}} \mathsf{A} : \alpha}{\Gamma \vdash_{\mathsf{TA}} [\lambda \mathsf{y}.\, \mathsf{A}] : \alpha\beta}\ \mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{\mathsf{C} : \alpha \in \Gamma \quad \mathsf{C} \text{ variable, parameter or constant}}{\Gamma \vdash_{\mathsf{TA}} \mathsf{C} : \alpha}\ \mathsf{Hyp}$$

$$\frac{\Gamma, \mathsf{y} : \beta \vdash_{\mathsf{TA}} \mathsf{A} : \alpha}{\Gamma \vdash_{\mathsf{TA}} [\lambda \mathsf{y}.\, \mathsf{A}] : \alpha\beta}\ \mathsf{Lam} \qquad \frac{\Gamma \vdash_{\mathsf{TA}} \mathsf{F} : \alpha\beta \quad \Gamma \vdash_{\mathsf{TA}} \mathsf{B} : \beta}{\Gamma \vdash_{\mathsf{TA}} [\mathsf{F}\,\mathsf{B}] : \alpha}\ \mathsf{App}$$

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{\mathsf{C} : \alpha \in \Gamma \quad \mathsf{C}\ \text{variable, parameter or constant}}{\Gamma \vdash_{\mathsf{TA}} \mathsf{C} : \alpha}\ \mathsf{Hyp}$$

$$\frac{\Gamma, \mathsf{y} : \beta \vdash_{\mathsf{TA}} \mathsf{A} : \alpha}{\Gamma \vdash_{\mathsf{TA}} [\lambda \mathsf{y}.\, \mathsf{A}] : \alpha\beta}\ \mathsf{Lam} \qquad \frac{\Gamma \vdash_{\mathsf{TA}} \mathsf{F} : \alpha\beta \quad \Gamma \vdash_{\mathsf{TA}} \mathsf{B} : \beta}{\Gamma \vdash_{\mathsf{TA}} [\mathsf{F}\,\mathsf{B}] : \alpha}\ \mathsf{App}$$

We can assign the type α to a term A in context Γ whenever we can derive

$$\Gamma \vdash_{\mathsf{TA}} \mathsf{A} : \alpha$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]] : \alpha$

$$\begin{array}{c} \vdots \\ \mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]] : \alpha \end{array}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x. [\mathsf{SQUARE}\, x]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \alpha$

α is $(\gamma\beta)$

$$\dfrac{\begin{matrix}\vdots \\ \mathsf{SQUARE} : (\iota\iota), x : \beta \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\, x] : \gamma\end{matrix}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \gamma\beta}\ \mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\, [\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\gamma\delta)\end{array} \qquad \begin{array}{c}\vdots\\ \mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{x} : \delta\end{array}}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\,\mathsf{x}] : \gamma}\,\mathsf{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\,\mathsf{x}]] : \gamma\beta}\,\mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x. [\mathsf{SQUARE}\, x]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \alpha$

γ and δ are both ι

$$\dfrac{\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), x : \beta \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathrm{Hyp} \qquad \begin{array}{c}\vdots\\ \mathsf{SQUARE} : (\iota\iota), x : \beta \vdash_{\mathsf{TA}} x : \iota\end{array}}{\mathsf{SQUARE} : (\iota\iota), x : \beta \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\, x] : \iota}\,\mathrm{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \iota\beta}\,\mathrm{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x. [\mathsf{SQUARE}\, x]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \alpha$

β is ι

$$\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathsf{Hyp} \quad \dfrac{}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} x : \iota}\,\mathsf{Hyp}}{\dfrac{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\, x] : \iota}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \iota\iota}\,\mathsf{Lam}}\,\mathsf{App}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x. [\mathsf{SQUARE}\, x]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \alpha$

β is ι

$$\dfrac{\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathsf{Hyp} \quad \dfrac{}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} x : \iota}\,\mathsf{Hyp}}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\, x] : \iota}\,\mathsf{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \iota\iota}\,\mathsf{Lam}$$

So $[\lambda x. [\mathsf{SQUARE}\, x]]$ can be assigned the type $(\iota\iota)$ in context $\mathsf{SQUARE} : (\iota\iota)$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x. [\mathsf{SQUARE}\, x]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \alpha$

β is ι

$$\dfrac{\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathsf{Hyp} \quad \dfrac{}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} x : \iota}\,\mathsf{Hyp}}{\mathsf{SQUARE} : (\iota\iota), x : \iota \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\, x] : \iota}\,\mathsf{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda x. [\mathsf{SQUARE}\, x]] : \iota\iota}\,\mathsf{Lam}$$

So $[\lambda x. [\mathsf{SQUARE}\, x]]$ can be assigned the type $(\iota\iota)$ in context $\mathsf{SQUARE} : (\iota\iota)$

Corresponding Typed Term: $[\lambda x_\iota. [\mathsf{SQUARE}_{\iota\iota}\, x_\iota]]$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x \,.\, \neg\, [x\, x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x \,.\, \neg\, [x\, x]] : \alpha$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x . \neg [x\,x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x . \neg [x\,x]] : \alpha$

$$\begin{array}{c} \vdots \\ \neg : (oo) \vdash_{TA} [\lambda x . \neg [x\,x]] : \alpha \end{array}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x\,.\,\neg\,[x\,x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x\,.\,\neg\,[x\,x]] : \alpha$

α is $(\gamma\beta)$

$$\dfrac{\begin{array}{c}\vdots\\ \neg : (oo), x : \beta \vdash_{TA} [\neg\,[x\,x]] : \gamma\end{array}}{\neg : (oo) \vdash_{TA} [\lambda x\,.\,\neg\,[x\,x]] : \gamma\beta}\,\text{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x . \neg [x\, x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x . \neg [x\, x]] : \alpha$

$$
\dfrac{\dfrac{\dfrac{\vdots}{\neg : (oo), x : \beta \vdash_{TA} \neg : (\gamma\delta)} \qquad \dfrac{\vdots}{\neg : (oo), x : \beta \vdash_{TA} [x\, x] : \delta}}{\neg : (oo), x : \beta \vdash_{TA} [\neg [x\, x]] : \gamma}\,\text{App}}{\neg : (oo) \vdash_{TA} [\lambda x. [\neg [x\, x]] : \gamma\beta}\,\text{Lam}
$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x . \neg [x\, x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x . \neg [x\, x]] : \alpha$

γ and δ are both o

$$
\dfrac{\dfrac{}{\neg : (oo), x : \beta \vdash_{TA} \neg : (oo)}\,\mathrm{Hyp} \qquad \begin{array}{c}\vdots\\ \neg : (oo), x : \beta \vdash_{TA} [x\, x] : o\end{array}}{\dfrac{\neg : (oo), x : \beta \vdash_{TA} [\neg [x\, x]] : o}{\neg : (oo) \vdash_{TA} [\lambda x . [\neg [x\, x]] : o\beta}\,\mathrm{Lam}}\,\mathrm{App}
$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x . \neg [x\,x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x . \neg [x\,x]] : \alpha$

$$\vdots$$

$$\neg : (oo), x : \beta \vdash_{TA} [x\,x] : o$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x . \neg [x\, x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x . \neg [x\, x]] : \alpha$

$$\dfrac{\dfrac{\vdots}{\neg : (oo), x : \beta \vdash_{TA} x : (o\epsilon)} \qquad \dfrac{\vdots}{\neg : (oo), x : \beta \vdash_{TA} x : \epsilon}}{\neg : (oo), x : \beta \vdash_{TA} [x\, x] : o}\ \text{App}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]] : \alpha$

β is $(\mathsf{o}\epsilon)$

$$\dfrac{\dfrac{}{\neg : (\mathsf{oo}), \mathsf{x} : (\mathsf{o}\epsilon) \vdash_{\mathsf{TA}} \mathsf{x} : (\mathsf{o}\epsilon)}\,\mathsf{Hyp} \qquad \begin{array}{c}\vdots \\ \neg : (\mathsf{oo}), \mathsf{x} : (\mathsf{o}\epsilon) \vdash_{\mathsf{TA}} \mathsf{x} : \epsilon\end{array}}{\neg : (\mathsf{oo}), \mathsf{x} : (\mathsf{o}\epsilon) \vdash_{\mathsf{TA}} [\mathsf{x}\,\mathsf{x}] : \mathsf{o}}\,\mathsf{App}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x . \neg [x\, x]]$
Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x . \neg [x\, x]] : \alpha$
Only remaining subgoal:

$$\neg : (oo), x : (o\epsilon) \vdash_{TA} x : \epsilon$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x} \,.\, \neg\, [\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x} \,.\, \neg\, [\mathsf{x}\,\mathsf{x}]] : \alpha$

Only remaining subgoal:

$$\neg : (\mathsf{oo}), \mathsf{x} : (\mathsf{o}\epsilon) \vdash_{\mathsf{TA}} \mathsf{x} : \epsilon$$

This goal cannot be solved since $(\mathsf{o}\epsilon)$ cannot equal ϵ.

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x\,.\,\neg\,[x\,x]]$
Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x\,.\,\neg\,[x\,x]] : \alpha$
Only remaining subgoal:

$$\neg : (oo), x : (o\epsilon) \vdash_{TA} x : \epsilon$$

This goal cannot be solved since $(o\epsilon)$ cannot equal ϵ.

Hence $[\lambda x.\,[\neg\,[x\,x]]]$ cannot be typed – avoiding Russell's Paradox.

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda y_\beta . A_\alpha]\ B_\beta] \longrightarrow_\beta A_\alpha[y_\beta / B_\beta]$$

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda \mathsf{y}_\beta \,.\, \mathsf{A}_\alpha]\ \mathsf{B}_\beta] \longrightarrow_\beta \mathsf{A}_\alpha[\mathsf{y}_\beta/\mathsf{B}_\beta]$$

η-reduction:

$$[\lambda \mathsf{y}_\beta . \mathsf{F}_{\alpha\beta}\ \mathsf{y}_\beta] \longrightarrow_\eta \mathsf{F}_{\alpha\beta}$$

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda \mathsf{y}_\beta \,.\, \mathsf{A}_\alpha]\; \mathsf{B}_\beta] \longrightarrow_\beta \mathsf{A}_\alpha[\mathsf{y}_\beta/\mathsf{B}_\beta]$$

η-reduction:

$$[\lambda \mathsf{y}_\beta . \mathsf{F}_{\alpha\beta}\, \mathsf{y}_\beta] \longrightarrow_\eta \mathsf{F}_{\alpha\beta}$$

Facts:

- $\beta\eta$-normalization terminates for typed terms.

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda \mathsf{y}_\beta \,.\, \mathsf{A}_\alpha]\ \mathsf{B}_\beta] \longrightarrow_\beta \mathsf{A}_\alpha[\mathsf{y}_\beta/\mathsf{B}_\beta]$$

η-reduction:

$$[\lambda \mathsf{y}_\beta.\mathsf{F}_{\alpha\beta}\,\mathsf{y}_\beta] \longrightarrow_\eta \mathsf{F}_{\alpha\beta}$$

Facts:

- $\beta\eta$-normalization terminates for typed terms.
- Every typed term has a unique $\beta\eta$-normal form.

Introduction (Contd.)

Typed λ-Calculus: Logical Constants

We gain expressive power by combining typed λ-calculus with logical constants.

Typed λ-Calculus: Logical Constants

We gain expressive power by combining typed λ-calculus with logical constants.

$\top_o$ – true

$\bot_o$ – false

$\neg_{oo}$ – negation

$\vee_{ooo}$ – disjunction

$\wedge_{ooo}$ – conjunction

$\supset_{ooo}$ – implication

$\equiv_{ooo}$ – equivalence

Typed λ-Calculus: Logical Constants

We gain expressive power by combining typed λ-calculus with logical constants.

$=^{\alpha}_{o\alpha\alpha}$ – equality at type α

$\Pi^{\alpha}_{o(o\alpha)}$ – universal quantification over type α

$\Sigma^{\alpha}_{o(o\alpha)}$ – existential quantification over type α

Intuition: $[\Sigma^{\alpha} . \lambda x_{\alpha} . C_o]$ is to true iff $\{x_{\alpha} | C\}$ is nonempty.

Church's Classical Type Theory: HOL

HOL: Abbreviations

$[A_o \lor B_o]$ means $[\lor_{ooo} A_o B_o]$

$[A_o \land B_o]$ means $[\land_{ooo} A_o B_o]$

$[A_o \supset B_o]$ means $[\supset_{ooo} A_o B_o]$

$[A_o \equiv B_o]$ means $[\equiv_{ooo} A_o B_o]$

$[A_\alpha =^\alpha B_\alpha]$ means $[=^\alpha_{o\alpha\alpha} A_\alpha B_\alpha]$

$[\forall x_\alpha . A_o]$ means $[\Pi^\alpha_{o(o\alpha)} . \lambda x_\alpha . A_o]$.

$[\exists x_\alpha . A_o]$ means $[\Sigma^\alpha_{o(o\alpha)} . \lambda x_\alpha . A_o]$.

HOL: Expressing Properties

$$[\lambda x_\iota . x^2 - 1]$$

HOL: Expressing Properties

$$[\lambda \mathsf{x}_\iota . \mathsf{x}^2 - 1]$$

$$[\lambda \mathsf{x}_\iota . [\mathsf{MINUS}_{\iota\iota\iota}\ [\mathsf{SQUARE}_{\iota\iota}\ \mathsf{x}] 1_\iota]]_{\iota\iota}$$

HOL: Expressing Properties

$$[\lambda x_\iota . x^2 - 1]$$

Term of type o expressing existence of an f with two roots:

$$[\exists f_{\iota\iota} . \exists n_\iota . \exists m_\iota . [[f\ n] =^\iota 0_\iota] \wedge [[f\ m] =^\iota 0_\iota] \wedge \neg[n =^\iota m]]_o$$

HOL: Expressing Properties

$$[\lambda x_\iota . x^2 - 1]$$

Term of type o expressing existence of an f with two roots:

$$[\underbrace{\exists f_{\iota\iota}}_{\Sigma^{\iota\iota}\lambda f_{\iota\iota}} . \exists n_\iota . \exists m_\iota . [[f\ n] =^\iota 0_\iota] \wedge [[f\ m] =^\iota 0_\iota] \wedge \neg [n =^\iota m]]_o$$

HOL: Expressing Properties

$$[\lambda x_\iota . x^2 - 1]$$

Term of type o expressing existence of an f with two roots:

$$[\exists f_{\iota\iota} . \exists n_\iota . \exists m_\iota . \underbrace{[[f\ n] =^\iota 0_\iota]}_{[=^\iota [f\ n]\ 0]} \wedge [[f\ m] =^\iota 0_\iota] \wedge \neg[n =^\iota m]]_o$$

HOL: Expressing Properties

$$[\lambda x_\iota . x^2 - 1]$$

$$[\lambda x_\iota . [x^2 - 1] = 0]$$

HOL: Expressing Properties

$$[\lambda x_\iota . x^2 - 1]$$

$$[\lambda x_\iota . [x^2 - 1] = 0]$$

$$[\lambda x_\iota . [=^\iota \; [\mathsf{MINUS}_{\iota\iota\iota} \, [\mathsf{SQUARE}_{\iota\iota} \, x] 1_\iota] \, 0_\iota]]_{o\iota}$$

HOL: Expressing Properties

$$[\lambda x_\iota . x^2 - 1]$$

$$[\lambda x_\iota . [x^2 - 1] = 0]$$

Term of type o expressing existence of a set (characteristic function) P with two elements

$$[\exists P_{o\iota} . \exists m_\iota . \exists n_\iota . [P\, m] \wedge [P\, n] \wedge \neg [m = n]]_o$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.

Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$

We can give the usual $\epsilon - \delta$ definition of limits.

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda f_{\iota\iota}.\lambda a_{\iota}.\lambda L_{\iota}.\forall \epsilon_{\iota}.[\epsilon > 0] \supset .\exists \delta_{\iota}.[\delta > 0]$$
$$\wedge .\forall x_{\iota}.[|x - a| < \delta] \supset [|[f\,x] - L| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda \mathsf{f}_{\iota\iota}.\lambda \mathsf{a}_{\iota}.\lambda \mathsf{L}_{\iota}.\forall \epsilon_{\iota}.\overbrace{[\epsilon > 0]}^{[<0\,\epsilon]} \supset .\exists \delta_{\iota}.[\delta > 0]$$

$$\wedge .\forall \mathsf{x}_{\iota}.[|\mathsf{x} - \mathsf{a}| < \delta] \supset [|[\mathsf{f}\,\mathsf{x}] - \mathsf{L}| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda \mathsf{f}_{\iota\iota}.\lambda \mathsf{a}_{\iota}.\lambda \mathsf{L}_{\iota}.\forall \epsilon_{\iota}.[\epsilon > 0] \supset .\exists \delta_{\iota}.\overbrace{[\delta > 0]}^{[<0\,\delta]}$$
$$\wedge .\forall \mathsf{x}_{\iota}.[|\mathsf{x} - \mathsf{a}| < \delta] \supset [|[\mathsf{f}\,\mathsf{x}] - \mathsf{L}| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda \mathsf{f}_{\iota\iota}.\lambda \mathsf{a}_\iota.\lambda \mathsf{L}_\iota.\forall \epsilon_\iota.[\epsilon > 0] \supset .\exists \delta_\iota.[\delta > 0]$$
$$\wedge .\forall \mathsf{x}_\iota.[|\mathsf{x} - \mathsf{a}| < \delta] \supset [|[\mathsf{f}\,\mathsf{x}] - \mathsf{L}| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota(\iota\iota)}$:

$$[\lambda f_{\iota\iota}.\lambda a_{\iota}.\lambda L_{\iota}.\forall \epsilon_{\iota}.[\epsilon > 0] \supset .\exists \delta_{\iota}.[\delta > 0]$$
$$\wedge .\forall x_{\iota}.[|\underbrace{x - a}_{[\mathsf{MINUS}\, x\, a]}| < \delta] \supset [|[f\, x] - L| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda f_{\iota\iota}.\lambda a_\iota.\lambda L_\iota.\forall\epsilon_\iota.[\epsilon > 0] \supset .\exists\delta_\iota.[\delta > 0]$$
$$\wedge .\forall x_\iota.[\underbrace{|x - a|}_{[\mathsf{ABS}\,[\mathsf{MINUS}\,x\,a]]} < \delta] \supset [|[f\,x] - L| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda \mathsf{f}_{\iota\iota}.\lambda \mathsf{a}_{\iota}.\lambda \mathsf{L}_{\iota}.\forall \epsilon_{\iota}.[\epsilon > 0] \supset .\exists \delta_{\iota}.[\delta > 0]$$

$$\wedge .\forall \mathsf{x}_{\iota}. \quad \underbrace{[|\mathsf{x} - \mathsf{a}| < \delta]}_{[< [\mathsf{ABS}\, [\mathsf{MINUS}\, \mathsf{x}\, \mathsf{a}]]\, \delta]} \quad \supset [|[\mathsf{f}\, \mathsf{x}] - \mathsf{L}| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda \mathsf{f}_{\iota\iota}.\lambda \mathsf{a}_{\iota}.\lambda \mathsf{L}_{\iota}.\forall \epsilon_{\iota}.[\epsilon > 0] \supset .\exists \delta_{\iota}.[\delta > 0]$$
$$\wedge .\forall \mathsf{x}_{\iota}.[|\mathsf{x} - \mathsf{a}| < \delta] \supset [|[\mathsf{f}\,\mathsf{x}] - \mathsf{L}| < \epsilon]]$$

HOL: Expressing Properties

Suppose ι corresponds to real numbers.
Given constants: $<_{o\iota\iota}$, $\mathsf{ABS}_{\iota\iota}$, $\mathsf{MINUS}_{\iota\iota\iota}$
We can give the usual $\epsilon - \delta$ definition of limits.
$\mathsf{LIM}_{o\iota\iota(\iota\iota)}$:

$$[\lambda f_{\iota\iota}.\lambda a_\iota.\lambda L_\iota.\forall \epsilon_\iota.[\epsilon > 0] \supset .\exists \delta_\iota.[\delta > 0] \\ \wedge .\forall x_\iota.[|x - a| < \delta] \supset [|[f\,x] - L| < \epsilon]]$$

Similarly can define continuity, differentiation, etc.

HOL: Prefix Polymorphism

Some definitions are naturally expressed using type variables:

HOL: Prefix Polymorphism

Some definitions are naturally expressed using type variables:

Consider the notion of subset:

For each type α we can define $\subseteq_{o(o\alpha)(o\alpha)}$ to be:

$$\lambda X_{o\alpha}.\, \lambda Y_{o\alpha}.\, [\forall z_{\alpha}.\, [X\ z] \supset [Y\ z]]$$

HOL: Prefix Polymorphism

Some definitions are naturally expressed using type variables:
Consider the notion of subset:
For each type α we can define $\subseteq_{o(o\alpha)(o\alpha)}$ to be:

$$\lambda X_{o\alpha}.\, \lambda Y_{o\alpha}.\, [\forall z_\alpha.\, [X\ z] \supset [Y\ z]]$$

We can think of α as a type variable and $\subseteq_{o(o\alpha)(o\alpha)}$ to be polymorphic.

HOL: Prefix Polymorphism

Some definitions are naturally expressed using type variables:
Consider the notion of subset:
For each type α we can define $\subseteq_{o(o\alpha)(o\alpha)}$ to be:

$$\lambda X_{o\alpha}.\, \lambda Y_{o\alpha}.\, [\forall z_\alpha.\, [X\ z] \supset [Y\ z]]$$

We can think of α as a type variable and $\subseteq_{o(o\alpha)(o\alpha)}$ to be polymorphic. In any particular occurrence of $\subseteq_{o(o\alpha)(o\alpha)}$ we should be able to instantiate the type variable α.

HOL: Prefix Polymorphism

Some definitions are naturally expressed using type variables:
Consider the notion of subset:
For each type α we can define $\subseteq_{o(o\alpha)(o\alpha)}$ to be:

$$\lambda X_{o\alpha}.\,\lambda Y_{o\alpha}.\,[\forall z_{\alpha}.\,[X\ z] \supset [Y\ z]]$$

We can think of α as a type variable and $\subseteq_{o(o\alpha)(o\alpha)}$ to be polymorphic. In any particular occurrence of $\subseteq_{o(o\alpha)(o\alpha)}$ we should be able to instantiate the type variable α.
Example: (using infix notation)

$$[\lambda U_{o\iota}.\,[U \subseteq_{o(o\iota)(o\iota)} X_{o\iota}]] \subseteq_{o(o(o\iota))(o(o\iota))} [\lambda U_{o\iota}.\,[U \subseteq_{o(o\iota)(o\iota)} Y_{o\iota}]]$$

HOL: Cantor's Theorem

There is no surjection from a set A onto the power set $\mathcal{P}(\mathrm{A})$ of A.

HOL: Cantor's Theorem

There is no surjection from a set A onto the power set $\mathcal{P}(\mathrm{A})$ of A.

- Suppose A corresponds to type ι.

HOL: Cantor's Theorem

There is no surjection from a set A onto the power set $\mathcal{P}(\mathrm{A})$ of A.

- Suppose A corresponds to type ι.
- Then $\mathcal{P}(\mathrm{A})$ corresponds to type $(o\iota)$.

HOL: Cantor's Theorem

There is no surjection from a set A onto the power set $\mathcal{P}(A)$ of A.

- Suppose A corresponds to type ι.
- Then $\mathcal{P}(A)$ corresponds to type $(o\iota)$.

$$\neg \exists\, g_{o\iota\iota} . \forall\, f_{o\iota} . \exists\, x_\iota \,.\, g\, x =^{o\iota} f$$

HOL: Standard Higher-Order Model

$\mathcal{D}_\iota$ (individuals)

HOL: Standard Higher-Order Model

$\mathcal{P}(\mathcal{D}_\iota)$	(all sets)
$\mathcal{D}_\iota$	(individuals)

HOL: Standard Higher-Order Model

(all sets of sets)

$\mathcal{P}(\mathcal{P}(\mathcal{D}_\iota))$

$\mathcal{P}(\mathcal{D}_\iota)$ (all sets)

$\mathcal{D}_\iota$ (individuals)

HOL: Standard Higher-Order Model

$$\vdots$$

$$\mathcal{P}(\mathcal{P}(\mathcal{D}_\iota))$$

$$\mathcal{P}(\mathcal{D}_\iota)$$

$$\mathcal{D}_\iota$$

HOL: Henkin-Style Model

$$\mathcal{D}_{o\iota} \subseteq \mathcal{P}(\mathcal{D}_\iota) \quad \text{(some sets)}$$

$$\mathcal{D}_\iota \quad \text{(individuals)}$$

HOL: Henkin-Style Model

(some sets of sets)

$\mathcal{D}_{\mathsf{o}(\mathsf{o}\iota)} \subseteq \mathcal{P}(\mathcal{D}_{\mathsf{o}\iota})$

$\mathcal{D}_{\mathsf{o}\iota} \subseteq \mathcal{P}(\mathcal{D}_{\iota})$ (some sets)

$\mathcal{D}_{\iota}$ (individuals)

HOL: Henkin-Style Model

$$\vdots$$

$$\mathcal{D}_{o(o\iota)} \subseteq \mathcal{P}(\mathcal{D}_{o\iota})$$

$$\mathcal{D}_{o\iota} \subseteq \mathcal{P}(\mathcal{D}_{\iota})$$

$$\mathcal{D}_{\iota}$$

Types, Frames, and Applicative Structures

Def.: Types

Let $\mathcal{T}$ be the least set s.t:

$$\mathsf{o} \in \mathcal{T}$$

$$\iota \in \mathcal{T}$$

$$\forall \alpha, \beta \in \mathcal{T} : (\alpha\beta) \in \mathcal{T}$$

Def.: Types

Let $\mathcal{T}$ be the least set s.t:

$$\mathsf{o} \in \mathcal{T}$$

$$\iota \in \mathcal{T}$$

$$\forall \alpha, \beta \in \mathcal{T} : (\alpha\beta) \in \mathcal{T}$$

We say that $\alpha \in \mathcal{T}$ is a simple type (or type).
$(\alpha\beta)$ is called a function type.

Def.: Types

Let $\mathcal{T}$ be the least set s.t:

$$\mathsf{o} \in \mathcal{T}$$

$$\iota \in \mathcal{T}$$

$$\forall \alpha, \beta \in \mathcal{T} : (\alpha\beta) \in \mathcal{T}$$

We say that $\alpha \in \mathcal{T}$ is a simple type (or type).
$(\alpha\beta)$ is called a function type.

- The set $\mathcal{T}$ is defined inductively.
- The set $\mathcal{T}$ is "freely generated".

Ex.: Freely Generated

Consider the set $\mathbb{N} = \{0, 1, 2...\}$.

- $0 \in \mathbb{N}$

Ex.: Freely Generated

Consider the set $\mathbb{N} = \{0, 1, 2 \ldots\}$.

- $0 \in \mathbb{N}$
- $\forall \mathsf{n} \in \mathbb{N} : \mathsf{s}(\mathsf{n}) \in \mathbb{N}$.

Ex.: Freely Generated

Consider the set $\mathbb{N} = \{0, 1, 2 \ldots\}$.

- $0 \in \mathbb{N}$
- $\forall n \in \mathbb{N} : s(n) \in \mathbb{N}$.
- $\forall n : 0 \neq s(n)$.

Ex.: Freely Generated

Consider the set $\mathbb{N} = \{0, 1, 2 \ldots\}$.

- $0 \in \mathbb{N}$
- $\forall \mathsf{n} \in \mathbb{N} : \mathsf{s}(\mathsf{n}) \in \mathbb{N}$.
- $\forall \mathsf{n} : 0 \neq \mathsf{s}(\mathsf{n})$.
- $\forall \mathsf{m}, \mathsf{n} : \mathsf{s}(\mathsf{m}) = \mathsf{s}(\mathsf{n}) \Rightarrow \mathsf{m} = \mathsf{n}$.

Ex.: Freely Generated

Consider the set $\mathbb{N} = \{0, 1, 2...\}$.

- $0 \in \mathbb{N}$
- $\forall n \in \mathbb{N} : s(n) \in \mathbb{N}$.
- $\forall n : 0 \neq s(n)$.
- $\forall m, n : s(m) = s(n) \Rightarrow m = n$.

The set $\mathbb{N}$ is "freely generated".

Ex.: Freely Generated

Consider the set $\mathbb{N} = \{0, 1, 2...\}$.

- $0 \in \mathbb{N}$
- $\forall \mathsf{n} \in \mathbb{N} : \mathsf{s}(\mathsf{n}) \in \mathbb{N}$.
- $\forall \mathsf{n} : 0 \neq \mathsf{s}(\mathsf{n})$.
- $\forall \mathsf{m}, \mathsf{n} : \mathsf{s}(\mathsf{m}) = \mathsf{s}(\mathsf{n}) \Rightarrow \mathsf{m} = \mathsf{n}$.

The set $\mathbb{N}$ is "freely generated".

Contrast $\mathbb{N}$ to $\mathbb{Z} = \{..., -1, 0, 1, ...\}$.
Note that $\mathbb{Z}$ contains 0 and is closed under successor, but is not the least such set.

Ex.: Freely Generated

The set $\mathcal{T}$ is "freely generated":

- $\mathsf{o} \neq \iota$

Ex.: Freely Generated

The set $\mathcal{T}$ is "freely generated":

- $\mathsf{o} \neq \iota$
- $\mathsf{o} \neq (\alpha\beta)$

Ex.: Freely Generated

The set $\mathcal{T}$ is "freely generated":

- $o \neq \iota$
- $o \neq (\alpha\beta)$
- $\iota \neq (\alpha\beta)$

Ex.: Freely Generated

The set $\mathcal{T}$ is "freely generated":

- $\mathsf{o} \neq \iota$
- $\mathsf{o} \neq (\alpha\beta)$
- $\iota \neq (\alpha\beta)$
- $(\alpha\beta) = (\gamma\delta) \Rightarrow \alpha = \gamma \wedge \beta = \delta$

Ex.: Types

- $(o\iota) \in \mathcal{T}$

Ex.: Types

- $(o\iota) \in \mathcal{T}$
- $(o(o\iota)) \in \mathcal{T}$

Ex.: Types

- $(\mathsf{o}\iota) \in \mathcal{T}$
- $(\mathsf{o}(\mathsf{o}\iota)) \in \mathcal{T}$
- $(\iota\iota) \in \mathcal{T}$

Ex.: Types

- $(o\iota) \in \mathcal{T}$
- $(o(o\iota)) \in \mathcal{T}$
- $(\iota\iota) \in \mathcal{T}$
- $((o\iota)\iota) \in \mathcal{T}$

Ex.: Types

- $(o\iota) \in \mathcal{T}$
- $(o(o\iota)) \in \mathcal{T}$
- $(\iota\iota) \in \mathcal{T}$
- $((o\iota)\iota) \in \mathcal{T}$

Is $(o\iota\iota)$ also a type?

Ex.: Types

- $(o\iota) \in \mathcal{T}$
- $(o(o\iota)) \in \mathcal{T}$
- $(\iota\iota) \in \mathcal{T}$
- $((o\iota)\iota) \in \mathcal{T}$

Is $(o\iota\iota)$ also a type? – no

Ex.: Types

- $(o\iota) \in \mathcal{T}$
- $(o(o\iota)) \in \mathcal{T}$
- $(\iota\iota) \in \mathcal{T}$
- $((o\iota)\iota) \in \mathcal{T}$

Is $(o\iota\iota)$ also a type? – no

But we can and will consider it shorthand by replacing missing parenthesis, associating to the left: $(o\iota\iota) = ((o\iota)\iota) \neq (o(\iota\iota))$.

Def.: Functions

Let A, B be sets.

Def.: Functions

Let A, B be sets.

$f : B \longrightarrow A$: a function from B to A.

Def.: Functions

Let A, B be sets.

$f : B \longrightarrow A$: a function from B to A.

A^B: set of functions from B to A.

Def.: Functions

Let A, B be sets.

$f : B \longrightarrow A$: a function from B to A.

A^B: set of functions from B to A.

Assume (only for the moment) that A, B are finite.

Def.: Functions

Let A, B be sets.

$f : B \longrightarrow A$: a function from B to A.

A^B: set of functions from B to A.

Assume (only for the moment) that A, B are finite.

Let $|A| = m, |B| = n$. Then $|A^B| = m^n = |A|^{|B|}$.

Def.: Functions

Let A, B be sets.

$f : B \longrightarrow A$: a function from B to A.

A^B: set of functions from B to A.

Assume (only for the moment) that A, B are finite.

Let $|A| = m, |B| = n$. Then $|A^B| = m^n = |A|^{|B|}$.

Example:

Def.: Functions

Let A, B be sets.

$f : B \longrightarrow A$: a function from B to A.

A^B: set of functions from B to A.

Assume (only for the moment) that A, B are finite.

Let $|A| = m, |B| = n$. Then $|A^B| = m^n = |A|^{|B|}$.

Example:

- $f : \{0, 1, 2\} \longrightarrow \{0, 1\}$

Def.: Functions

Let A, B be sets.

$f : B \to A$: a function from B to A.

A^B: set of functions from B to A.

Assume (only for the moment) that A, B are finite.

Let $|A| = m, |B| = n$. Then $|A^B| = m^n = |A|^{|B|}$.

Example:

- $f : \{0, 1, 2\} \to \{0, 1\}$
- $f(0), f(1), f(2) \in \{0, 1\}$

Def.: Functions

Let A, B be sets.
$f : B \longrightarrow A$: a function from B to A.
A^B: set of functions from B to A.

Assume (only for the moment) that A, B are finite.
Let $|A| = m$, $|B| = n$. Then $|A^B| = m^n = |A|^{|B|}$.

Example:

- $f : \{0, 1, 2\} \longrightarrow \{0, 1\}$
- $f(0), f(1), f(2) \in \{0, 1\}$
- $A = \{0, 1\}, B = \{0, 1, 2\}$

Def.: Functions

Let A, B be sets.

$f : B \to A$: a function from B to A.

A^B: set of functions from B to A.

Assume (only for the moment) that A, B are finite.

Let $|A| = m, |B| = n$. Then $|A^B| = m^n = |A|^{|B|}$.

Example:

- $f : \{0, 1, 2\} \to \{0, 1\}$
- $f(0), f(1), f(2) \in \{0, 1\}$
- $A = \{0, 1\}, B = \{0, 1, 2\}$
- $|A^B| = 2 \cdot 2 \cdot 2 = 2^3 = 8$

Ex.: Sets of Functions

Let $F = \{f : B \to A | \forall x, y \in B : x \leq y \Rightarrow f(x) \leq f(y)\} \subseteq A^B$.

Ex.: Sets of Functions

Let $F = \{f : B \to A | \forall x, y \in B : x \leq y \Rightarrow f(x) \leq f(y)\} \subseteq A^B$.

$|F| = ?$

Ex.: Sets of Functions

Let $F = \{f : B \to A | \forall x, y \in B : x \leq y \Rightarrow f(x) \leq f(y)\} \subseteq A^B$.

$|F| = ?$

A^B	$f(0)$	$f(1)$	$f(2)$
$K_0 \in F$	0	0	0
$\in F$	0	0	1
$\notin F$	0	1	0
$\in F$	0	1	1
$g \notin F$	1	0	0
$\notin F$	1	0	1
$\notin F$	1	1	0
$K_1 \in F$	1	1	1

Consider:
$g : x = 0, y = 1, x \leq y$, but $f(x) \geq f(y) \Rightarrow g \notin F$.

Ex.: Sets of Functions

Let $F = \{f : B \to A | \forall x, y \in B : x \leq y \Rightarrow f(x) \leq f(y)\} \subseteq A^B$.

$|F| = ?$

A^B	$f(0)$	$f(1)$	$f(2)$
$K_0 \in F$	0	0	0
$\in F$	0	0	1
$\notin F$	0	1	0
$\in F$	0	1	1
$g \notin F$	1	0	0
$\notin F$	1	0	1
$\notin F$	1	1	0
$K_1 \in F$	1	1	1

Consider:
$g : x = 0, y = 1, x \leq y$, but $f(x) \geq f(y) \Rightarrow g \notin F$.

$|F| = 4$

Ex.: Sets of Labelled Functions

$$C = \{\mathsf{red}, \mathsf{blue}, \mathsf{green}\}$$

Ex.: Sets of Labelled Functions

$$C = \{\mathsf{red}, \mathsf{blue}, \mathsf{green}\}$$

$$F_C = \{\langle c, f\rangle | c \in C, f \in F\}$$

Ex.: Sets of Labelled Functions

$$C = \{\text{red}, \text{blue}, \text{green}\}$$

$$F_C = \{\langle c, f\rangle | c \in C, f \in F\}$$

$$|F_C| = 3 \cdot 4 = 12$$

Def.: Frames

A frame is a family $(\mathrm{D}_\alpha)_{\alpha \in \mathcal{T}}$ of nonempty sets s.t:

Def.: Frames

A frame is a family $(\mathsf{D}_\alpha)_{\alpha \in \mathcal{T}}$ of nonempty sets s.t:

$$\forall \alpha, \beta \in \mathcal{T} : \mathsf{D}_{\alpha\beta} \subseteq \mathsf{D}_\alpha^{\mathsf{D}_\beta}$$

Def.: Frames

A frame is a family $(\mathsf{D}_\alpha)_{\alpha\in\mathcal{T}}$ of nonempty sets s.t:

$$\forall \alpha, \beta \in \mathcal{T} : \mathsf{D}_{\alpha\beta} \subseteq \mathsf{D}_\alpha^{\mathsf{D}_\beta}$$

A Frame is called standard if

Def.: Frames

A frame is a family $(\mathsf{D}_\alpha)_{\alpha\in\mathcal{T}}$ of nonempty sets s.t:

$$\forall\alpha,\beta\in\mathcal{T}:\mathsf{D}_{\alpha\beta}\subseteq\mathsf{D}_\alpha^{\mathsf{D}_\beta}$$

A Frame is called standard if

$$\mathsf{D}_{\alpha\beta}=\mathsf{D}_\alpha^{\mathsf{D}_\beta}\quad\forall\alpha,\beta\in\mathcal{T}$$

Ex.: Frames

$D_o = \{\bot, \top\}$

Ex.: Frames

$D_o = \{\bot, \top\}$

$D_\iota = \{1\}$

Ex.: Frames

$$\mathsf{D}_o = \{\bot, \top\}$$

$$\mathsf{D}_\iota = \{1\}$$

$$\mathsf{D}_{\alpha\beta} = \mathsf{D}_\alpha^{\mathsf{D}_\beta}$$

Ex.: Frames

$\mathsf{D_o} = \{\bot, \top\}$

$\mathsf{D}_\iota = \{1\}$

$\mathsf{D}_{\alpha\beta} = \mathsf{D}_\alpha^{\mathsf{D}_\beta}$

D: the standard frame with $\mathsf{D_o} = \{\bot, \top\}, \mathsf{D_i} = \{1\}$

Ex.: Frames (Contd.)

Consider the set $\mathsf{D}_{\mathsf{o}(\iota\iota)((\mathsf{o}(\iota\mathsf{o})))}$. Is the set empty?

Ex.: Frames (Contd.)

Consider the set $\mathsf{D}_{o(\iota\iota)((o(\iota o)))}$. Is the set empty? — no!

Ex.: Frames (Contd.)

Consider the set $D_{o(\iota\iota)((o(\iota o)))}$. Is the set empty? — no!

Claim: $\forall\alpha \in \mathcal{T} : D_\alpha \neq \emptyset$.

Ex.: Frames (Contd.)

Consider the set $\mathsf{D}_{o(\iota\iota)((o(\iota o)))}$. Is the set empty? — no!

Claim: $\forall \alpha \in \mathcal{T} : \mathsf{D}_\alpha \neq \emptyset$.

Proof: induction on type.

Ex.: Frames (Contd.)

Consider the set $\mathsf{D}_{\mathsf{o}(\iota\iota)((\mathsf{o}(\iota\mathsf{o})))}$. Is the set empty? — no!

Claim: $\forall\alpha \in \mathcal{T} : \mathsf{D}_\alpha \neq \emptyset$.
Proof: induction on type.

- Base: $\mathsf{D}_\mathsf{o} = \{\bot, \top\} \neq \emptyset, \mathsf{D}_\mathsf{i} = \{1\} \neq \emptyset$.

Ex.: Frames (Contd.)

Consider the set $\mathsf{D}_{\mathsf{o}(\iota\iota)((\mathsf{o}(\iota\mathsf{o})))}$. Is the set empty? — no!

Claim: $\forall\alpha \in \mathcal{T} : \mathsf{D}_\alpha \neq \emptyset$.
Proof: induction on type.

- Base: $\mathsf{D}_\mathsf{o} = \{\bot, \top\} \neq \emptyset, \mathsf{D}_\mathsf{i} = \{1\} \neq \emptyset$.
- Step: Assume $\mathsf{D}_\alpha \neq \emptyset \wedge \mathsf{D}_\beta \neq \emptyset$. Want to show: $\mathsf{D}_{\alpha\beta} \neq \emptyset$.

Ex.: Frames (Contd.)

Consider the set $D_{o(\iota\iota)((o(\iota o)))}$. Is the set empty? — no!

Claim: $\forall \alpha \in \mathcal{T} : D_\alpha \neq \emptyset$.

Proof: induction on type.

- Base: $D_o = \{\bot, \top\} \neq \emptyset, D_i = \{1\} \neq \emptyset$.
- Step: Assume $D_\alpha \neq \emptyset \wedge D_\beta \neq \emptyset$. Want to show: $D_{\alpha\beta} \neq \emptyset$.
  Since $D_\alpha \neq \emptyset \Rightarrow \exists a \in D_\alpha$,

Ex.: Frames (Contd.)

Consider the set $D_{o(\iota\iota)((o(\iota o)))}$. Is the set empty? — no!

Claim: $\forall \alpha \in \mathcal{T} : D_\alpha \neq \emptyset$.
Proof: induction on type.

- Base: $D_o = \{\bot, \top\} \neq \emptyset, D_i = \{1\} \neq \emptyset$.
- Step: Assume $D_\alpha \neq \emptyset \wedge D_\beta \neq \emptyset$. Want to show: $D_{\alpha\beta} \neq \emptyset$.
  Since $D_\alpha \neq \emptyset \Rightarrow \exists a \in D_\alpha$, hence $K_a \in D_{\alpha\beta}$.

Ex.: Frames (Contd.)

Consider the set $D_{o(\iota\iota)((o(\iota o)))}$. Is the set empty? — no!

Claim: $\forall \alpha \in \mathcal{T} : D_\alpha \neq \emptyset$.
Proof: induction on type.

- Base: $D_o = \{\bot, \top\} \neq \emptyset, D_i = \{1\} \neq \emptyset$.
- Step: Assume $D_\alpha \neq \emptyset \wedge D_\beta \neq \emptyset$. Want to show: $D_{\alpha\beta} \neq \emptyset$.
  Since $D_\alpha \neq \emptyset \Rightarrow \exists a \in D_\alpha$, hence $K_a \in D_{\alpha\beta}$.

  (Here K_a is the constant function which always returns a. We will often use this notation for constant functions.)

Def.: Typed Applicative Structure

A (typed) applicative structure is a tupel

Def.: Typed Applicative Structure

A (typed) applicative structure is a tupel

$$\langle \mathcal{D}, @ \rangle$$

where

Def.: Typed Applicative Structure

A (typed) applicative structure is a tupel

$$\langle \mathrm{D}, @ \rangle$$

where

- $\mathrm{D} := (\mathrm{D}_\alpha)_{\alpha \in \mathcal{T}}$ is a family of nonempty sets

Def.: Typed Applicative Structure

A (typed) applicative structure is a tupel

$$\langle \mathsf{D}, @ \rangle$$

where

- $\mathsf{D} := (\mathsf{D}_\alpha)_{\alpha \in \mathcal{T}}$ is a family of nonempty sets
- $@ := (@^{\alpha\beta} : \mathsf{D}_{\alpha\beta} \times \mathsf{D}_\beta \longrightarrow \mathsf{D}_\alpha)_{\alpha,\beta \in \mathcal{T}}$

Def.: Typed Applicative Structure

A (typed) applicative structure is a tupel

$$\langle \mathsf{D}, @ \rangle$$

where

- $\mathsf{D} := (\mathsf{D}_\alpha)_{\alpha \in \mathcal{T}}$ is a family of nonempty sets
- $@ := (@^{\alpha\beta} : \mathsf{D}_{\alpha\beta} \times \mathsf{D}_\beta \longrightarrow \mathsf{D}_\alpha)_{\alpha,\beta \in \mathcal{T}}$

Usually we write $\mathsf{f}@\mathsf{b}$ for $@^{\alpha\beta}(\mathsf{f}, \mathsf{b})$ when $\mathsf{f} \in \mathsf{D}_{\alpha\beta} \wedge \mathsf{b} \in \mathsf{D}_\beta$

Rem.: Currying

The application operator $@$ in an applicative structure is an abstract version of function application.

Rem.: Currying

The application operator $@$ in an applicative structure is an abstract version of function application. It is no restriction to exclusively use a binary application operator, which corresponds to unary function application,

Rem.: Currying

The application operator $@$ in an applicative structure is an abstract version of function application. It is no restriction to exclusively use a binary application operator, which corresponds to unary function application, since we can define higher-arity application operators from the binary one by setting $\mathsf{f}@(\mathsf{a}^1, \ldots, \mathsf{a}^n) := (\ldots(\mathsf{f}@\mathsf{a}^1)\ldots@\mathsf{a}^n)$ ("Currying").

Interesting Properties

Let $\mathcal{D}$ be a frame.

Interesting Properties

Let D be a frame.

$$\forall f, g \in D_{\alpha\beta} \quad (\forall b \in D_\beta : f(b) = g(b)) \Rightarrow f = g.$$

Interesting Properties

Let D be a frame.

$$\forall f, g \in D_{\alpha\beta} \quad (\forall b \in D_\beta : f(b) = g(b)) \Rightarrow f = g.$$

Let $\langle D, @\rangle$ be an applicative structure. Consider the property:

Interesting Properties

Let D be a frame.

$$\forall f, g \in D_{\alpha\beta} \quad (\forall b \in D_\beta : f(b) = g(b)) \Rightarrow f = g.$$

Let $\langle D, @\rangle$ be an applicative structure. Consider the property:

$$\forall f, g \in D_{\alpha\beta} \quad (\forall b \in D_\beta : f@b = g@b) \Rightarrow f = g.$$

Def.: Functional Applicative Structures

Given an applicative structure $\langle \mathcal{D}, @ \rangle$.

Def.: Functional Applicative Structures

Given an applicative structure $\langle \mathrm{D}, @ \rangle$. We say that $\langle \mathrm{D}, @ \rangle$ is functional if

Def.: Functional Applicative Structures

Given an applicative structure $\langle \mathsf{D}, @ \rangle$. We say that $\langle \mathsf{D}, @ \rangle$ is functional if

$$\forall \alpha, \beta \in \mathcal{T} : \forall \mathsf{f}, \mathsf{g} \in \mathsf{D}_{\alpha\beta}(\forall \mathsf{b} \in \mathsf{D}_\beta : \mathsf{f}@\mathsf{b} = \mathsf{g}@\mathsf{b}) \Rightarrow \mathsf{f} = \mathsf{g}$$

Def.: Full Applicative Structures

Given an applicative structure $\langle \mathcal{D}, @ \rangle$.

Def.: Full Applicative Structures

Given an applicative structure $\langle \mathrm{D}, @ \rangle$. We say that $\langle \mathrm{D}, @ \rangle$ is full if

Def.: Full Applicative Structures

Given an applicative structure $\langle \mathsf{D}, @ \rangle$. We say that $\langle \mathsf{D}, @ \rangle$ is full if

$$\forall \alpha, \beta \quad \forall \mathsf{h} : \mathsf{D}_\beta \longrightarrow \mathsf{D}_\alpha \quad \exists \mathsf{f} \in \mathsf{D}_{\alpha\beta} \forall \mathsf{b} \in \mathsf{D}_\beta : \mathsf{f} @ \mathsf{b} = \mathsf{h}(\mathsf{b})$$

Def.: Standard Applicative Structures

An applicative structure $\mathcal{A} := \langle \mathrm{D}, @\rangle$ is called standard if

Def.: Standard Applicative Structures

An applicative structure $\mathcal{A} := \langle \mathrm{D}, @ \rangle$ is called standard if it is a frame structure (i.e. $@$ is function application) where $\mathcal{D}$ is standard.

Def.: Standard Applicative Structures

An applicative structure $\mathcal{A} := \langle \mathrm{D}, @ \rangle$ is called standard if it is a frame structure (i.e. $@$ is function application) where $\mathcal{D}$ is standard.

Note that the definitions of functional, full, and standard impose restrictions on the domains for function types only.

Rem.: Frames and Applicative Structures

It is easy to show that every frame is functional.

Rem.: Frames and Applicative Structures

It is easy to show that every frame is functional.

Furthermore, an applicative structure is standard iff it is a full frame.

Example: Full Functional Appl. Structure

Let $D_\alpha = \{1\} \quad \forall\alpha$

Example: Full Functional Appl. Structure

Let $D_\alpha = \{1\} \quad \forall\alpha$

Let $f@b = 1 \quad \forall f \in D_{\alpha\beta} \quad \forall b \in D_\beta$

Example: Full Functional Appl. Structure

Let $\mathsf{D}_\alpha = \{1\} \quad \forall \alpha$

Let $\mathsf{f}@\mathsf{b} = 1 \quad \forall \mathsf{f} \in \mathsf{D}_{\alpha\beta} \quad \forall \mathsf{b} \in \mathsf{D}_\beta$

$\langle \mathsf{D}, @ \rangle$ is a full functional applicative structure, but it is not a frame.

Example: Full Functional Appl. Structure

Let $D_\alpha = \{1\} \quad \forall\alpha$

Let $f@b = 1 \quad \forall f \in D_{\alpha\beta} \quad \forall b \in D_\beta$

$\langle D, @\rangle$ is a full functional applicative structure, but it is not a frame.

$1 \in D_{oo}$ but $1 \notin D_o^{D_o} \Rightarrow D_{oo} \not\subseteq D_o^{D_o}$

Def.: Homomorphic Appl. Structures

Let $\langle \mathrm{D}^1, @^1 \rangle$ and $\langle \mathrm{D}^2, @^2 \rangle$ are applicative structures.

Def.: Homomorphic Appl. Structures

Let $\langle \mathrm{D}^1, @^1 \rangle$ and $\langle \mathrm{D}^2, @^2 \rangle$ are applicative structures. We say that κ is a homomorphism from $\langle \mathrm{D}^1, @^1 \rangle$ to $\langle \mathrm{D}^2, @^2 \rangle$ if

Def.: Homomorphic Appl. Structures

Let $\langle \mathsf{D}^1, @^1 \rangle$ and $\langle \mathsf{D}^2, @^2 \rangle$ are applicative structures. We say that κ is a homomorphism from $\langle \mathsf{D}^1, @^1 \rangle$ to $\langle \mathsf{D}^2, @^2 \rangle$ if

- $\kappa_\alpha : \mathsf{D}^1_\alpha \longrightarrow \mathsf{D}^2_\alpha \quad \forall \alpha \in \mathcal{T}$

Def.: Homomorphic Appl. Structures

Let $\langle \mathsf{D}^1, @^1 \rangle$ and $\langle \mathsf{D}^2, @^2 \rangle$ are applicative structures. We say that κ is a homomorphism from $\langle \mathsf{D}^1, @^1 \rangle$ to $\langle \mathsf{D}^2, @^2 \rangle$ if

- $\kappa_\alpha : \mathsf{D}^1_\alpha \longrightarrow \mathsf{D}^2_\alpha \quad \forall \alpha \in \mathcal{T}$
- $\forall \alpha, \beta \in \mathcal{T}, \quad \forall \mathsf{f} \in \mathsf{D}^1_{\alpha\beta}, \quad \forall \mathsf{b} \in \mathsf{D}^1_\beta$:

$$\kappa(\mathsf{f})@^2\kappa(\mathsf{b}) = \kappa(\mathsf{f}@^1\mathsf{b})$$

Def.: Isomorphic Appl. Structures

We say that $\langle \mathcal{D}^1, @^1 \rangle$ and $\langle \mathcal{D}^2, @^2 \rangle$ are isomorphic if $\exists i, j$ s.t:

Def.: Isomorphic Appl. Structures

We say that $\langle \mathrm{D}^1, @^1 \rangle$ and $\langle \mathrm{D}^2, @^2 \rangle$ are isomorphic if $\exists \mathrm{i}, \mathrm{j}$ s.t:

- i is a homomorphism from $\langle \mathrm{D}^1, @^1 \rangle$ to $\langle \mathrm{D}^2, @^2 \rangle$

Def.: Isomorphic Appl. Structures

We say that $\langle \mathrm{D}^1, @^1 \rangle$ and $\langle \mathrm{D}^2, @^2 \rangle$ are isomorphic if $\exists \mathrm{i}, \mathrm{j}$ s.t:

- i is a homomorphism from $\langle \mathrm{D}^1, @^1 \rangle$ to $\langle \mathrm{D}^2, @^2 \rangle$
- j is a homomorphism from $\langle \mathrm{D}^2, @^2 \rangle$ to $\langle \mathrm{D}^1, @^1 \rangle$

Def.: Isomorphic Appl. Structures

We say that $\langle D^1, @^1\rangle$ and $\langle D^2, @^2\rangle$ are isomorphic if $\exists i, j$ s.t:

- i is a homomorphism from $\langle D^1, @^1\rangle$ to $\langle D^2, @^2\rangle$
- j is a homomorphism from $\langle D^2, @^2\rangle$ to $\langle D^1, @^1\rangle$
- i and j are inverses (i.e $i(j(a^2)) = a^2$ and $j(i(a^1)) = a^1$).

Simply Typed λ-Calculus

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

- $\mathcal{V}$ — countably infinite set of variables

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

- $\mathcal{V}$ — countably infinite set of variables
- $\mathcal{C}$ — possibly empty set of constants

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

- $\mathcal{V}$ — countably infinite set of variables
- $\mathcal{C}$ — possibly empty set of constants

We define the set $\Lambda = \mathit{wff}_{\Sigma}(\Sigma)$ to be the smallest set s.t:

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

- $\mathcal{V}$ — countably infinite set of variables
- $\mathcal{C}$ — possibly empty set of constants

We define the set $\Lambda = wff_{\Sigma}(\Sigma)$ to be the smallest set s.t:

- $x \in \mathcal{V}$ then $x \in \Lambda$

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

- $\mathcal{V}$ — countably infinite set of variables
- $\mathcal{C}$ — possibly empty set of constants

We define the set $\Lambda = \mathit{wff}_\Sigma(\Sigma)$ to be the smallest set s.t:

- $\mathsf{x} \in \mathcal{V}$ then $\mathsf{x} \in \Lambda$
- $\mathsf{c} \in \mathcal{C}$ then $\mathsf{c} \in \Lambda$

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

- $\mathcal{V}$ — countably infinite set of variables
- $\mathcal{C}$ — possibly empty set of constants

We define the set $\Lambda = \mathit{wff}_\Sigma(\Sigma)$ to be the smallest set s.t:

- $\mathsf{x} \in \mathcal{V}$ then $\mathsf{x} \in \Lambda$
- $\mathsf{c} \in \mathcal{C}$ then $\mathsf{c} \in \Lambda$
- $\mathsf{A} \in \Lambda$, $\mathsf{B} \in \Lambda$ then $(\mathsf{A}\,\mathsf{B}) \in \Lambda$

Def.: Untyped λ-Calculus

Let $\Sigma = (\mathcal{V}, \mathcal{C})$ be a signature where

- $\mathcal{V}$ — countably infinite set of variables
- $\mathcal{C}$ — possibly empty set of constants

We define the set $\Lambda = \mathit{wff}_\Sigma(\Sigma)$ to be the smallest set s.t:

- $\mathsf{x} \in \mathcal{V}$ then $\mathsf{x} \in \Lambda$
- $\mathsf{c} \in \mathcal{C}$ then $\mathsf{c} \in \Lambda$
- $\mathsf{A} \in \Lambda$, $\mathsf{B} \in \Lambda$ then $(\mathsf{A}\,\mathsf{B}) \in \Lambda$
- $\mathsf{x} \in \mathcal{V}$, $\mathsf{A} \in \Lambda$ then $(\lambda \mathsf{x}.\mathsf{A}) \in \Lambda$

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^\alpha, \mathcal{C}^\alpha)$ be a signature where

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^\alpha, \mathcal{C}^\alpha)$ be a signature where

- $\mathcal{V}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$ — countably infinite sets of variables

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^{\alpha}, \mathcal{C}^{\alpha})$ be a signature where

- $\mathcal{V}^{\alpha} = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_{\alpha}$ — countably infinite sets of variables
- $\mathcal{C}^{\alpha} = \bigcup_{\alpha \in \mathcal{T}} \mathcal{C}_{\alpha}$ — possibly empty sets of constants

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^\alpha, \mathcal{C}^\alpha)$ be a signature where

- $\mathcal{V}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$ — countably infinite sets of variables
- $\mathcal{C}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{C}_\alpha$ — possibly empty sets of constants

We define the set $\Lambda^\alpha = \mathit{wff}_\Sigma(\Sigma)_\alpha = \bigcup_{\alpha \in \mathcal{T}} \Lambda_\alpha$ to be the smallest set s.t:

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^\alpha, \mathcal{C}^\alpha)$ be a signature where

- $\mathcal{V}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$ — countably infinite sets of variables
- $\mathcal{C}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{C}_\alpha$ — possibly empty sets of constants

We define the set $\Lambda^\alpha = \mathit{wff}_\Sigma(\Sigma)_\alpha = \bigcup_{\alpha \in \mathcal{T}} \Lambda_\alpha$ to be the smallest set s.t:

- $\mathsf{x}_\alpha \in \mathcal{V}_\alpha$ then $\mathsf{x}_\alpha \in \Lambda_\alpha$

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^\alpha, \mathcal{C}^\alpha)$ be a signature where

- $\mathcal{V}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$ — countably infinite sets of variables
- $\mathcal{C}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{C}_\alpha$ — possibly empty sets of constants

We define the set $\Lambda^\alpha = \mathit{wff}_\Sigma(\Sigma)_\alpha = \bigcup_{\alpha \in \mathcal{T}} \Lambda_\alpha$ to be the smallest set s.t:

- $\mathsf{x}_\alpha \in \mathcal{V}_\alpha$ then $\mathsf{x}_\alpha \in \Lambda_\alpha$
- $\mathsf{c}_\alpha \in \mathcal{C}_\alpha$ then $\mathsf{c}_\alpha \in \Lambda_\alpha$

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^\alpha, \mathcal{C}^\alpha)$ be a signature where

- $\mathcal{V}^\alpha = \bigcup\limits_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$ — countably infinite sets of variables
- $\mathcal{C}^\alpha = \bigcup\limits_{\alpha \in \mathcal{T}} \mathcal{C}_\alpha$ — possibly empty sets of constants

We define the set $\Lambda^\alpha = \mathit{wff}_\Sigma(\Sigma)_\alpha = \bigcup\limits_{\alpha \in \mathcal{T}} \Lambda_\alpha$ to be the smallest set s.t:

- $x_\alpha \in \mathcal{V}_\alpha$ then $x_\alpha \in \Lambda_\alpha$
- $c_\alpha \in \mathcal{C}_\alpha$ then $c_\alpha \in \Lambda_\alpha$
- $A_{\alpha\beta} \in \Lambda_{\alpha\beta}$, $B_\beta \in \Lambda_\beta$ then $(A\,B) \in \Lambda_\alpha$

Simply Typed λ-Calculus

Let $\Sigma = (\mathcal{V}^\alpha, \mathcal{C}^\alpha)$ be a signature where

- $\mathcal{V}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$ — countably infinite sets of variables
- $\mathcal{C}^\alpha = \bigcup_{\alpha \in \mathcal{T}} \mathcal{C}_\alpha$ — possibly empty sets of constants

We define the set $\Lambda^\alpha = \mathit{wff}_\Sigma(\Sigma)_\alpha = \bigcup_{\alpha \in \mathcal{T}} \Lambda_\alpha$ to be the smallest set s.t:

- $\mathsf{x}_\alpha \in \mathcal{V}_\alpha$ then $\mathsf{x}_\alpha \in \Lambda_\alpha$
- $\mathsf{c}_\alpha \in \mathcal{C}_\alpha$ then $\mathsf{c}_\alpha \in \Lambda_\alpha$
- $\mathsf{A}_{\alpha\beta} \in \Lambda_{\alpha\beta}$, $\mathsf{B}_\beta \in \Lambda_\beta$ then $(\mathsf{A}\,\mathsf{B}) \in \Lambda_\alpha$
- $\mathsf{x}_\alpha \in \mathcal{V}_\alpha$, $\mathsf{A}_\beta \in \Lambda_\beta$ then $(\lambda \mathsf{x}_\alpha . \mathsf{A}_\beta)_{\beta\alpha} \in \Lambda_{\beta\alpha}$

Notational Conventions

- brackets may be avoided: $A\,B\,C \rightsquigarrow ((A\,B)\,C)$

Notational Conventions

- brackets may be avoided: $A\,B\,C \rightsquigarrow ((A\,B)\,C)$
- $\lambda x_\iota.A_{o\iota}\,B_\iota\,C_\iota$ — dots as far to the right as is consistent: $((\lambda x_\iota.A_{o\iota}B_\iota)C_\iota)$

Notational Conventions

- brackets may be avoided: $\mathsf{A\,B\,C} \rightsquigarrow ((\mathsf{A\,B})\,\mathsf{C})$
- $\lambda \mathsf{x}_\iota.\mathsf{A}_{\mathsf{o}\iota}\,\mathsf{B}_\iota\,\mathsf{C}_\iota$ — dots as far to the right as is consistent: $((\lambda \mathsf{x}_\iota.\mathsf{A}_{\mathsf{o}\iota}\mathsf{B}_\iota)\mathsf{C}_\iota)$
- $\lambda \mathsf{x}, \mathsf{y}.\mathsf{A} \rightsquigarrow (\lambda \mathsf{x}.(\lambda \mathsf{y}.\mathsf{A}))$

Notational Conventions

- brackets may be avoided: $A\,B\,C \rightsquigarrow ((A\,B)\,C)$
- $\lambda x_\iota . A_{o\iota}\, B_\iota\, C_\iota$ — dots as far to the right as is consistent: $((\lambda x_\iota . A_{o\iota} B_\iota) C_\iota)$
- $\lambda x, y.A \rightsquigarrow (\lambda x.(\lambda y.A))$
- $\lambda \overline{x}^n.A \rightsquigarrow (\lambda x_1.(\ldots(\lambda x_n.A)\ldots))$

Notational Conventions

- brackets may be avoided: $A\,B\,C \rightsquigarrow ((A\,B)\,C)$
- $\lambda x_\iota.A_{o\iota}\,B_\iota\,C_\iota$ — dots as far to the right as is consistent: $((\lambda x_\iota.A_{o\iota}B_\iota)C_\iota)$
- $\lambda x, y.A \rightsquigarrow (\lambda x.(\lambda y.A))$
- $\lambda \overline{x}^n.A \rightsquigarrow (\lambda x_1.(\ldots(\lambda x_n.A)\ldots))$
- $\lambda \overline{x}.A$ — n is not important

Notational Conventions

- brackets may be avoided: $A\,B\,C \rightsquigarrow ((A\,B)\,C)$
- $\lambda x_\iota . A_{o\iota}\, B_\iota\, C_\iota$ — dots as far to the right as is consistent: $((\lambda x_\iota . A_{o\iota} B_\iota) C_\iota)$
- $\lambda x, y.A \rightsquigarrow (\lambda x.(\lambda y.A))$
- $\lambda \overline{x}^n.A \rightsquigarrow (\lambda x_1.(\ldots(\lambda x_n.A)\ldots))$
- $\lambda \overline{x}.A$ — n is not important
- $(f\,\overline{A}^n) \rightsquigarrow (\ldots((f\,A^1)\,A^2)\ldots A^n)$

Def.: Positions in λ-Terms

Consider the following term:

$$((\lambda x.x)((\lambda y.y)(\lambda z.z)))$$

Def.: Positions in λ-Terms

Consider the following term:

$$((\lambda x.x)((\lambda y.y)(\lambda z.z)))$$

The position $[212]$ points to the red y in

$$((\lambda x.x)((\lambda y.y)(\lambda z.z)))$$

Def.: Positions in λ-Terms

Consider the following term:

$$((\lambda x.x)((\lambda y.y)(\lambda z.z)))$$

The position $[212]$ points to the red y in

$$((\lambda x.x)((\lambda y.y)(\lambda z.z)))$$

. . . Graphics on Blackboard . . .

Def.: Position (Contd.)

The expression

$$A_p$$

refers to the subterm of A at position p.

Def.: Position (Contd.)

The expression

$$A_p$$

refers to the subterm of A at position p.

Example: Consider $T := ((\lambda x.x)((\lambda y.y)(\lambda z.z)))$

Def.: Position (Contd.)

The expression

$$A_p$$

refers to the subterm of A at position p.

Example: Consider $T := ((\lambda x.x)((\lambda y.y)(\lambda z.z)))$

$$T_{[212]} = y$$

Def.: Replacement at Position

Replacement of A_p in A by a term B is denoted as

$$A[B]_p$$

Def.: Replacement at Position

Replacement of A_p in A by a term B is denoted as

$$A[B]_p$$

Example:

$$T[(f\,x)]_{[212]} = ((\lambda x.x)((\lambda y.(fx))(\lambda z.z)))$$

Def.: Scope of λ-Term

$(\lambda x.A)$: We say that A is in the scope of λ-binder that binds x.

Def.: Free and Bound Variables

An occurrence of a variable x in a term A is called bound if it is in the scope of a λ-binder that binds x.

Def.: Free and Bound Variables

An occurrence of a variable x in a term A is called bound if it is in the scope of a λ-binder that binds x.

Otherwise it is called free.

Def.: Free and Bound Variables

An occurrence of a variable x in a term A is called bound if it is in the scope of a λ-binder that binds x.

Otherwise it is called free.

We denote the set of all free variables in a λ-term as $FV(A)$.

Syntax: Simply Typed λ-Calculus (Contd.)

Def.: Substitution

Substitution is a map

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad (\text{untyped})$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad (\text{typed})$$

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad \text{(typed)}$$

and is defined as follows:

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad \text{(typed)}$$

and is defined as follows:

1. $[N_\alpha/x_\alpha]x_\alpha = N_\alpha$

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad \text{(typed)}$$

and is defined as follows:

1. $[N_\alpha/x_\alpha]x_\alpha = N_\alpha$
2. $[N_\alpha/x_\alpha]a_\beta = a_\beta$ if $a_\beta \neq x_\alpha \wedge a_\beta \in \mathcal{V}_\beta \cup \mathcal{C}_\beta$

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad \text{(typed)}$$

and is defined as follows:

1. $[N_\alpha/x_\alpha]x_\alpha = N_\alpha$
2. $[N_\alpha/x_\alpha]a_\beta = a_\beta$ if $a_\beta \neq x_\alpha \wedge a_\beta \in \mathcal{V}_\beta \cup \mathcal{C}_\beta$
3. $[N_\alpha/x_\alpha](A_{\alpha\alpha}B_\beta) = ([N_\alpha/x_\alpha]A)([N_\alpha/x_\alpha]B)$

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad \text{(typed)}$$

and is defined as follows:

1. $[N_\alpha/x_\alpha]x_\alpha = N_\alpha$
2. $[N_\alpha/x_\alpha]a_\beta = a_\beta$ if $a_\beta \neq x_\alpha \wedge a_\beta \in \mathcal{V}_\beta \cup \mathcal{C}_\beta$
3. $[N_\alpha/x_\alpha](A_{\alpha\alpha}B_\beta) = ([N_\alpha/x_\alpha]A)([N_\alpha/x_\alpha]B)$
4. $[N_\alpha/x_\alpha](\lambda x_\alpha.A_\gamma) = (\lambda x_\alpha A_\gamma)$

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad \text{(typed)}$$

and is defined as follows:

1. $[N_\alpha/x_\alpha]x_\alpha = N_\alpha$
2. $[N_\alpha/x_\alpha]a_\beta = a_\beta$ if $a_\beta \neq x_\alpha \wedge a_\beta \in \mathcal{V}_\beta \cup \mathcal{C}_\beta$
3. $[N_\alpha/x_\alpha](A_{\alpha\alpha}B_\beta) = ([N_\alpha/x_\alpha]A)([N_\alpha/x_\alpha]B)$
4. $[N_\alpha/x_\alpha](\lambda x_\alpha.A_\gamma) = (\lambda x_\alpha A_\gamma)$
5. $[N_\alpha/x_\alpha](\lambda y_\beta.A_\gamma) = (\lambda y_\beta.[N_\alpha/x_\alpha]A_\gamma)$ if
   $x_\alpha \neq y_\beta \wedge (y_\beta \notin FV(N_\alpha) \vee x_\alpha \notin FV(A_\gamma))$

Def.: Substitution

Substitution is a map

$$[A/x] : \Lambda \longrightarrow \Lambda \quad \text{(untyped)}$$
$$[A_\alpha/x_\alpha] : \Lambda_\alpha \longrightarrow \Lambda_\alpha \quad \text{(typed)}$$

and is defined as follows:

1. $[N_\alpha/x_\alpha]x_\alpha = N_\alpha$
2. $[N_\alpha/x_\alpha]a_\beta = a_\beta$ if $a_\beta \neq x_\alpha \wedge a_\beta \in \mathcal{V}_\beta \cup \mathcal{C}_\beta$
3. $[N_\alpha/x_\alpha](A_{\alpha\alpha}B_\beta) = ([N_\alpha/x_\alpha]A)([N_\alpha/x_\alpha]B)$
4. $[N_\alpha/x_\alpha](\lambda x_\alpha.A_\gamma) = (\lambda x_\alpha A_\gamma)$
5. $[N_\alpha/x_\alpha](\lambda y_\beta.A_\gamma) = (\lambda y_\beta.[N_\alpha/x_\alpha]A_\gamma)$ if $x_\alpha \neq y_\beta \wedge (y_\beta \notin FV(N_\alpha) \vee x_\alpha \notin FV(A_\gamma))$
6. $[N_\alpha/x_\alpha](\lambda y_\alpha.A_\gamma) = (\lambda z_\beta.[N_\alpha/x_\alpha][z_\beta/y_\beta]A_\gamma)$ if $x_\alpha \neq y_\beta \wedge$ $(y_\beta \in FV(N_\alpha) \wedge x_\alpha \in FV(A_\gamma))$ and z is a 'fresh' variable.

Ex.: Substitution

- $[y/x](\lambda y.x)$ — the occurrence of x is free
  $\neq (\lambda y.y)$ — if we replace x with y, the variable y becomes bound.

Ex.: Substitution

- $[y/x](\lambda y.x)$ — the occurrence of x is free
  $\neq (\lambda y.y)$ — if we replace x with y, the variable y becomes bound.
- $[y/x](\lambda y.x)$ — the occurrence of x is free
  $= (\lambda z[y/x][z/y]x)$ — we need a fresh variable
  $= (\lambda z.y)$ — the occurrence of y is free

Ex.: Substitution

- $[y/x](\lambda y.x)$ — the occurrence of x is free
  $\neq (\lambda y.y)$ — if we replace x with y, the variable y becomes bound.
- $[y/x](\lambda y.x)$ — the occurrence of x is free
  $= (\lambda z[y/x][z/y]x)$ — we need a fresh variable
  $= (\lambda z.y)$ — the occurrence of y is free
- Further Examples on Blackboard

Ex.: Substitution

- $[y/x](\lambda y.x)$ — the occurrence of x is free
  $\neq (\lambda y.y)$ — if we replace x with y, the variable y becomes bound.
- $[y/x](\lambda y.x)$ — the occurrence of x is free
  $= (\lambda z[y/x][z/y]x)$ — we need a fresh variable
  $= (\lambda z.y)$ — the occurrence of y is free
- Further Examples on Blackboard
- Claim: $[N/x]A = A$ if $x \notin FV(A)$
  Proof: Induction on A

Def.: α-Conversion

$$[\lambda x.\, M] \rightarrow_{\alpha} [\lambda y.\, [y/x]M]$$

where $y \notin FV(M)$

Def.: α-Conversion

$$[\lambda x.\, M] \rightarrow_{\alpha} [\lambda y.\, [y/x]M]$$

where $y \notin FV(M)$

$$A =^{\alpha} B$$

if A can be converted to B by renaming the bound variables. We read $A =_{\alpha} B$ as A is α-equal to B.

Def.: α-Conversion

$$[\lambda x.\, M] \rightarrow_\alpha [\lambda y.\, [y/x]M]$$

where $y \notin FV(M)$

$$A =^\alpha B$$

if A can be converted to B by renaming the bound variables. We read $A =_\alpha B$ as A is α-equal to B.

From now on $(\lambda y.\, y) = (\lambda z.\, z)$, that is, we will say that two terms are simply equal, if they are α-equal. Two terms are equal means that two terms are α-convertable.

Def.: β-Conversion

A β-redex is a term $((\lambda x.\, A)B)$. The β-reduct of this redex is $[B/x]A$.

Def.: β-Conversion

A β-redex is a term $((\lambda x.\, A)B)$. The β-reduct of this redex is $[B/x]A$.

We say $M \to_\beta N$, ie. β-reduces in 1 step, if

$$
\begin{aligned}
M &= P[(\lambda x.\, A)B]_p \\
N &= P[[B/x]A]_p
\end{aligned}
$$

Def.: β-Conversion

A β-redex is a term $((\lambda x.\, A)B)$. The β-reduct of this redex is $[B/x]A$.

We say $M \rightarrow_\beta N$, ie. β-reduces in 1 step, if

$$\begin{aligned} M &= P[(\lambda x.\, A)B]_p \\ N &= P[[B/x]A]_p \end{aligned}$$

We say $M \twoheadrightarrow_\beta N$, ie. β-reduces in several steps, if $\exists M^1, \ldots, M^n$ for $n \geq 1$ such that $M = M^1$ and $N = M^n$ and $M^i \rightarrow_\beta M^{i+1}$.

Def.: β-Normal Form

A term is called β-normal if it contains no β-redexes.

Def.: β-Normal Form

A term is called β-normal if it contains no β-redexes.

Any term that does not contain λ-abstractions is β-normal.

Def.: β-Normal Form

A term is called β-normal if it contains no β-redexes.

Any term that does not contain λ-abstractions is β-normal.

A term is called β-head normal if the head term of its outermost application can not be further reduced.

Def.: β-Normal Form

A term is called β-normal if it contains no β-redexes.

Any term that does not contain λ-abstractions is β-normal.

A term is called β-head normal if the head term of its outermost application can not be further reduced.

Any term that does not contain λ-abstractions is β-head normal.

Thm.: Church-Rosser Property for $\twoheadrightarrow_\beta$

$$
\begin{tikzcd}
 & \mathsf{T}_\alpha \arrow[dl, two heads, "\beta"'] \arrow[dr, two heads, "\beta"] & \\
\mathsf{L}_\alpha \arrow[dr, two heads, "\beta"'] & & \mathsf{R}_\alpha \arrow[dl, two heads, "\beta"] \\
 & \mathsf{B}_\alpha &
\end{tikzcd}
$$

Thm.: Church-Rosser Property for $\twoheadrightarrow_\beta$

$$
\begin{tikzcd}
 & T_\alpha \arrow[dl, two heads, "\beta"'] \arrow[dr, two heads, "\beta"] & \\
L_\alpha \arrow[dr, two heads, "\beta"'] & & R_\alpha \arrow[dl, two heads, "\beta"] \\
 & B_\alpha &
\end{tikzcd}
$$

If T_α β-reduces in multiple steps with one strategy to L_α and with another strategy to R_α then there exists a term B_α such that L_α and R_α β-reduce in multiple steps to B_α.

Thm.: Church-Rosser Property for $\twoheadrightarrow_\beta$

$$\begin{tikzcd}
 & \mathsf{T}_\alpha \arrow[dl, two heads, "\beta"'] \arrow[dr, two heads, "\beta"] & \\
\mathsf{L}_\alpha \arrow[dr, two heads, "\beta"'] & & \mathsf{R}_\alpha \arrow[dl, two heads, "\beta"] \\
 & \mathsf{B}_\alpha &
\end{tikzcd}$$

If T_α β-reduces in multiple steps with one strategy to L_α and with another strategy to R_α then there exists a term B_α such that L_α and R_α β-reduce in multiple steps to B_α.

Note that B_α is not necessarily in normal form.

Thm.: Church-Rosser Property for $\twoheadrightarrow_\beta$

$$\begin{tikzcd}
 & \mathsf{T}_\alpha \arrow[dl, two heads, "\beta"'] \arrow[dr, two heads, "\beta"] & \\
\mathsf{L}_\alpha \arrow[dr, two heads, "\beta"'] & & \mathsf{R}_\alpha \arrow[dl, two heads, "\beta"] \\
 & \mathsf{B}_\alpha &
\end{tikzcd}$$

If T_α β-reduces in multiple steps with one strategy to L_α and with another strategy to R_α then there exists a term B_α such that L_α and R_α β-reduce in multiple steps to B_α.

Note that B_α is not necessarily in normal form.

The Church-Rosser Property for $\twoheadrightarrow_\beta$ holds for Λ and Λ^α.

Ex.: Church-Rosser Property for $\twoheadrightarrow_\beta$

$$\begin{tikzcd}[column sep=small]
 & ((\lambda f.\, f)\, g)((\lambda x.\, x)\, a) \arrow[dl, two heads, "\beta"'] \arrow[dr, two heads, "\beta"] & \\
g((\lambda x.\, x)\, a) \arrow[dr, two heads, "\beta"'] & & ((\lambda f.\, f) g)\, a \arrow[dl, two heads, "\beta"] \\
 & g\, a &
\end{tikzcd}$$

Termination

Do we always get a β-normal form as we apply β-reduction?

Termination

Do we always get a β-normal form as we apply β-reduction?

Typed Case: Forall A_α there exists a unique (up to α-conversion) β-normal term B such that $\mathrm{A} \twoheadrightarrow_\beta \mathrm{B}$

Termination

Do we always get a β-normal form as we apply β-reduction?

Typed Case: Forall A_α there exists a unique (up to α-conversion) β-normal term B such that $A \twoheadrightarrow_\beta B$

Untyped Case: Consider the term $\omega = (\lambda x.\, xx)$

$$(\lambda x.\, xx)(\lambda x.\, xx) \rightarrow^1_\beta \omega\omega$$

Def.: η-Conversion

A η-redex is a term of the form $(\lambda x_\beta . F_{\alpha\beta}\, x)$ where $x \notin FV(F)$. The η-reduct of this term is F.

Def.: η-Conversion

A η-redex is a term of the form $(\lambda x_\beta . F_{\alpha\beta}\, x)$ where $x \notin FV(F)$. The η-reduct of this term is F.

We say $M \rightarrow_\eta N$, ie. η-reduces in 1 step, if

$$\begin{aligned} M &= P[(\lambda x_\beta . F_{\alpha\beta} x)]_p \\ N &= P[F]_p \end{aligned}$$

Def.: η-Conversion

A η-redex is a term of the form $(\lambda x_\beta . F_{\alpha\beta}\, x)$ where $x \notin FV(F)$. The η-reduct of this term is F.

We say $M \to_\eta N$, ie. η-reduces in 1 step, if

$$\begin{aligned} M &= P[(\lambda x_\beta . F_{\alpha\beta} x)]_p \\ N &= P[F]_p \end{aligned}$$

We say $M \twoheadrightarrow_\eta N$, ie. η-reduces in several steps, if $\exists M^1, \ldots, M^n$ for $n \geq 1$ such that $M = M^1$ and $N = M^n$ and $M^i \to_\beta M^{i+1}$.

Def.: η-Normal Form

A term is called η-normal if it contains no η-redexes.

Thm.: Church-Rosser Property for $\twoheadrightarrow_\eta$

$$\begin{tikzcd}[column sep=small]
 & \mathsf{T}_\alpha \arrow[dl, two heads, "\eta"'] \arrow[dr, two heads, "\eta"] & \\
\mathsf{L}_\alpha \arrow[dr, two heads, "\eta"'] & & \mathsf{R}_\alpha \arrow[dl, two heads, "\eta"] \\
 & \mathsf{B}_\alpha &
\end{tikzcd}$$

Thm.: Church-Rosser Property for $\twoheadrightarrow_\eta$

$$
\begin{tikzcd}
 & \mathsf{T}_\alpha \arrow[dl, two heads, "\eta"'] \arrow[dr, two heads, "\eta"] & \\
\mathsf{L}_\alpha \arrow[dr, two heads, "\eta"'] & & \mathsf{R}_\alpha \arrow[dl, two heads, "\eta"] \\
 & \mathsf{B}_\alpha &
\end{tikzcd}
$$

If T_α η-reduces in multiple steps with one strategy to L_α and with another strategy to R_α then there exists a term B_α such that L_α and R_α η-reduce in multiple steps to B_α.

Thm.: Church-Rosser Property for $\twoheadrightarrow_\eta$

$$\begin{tikzcd}
 & T_\alpha \arrow[dl, two heads, "\eta"'] \arrow[dr, two heads, "\eta"] & \\
L_\alpha \arrow[dr, two heads, "\eta"'] & & R_\alpha \arrow[dl, two heads, "\eta"] \\
 & B_\alpha &
\end{tikzcd}$$

If T_α η-reduces in multiple steps with one strategy to L_α and with another strategy to R_α then there exists a term B_α such that L_α and R_α η-reduce in multiple steps to B_α.

The Church-Rosser Property for $\twoheadrightarrow_\eta$ holds for Λ and Λ^α.

Def.: $\beta\eta$-Conversion

$$\longrightarrow_{\beta\eta} := \longrightarrow_{\beta} \cup \longrightarrow_{\eta}$$

Def.: $\beta\eta$-Conversion

$$\longrightarrow_{\beta\eta} := \longrightarrow_{\beta} \cup \longrightarrow_{\eta}$$

If M $\longrightarrow_{\beta\eta}$ N we say M $\beta\eta$-reduces in 1 step to N.

Def.: $\beta\eta$-Conversion

$$\longrightarrow_{\beta\eta} := \longrightarrow_{\beta} \cup \longrightarrow_{\eta}$$

If $M \longrightarrow_{\beta\eta} N$ we say M $\beta\eta$-reduces in 1 step to N.

We say $M \twoheadrightarrow_{\beta\eta} N$, ie. η-reduces in several steps, if $\exists M^1, \ldots, M^n$ for $n \geq 1$ such that $M = M^1$ and $N = M^n$ and $M^i \longrightarrow_{\beta\eta} M^{i+1}$.

Def.: $\beta\eta$-Normal Form

A term is $\beta\eta$-normal if it contains no β-redexes and no η-redexes.

Thm.: Church-Rosser Property for $\twoheadrightarrow_{\beta\eta}$

$$\begin{tikzcd}
 & \mathrm{T}_\alpha \arrow[dl, dashed, two heads, "\beta\eta"'] \arrow[dr, dashed, two heads, "\beta\eta"] & \\
\mathrm{L}_\alpha \arrow[dr, dashed, two heads, "\beta\eta"'] & & \mathrm{R}_\alpha \arrow[dl, dashed, two heads, "\beta\eta"] \\
 & \mathrm{B}_\alpha &
\end{tikzcd}$$

Thm.: Church-Rosser Property for $\twoheadrightarrow_{\beta\eta}$

$$\begin{tikzcd} & \mathsf{T}_\alpha \arrow[dl, two heads, dashed, "\beta\eta"'] \arrow[dr, two heads, dashed, "\beta\eta"] & \\ \mathsf{L}_\alpha \arrow[dr, two heads, dashed, "\beta\eta"'] & & \mathsf{R}_\alpha \arrow[dl, two heads, dashed, "\beta\eta"] \\ & \mathsf{B}_\alpha & \end{tikzcd}$$

If T_α $\beta\eta$-reduces in multiple steps with one strategy to L_α and with another strategy to R_α then there exists a term B_α such that L_α and R_α $\beta\eta$-reduce in multiple steps to B_α.

Thm.: Church-Rosser Property for $\twoheadrightarrow_{\beta\eta}$

$$\begin{tikzcd}
 & \mathsf{T}_\alpha \arrow[dl, two heads, "\beta\eta"'] \arrow[dr, two heads, "\beta\eta"] & \\
\mathsf{L}_\alpha \arrow[dr, two heads, "\beta\eta"'] & & \mathsf{R}_\alpha \arrow[dl, two heads, "\beta\eta"] \\
 & \mathsf{B}_\alpha &
\end{tikzcd}$$

If T_α $\beta\eta$-reduces in multiple steps with one strategy to L_α and with another strategy to R_α then there exists a term B_α such that L_α and R_α $\beta\eta$-reduce in multiple steps to B_α.

The Church-Rosser Property for $\twoheadrightarrow_{\beta\eta}$ holds for Λ and Λ^α.

Thm.: Strong Church-Rosser Property

In Λ^{α} (simply typed λ-calculus) the relations $\twoheadrightarrow_{\beta}$ and $\twoheadrightarrow_{\beta\eta}$ have the strong Church Rosser property:

Thm.: Strong Church-Rosser Property

In Λ^α (simply typed λ-calculus) the relations $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_{\beta\eta}$ have the strong Church Rosser property: for very term A_τ there exists a unique (up to α-renaming) β-normal resp. $\beta\eta$-normal term B_τ such that $A_\tau \twoheadrightarrow_\beta B_\tau$ resp. $A_\tau \twoheadrightarrow_{\beta\eta} B_\tau$.

Def.: Long $\beta\eta$-Normal Form

Let $\mathsf{n} \geq 0$, $\alpha^1, \ldots, \alpha^\mathsf{n} \in \mathcal{T}$, and $\beta \in \{\mathsf{o}, \iota\}$. A term A of type $(\beta, \alpha^\mathsf{n}, \ldots, \alpha^1)$ is in long $\beta\eta$-normal form if it is of form

$$\lambda \mathsf{x}^1_{\alpha^1} \ldots \mathsf{x}^\mathsf{n}_{\alpha^\mathsf{n}}.(\mathsf{h}_{\beta\gamma^\mathsf{m}\ldots\gamma^1} \mathsf{A}^1_{\gamma^1} \ldots \mathsf{A}^\mathsf{m}_{\gamma^\mathsf{m}})$$

for a variable or constant $\mathsf{h}_{\beta\gamma^\mathsf{m}\ldots\gamma^1}$, $\mathsf{m} \geq 0$ and long $\beta\eta$-normal forms $\mathsf{A}^1_{\gamma^1}, \ldots, \mathsf{A}^\mathsf{m}_{\gamma^\mathsf{m}}$.

Def.: Long $\beta\eta$-Normal Form

Let $n \geq 0$, $\alpha^1, \ldots, \alpha^n \in \mathcal{T}$, and $\beta \in \{o, \iota\}$. A term A of type $(\beta, \alpha^n, \ldots, \alpha^1)$ is in long $\beta\eta$-normal form if it is of form

$$\lambda x^1_{\alpha^1} \ldots x^n_{\alpha^n}.(h_{\beta\gamma^m\ldots\gamma^1} A^1_{\gamma^1} \ldots A^m_{\gamma^m})$$

for a variable or constant $h_{\beta\gamma^m\ldots\gamma^1}$, $m \geq 0$ and long $\beta\eta$-normal forms $A^1_{\gamma^1}, \ldots, A^m_{\gamma^m}$. Note that this is an inductive definition; the base case is when $m = 0$.

Def.: Long $\beta\eta$-Normal Form

Let $\mathsf{n} \geq 0$, $\alpha^1, \ldots, \alpha^{\mathsf{n}} \in \mathcal{T}$, and $\beta \in \{\mathsf{o}, \iota\}$. A term A of type $(\beta, \alpha^{\mathsf{n}}, \ldots, \alpha^1)$ is in long $\beta\eta$-normal form if it is of form

$$\lambda \mathsf{x}^1_{\alpha^1} \ldots \mathsf{x}^{\mathsf{n}}_{\alpha^{\mathsf{n}}}.(\mathsf{h}_{\beta\gamma^{\mathsf{m}}\ldots\gamma^1} \mathsf{A}^1_{\gamma^1} \ldots \mathsf{A}^{\mathsf{m}}_{\gamma^{\mathsf{m}}})$$

for a variable or constant $\mathsf{h}_{\beta\gamma^{\mathsf{m}}\ldots\gamma^1}$, $\mathsf{m} \geq 0$ and long $\beta\eta$-normal forms $\mathsf{A}^1_{\gamma^1}, \ldots, \mathsf{A}^{\mathsf{m}}_{\gamma^{\mathsf{m}}}$. Note that this is an inductive definition; the base case is when $\mathsf{m} = 0$. Note that if $\lambda \overline{\mathsf{x}^{\mathsf{n}}}.(\mathsf{h}\overline{\mathsf{A}^{\mathsf{m}}})$ is in long $\beta\eta$-normal form then $(\mathsf{h}\overline{\mathsf{A}^{\mathsf{m}}})$ is of base type.

Ex.: Long $\beta\eta$-Normal Form

Consider the $\beta\eta$-normal term $\mathsf{f}_{\iota(\iota\iota)}$.

$$\mathsf{f}_{\iota(\iota\iota)}$$

$$\uparrow^{\eta}$$

$$\lambda \mathsf{w}_{\iota\iota}.\,(\mathsf{f}_{\iota(\iota\iota)}\mathsf{w}_{\iota\iota})$$

$$\uparrow^{\eta}$$

$$\lambda \mathsf{w}_{\iota\iota}.\,(\mathsf{f}(\lambda \mathsf{x}_{\iota}.\,\mathsf{w}_{\iota\iota}\mathsf{x}))$$

Thm.: Long $\beta\eta$-Normal Form

For every term A there is unique long $\beta\eta$-normal form B such that A $=^{\beta\eta}$ B.

Rem.: $\beta\eta$-Head Normal Form

Instead of terms in long $\beta\eta$-normal form we often use in practice terms in $\beta\eta$-head normal form.

Rem.: $\beta\eta$-Head Normal Form

Instead of terms in long $\beta\eta$-normal form we often use in practice terms in $\beta\eta$-head normal form. Definition is similar to long $\beta\eta$-normal, but we do not require the embedded terms $A^{i}_{\gamma^{i}}$ to be in normal form.

Notation

- $A{\downarrow_\beta}$ is the β-normal form of A.

Notation

- $A\downarrow_\beta$ is the β-normal form of A.
- $A\downarrow_\eta$ is the η-normal form of A.

Notation

- $A{\downarrow_\beta}$ is the β-normal form of A.
- $A{\downarrow_\eta}$ is the η-normal form of A.
- $A{\downarrow}$ is the $\beta\eta$-normal form of A.

Notation

- $A\downarrow_\beta$ is the β-normal form of A.
- $A\downarrow_\eta$ is the η-normal form of A.
- $A\downarrow$ is the $\beta\eta$-normal form of A.
- $A\updownarrow$ is the long $\beta\eta$-normal form of A.

Semantics: Σ-Evaluations

Ex.: An Interesting Applicative Structure

$\mathsf{D}_\alpha := \{\mathsf{A}_\alpha \in \Lambda_\alpha | \mathsf{A} \text{ is closed}\}$.

- Is D_α non-empty for all α?

Ex.: An Interesting Applicative Structure

$\mathsf{D}_\alpha := \{\mathsf{A}_\alpha \in \Lambda_\alpha \,|\, \mathsf{A} \text{ is closed}\}$.

- Is D_α non-empty for all α?
- If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$, then $\forall \alpha \in \mathcal{T}.\, \Lambda_\alpha \neq \emptyset$.

Ex.: An Interesting Applicative Structure

$\mathsf{D}_\alpha := \{\mathsf{A}_\alpha \in \Lambda_\alpha | \, \mathsf{A} \text{ is closed}\}$.

- Is D_α non-empty for all α?
- If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$, then $\forall \alpha \in \mathcal{T}. \Lambda_\alpha \neq \emptyset$.
- Is $\mathsf{D}_{\alpha\beta}$ a set of functions? (ie. $\mathsf{D}_{\alpha\beta} \subseteq (\mathsf{D}_\alpha)^{\mathsf{D}_\beta}$?) — No!

Ex.: An Interesting Applicative Structure

$\mathsf{D}_\alpha := \{\mathsf{A}_\alpha \in \Lambda_\alpha | \, \mathsf{A} \text{ is closed}\}$.

- Is D_α non-empty for all α?
- If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$, then $\forall \alpha \in \mathcal{T}. \Lambda_\alpha \neq \emptyset$.
- Is $\mathsf{D}_{\alpha\beta}$ a set of functions? (ie. $\mathsf{D}_{\alpha\beta} \subseteq (\mathsf{D}_\alpha)^{\mathsf{D}_\beta}$?) — No!
- Is $(\lambda \mathsf{x}_\iota \, \mathsf{x}) \in \mathsf{D}_{\iota\iota}$? — Yes!

Ex.: An Interesting Applicative Structure

$D_\alpha := \{A_\alpha \in \Lambda_\alpha | \, A \text{ is closed}\}$.

- Is D_α non-empty for all α?
- If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$, then $\forall \alpha \in \mathcal{T}. \Lambda_\alpha \neq \emptyset$.
- Is $D_{\alpha\beta}$ a set of functions? (ie. $D_{\alpha\beta} \subseteq (D_\alpha)^{D_\beta}$?) — No!
- Is $(\lambda x_\iota \, x) \in D_{\iota\iota}$? — Yes!
- $D = (D_\alpha)_{\alpha \in \mathcal{T}}$ is not a frame!

Ex.: An Interesting Applicative Structure

$D_\alpha := \{A_\alpha \in \Lambda_\alpha | \, A \text{ is closed}\}$.

- Is D_α non-empty for all α?
- If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$, then $\forall \alpha \in \mathcal{T}. \Lambda_\alpha \neq \emptyset$.
- Is $D_{\alpha\beta}$ a set of functions? (ie. $D_{\alpha\beta} \subseteq (D_\alpha)^{D_\beta}$?) — No!
- Is $(\lambda x_\iota \, x) \in D_{\iota\iota}$? — Yes!
- $D = (D_\alpha)_{\alpha \in \mathcal{T}}$ is not a frame!
- It requires a specific application operator $@ : D_{\alpha\beta} \times D_\beta \longrightarrow D_\alpha$

Ex.: An Interesting Applicative Structure

$D_\alpha := \{A_\alpha \in \Lambda_\alpha | \, A \text{ is closed}\}$.

- Is D_α non-empty for all α?
- If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$, then $\forall \alpha \in \mathcal{T} . \Lambda_\alpha \neq \emptyset$.
- Is $D_{\alpha\beta}$ a set of functions? (ie. $D_{\alpha\beta} \subseteq (D_\alpha)^{D_\beta}$?) — No!
- Is $(\lambda x_\iota \, x) \in D_{\iota\iota}$? — Yes!
- $D = (D_\alpha)_{\alpha \in \mathcal{T}}$ is not a frame!
- It requires a specific application operator $@ : D_{\alpha\beta} \times D_\beta \longrightarrow D_\alpha$
- If Λ_α is non-empty for all $\alpha \in \mathcal{T}$, then $< D, @ >$ is an applicative structure.

Ex.: Interpretation of Terms

Syntax Semantics $< \mathrm{D}, @ >$

$(\lambda \mathrm{x}_\iota . \, \mathrm{x})$

Ex.: Interpretation of Terms

Syntax	Semantics $< \mathsf{D}, @ >$
$(\lambda \mathsf{x}_\iota . \mathsf{x})$	$(\lambda \mathsf{x}_\iota . \mathsf{x})$

Ex.: Interpretation of Terms

Syntax	Semantics	$< D, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$

Ex.: Interpretation of Terms

Syntax	Semantics	$< \mathsf{D}, @ >$
$(\lambda \mathsf{x}_\iota . \mathsf{x})$	$(\lambda \mathsf{x}_\iota . \mathsf{x})$	$\in \mathsf{D}_{\iota\iota}$
y_ι		

Ex.: Interpretation of Terms

Syntax	Semantics	$< \mathsf{D}, @ >$
$(\lambda \mathsf{x}_\iota . \mathsf{x})$	$(\lambda \mathsf{x}_\iota . \mathsf{x})$	$\in \mathsf{D}_{\iota\iota}$
y_ι	$\varphi(\mathsf{y})$	

Ex.: Interpretation of Terms

Syntax	Semantics	$< D, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$
y_ι	$\varphi(y)$	$\in D_\iota$

Ex.: Interpretation of Terms

Syntax	Semantics	$< D, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$
y_ι	$\varphi(y)$	$\in D_\iota$
$a_\iota \in C$		

Ex.: Interpretation of Terms

Syntax	Semantics	$< D, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$
y_ι	$\varphi(y)$	$\in D_\iota$
$a_\iota \in C$	a	

Ex.: Interpretation of Terms

Syntax	Semantics	$< \mathsf{D}, @ >$
$(\lambda \mathsf{x}_\iota . \mathsf{x})$	$(\lambda \mathsf{x}_\iota . \mathsf{x})$	$\in \mathsf{D}_{\iota\iota}$
y_ι	$\varphi(\mathsf{y})$	$\in \mathsf{D}_\iota$
$\mathsf{a}_\iota \in \mathsf{C}$	a	$\in \mathsf{D}_\iota$

Ex.: Interpretation of Terms

Syntax	Semantics	$< D, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$
y_ι	$\varphi(y)$	$\in D_\iota$
$a_\iota \in C$	a	$\in D_\iota$
$(\lambda x_\iota . x) a_\iota$		

Ex.: Interpretation of Terms

Syntax	Semantics	$< D, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$
y_ι	$\varphi(y)$	$\in D_\iota$
$a_\iota \in C$	a	$\in D_\iota$
$(\lambda x_\iota . x) a_\iota$	$(\lambda x_\iota . x) @ a_\iota$	

Ex.: Interpretation of Terms

Syntax	Semantics	$< D, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$
y_ι	$\varphi(y)$	$\in D_\iota$
$a_\iota \in C$	a	$\in D_\iota$
$(\lambda x_\iota . x) a_\iota$	$(\lambda x_\iota . x) @ a_\iota$	$\in D_\iota$

Ex.: Interpretation of Terms

Syntax	Semantics	$< \mathsf{D}, @ >$
$(\lambda x_\iota . x)$	$(\lambda x_\iota . x)$	$\in D_{\iota\iota}$
y_ι	$\varphi(y)$	$\in D_\iota$
$a_\iota \in C$	a	$\in D_\iota$
$(\lambda x_\iota . x) a_\iota$	$(\lambda x_\iota . x) @ a_\iota$	$\in D_\iota$

Remark: The variable y_ι is a non-closed well-formed formula of type ι. We need an assignment $\varphi_\alpha : V_\alpha \to D_\alpha$ to give it a meaning.

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha \downarrow_\beta := \{\mathsf{A}_\alpha \in \Lambda_\alpha |\, \mathsf{A}$ is closed and A is in β-normal form$\}$

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha \downarrow_\beta := \{\mathsf{A}_\alpha \in \Lambda_\alpha | \, \mathsf{A}$ is closed and A is in β-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha \downarrow_\beta)_{\alpha \in \mathcal{T}}$

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha\downarrow_\beta := \{\mathsf{A}_\alpha \in \Lambda_\alpha \mid \mathsf{A}$ is closed and A is in β-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha\downarrow_\beta)_{\alpha\in\mathcal{T}}$
- Let $@^\beta_{\gamma\delta} : \mathsf{D}_{\gamma\delta} \times \mathsf{D}_\delta \longrightarrow \mathsf{D}_\gamma$ be defined by

$$\mathsf{F}_{\gamma\delta} @^\beta_{\gamma\delta} \mathsf{G}_\delta = (\mathsf{F}\,\mathsf{G})\downarrow_\beta$$

for all $\mathsf{F}_{\gamma\delta} \in \mathsf{D}_{\gamma\delta}$ and $\mathsf{G}_\delta \in \mathsf{D}_\delta$.

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha \downarrow_\beta := \{\mathsf{A}_\alpha \in \Lambda_\alpha |\, \mathsf{A}$ is closed and A is in β-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha \downarrow_\beta)_{\alpha \in \mathcal{T}}$
- Let $@^\beta_{\gamma\delta} : \mathsf{D}_{\gamma\delta} \times \mathsf{D}_\delta \longrightarrow \mathsf{D}_\gamma$ be defined by

$$\mathsf{F}_{\gamma\delta} @^\beta_{\gamma\delta} \mathsf{G}_\delta = (\mathsf{F}\,\mathsf{G}) \downarrow_\beta$$

  for all $\mathsf{F}_{\gamma\delta} \in \mathsf{D}_{\gamma\delta}$ and $\mathsf{G}_\delta \in \mathsf{D}_\delta$.
- $@^\beta = (@^\beta_{\gamma\delta})_{\gamma\delta \in \mathcal{T}}$

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha\downarrow_\beta := \{\mathsf{A}_\alpha \in \Lambda_\alpha \,|\, \mathsf{A}$ is closed and A is in β-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha\downarrow_\beta)_{\alpha\in\mathcal{T}}$
- Let $@^\beta_{\gamma\delta} : \mathsf{D}_{\gamma\delta} \times \mathsf{D}_\delta \longrightarrow \mathsf{D}_\gamma$ be defined by

$$\mathsf{F}_{\gamma\delta} @^\beta_{\gamma\delta} \mathsf{G}_\delta = (\mathsf{F}\,\mathsf{G})\downarrow_\beta$$

  for all $\mathsf{F}_{\gamma\delta} \in \mathsf{D}_{\gamma\delta}$ and $\mathsf{G}_\delta \in \mathsf{D}_\delta$.

- $@^\beta = (@^\beta_{\gamma\delta})_{\gamma\delta\in\mathcal{T}}$

Claim: If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$ (i.e., at least one constant for each base type is given), then $(\mathsf{D}, @^\beta)$ is an applicative structure.

Ex.: Interesting Applicative Structures

Proof:

- Is $\mathrm{D}_\alpha \downarrow_\beta$ nonempty for all $\alpha \in \mathcal{T}$?

Ex.: Interesting Applicative Structures

Proof:

- Is $\mathrm{D}_\alpha \downarrow_\beta$ nonempty for all $\alpha \in \mathcal{T}$?
- Yes! This follows since $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_? \neq \emptyset$.

Ex.: Interesting Applicative Structures

Proof:

- Is $\mathsf{D}_\alpha\downarrow_\beta$ nonempty for all $\alpha \in \mathcal{T}$?
- Yes! This follows since $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_\wr \neq \emptyset$.
- Is $\mathsf{F}_{\gamma\delta} @^{\beta}_{\gamma\delta} \mathsf{G}_\delta \in \mathsf{D}_\gamma \downarrow_\beta$?

Ex.: Interesting Applicative Structures

Proof:

- Is $\mathsf{D}_\alpha\downarrow_\beta$ nonempty for all $\alpha \in \mathcal{T}$?
- Yes! This follows since $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_\wr \neq \emptyset$.
- Is $\mathsf{F}_{\gamma\delta}@^{\beta}_{\gamma\delta}\mathsf{G}_\delta \in \mathsf{D}_\gamma\downarrow_\beta$?
- Let's check: $\mathsf{F}_{\gamma\delta}@^{\beta}_{\gamma\delta}\mathsf{G}_\delta = (\mathsf{F}\,\mathsf{G})\downarrow_\beta \in \mathsf{D}_\gamma\downarrow_\beta$

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha\downarrow_{\beta\eta} := \{\mathsf{A}_\alpha \in \Lambda_\alpha |\ \mathsf{A}$ is closed and A is in $\beta\eta$-normal form$\}$

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha\downarrow_{\beta\eta} := \{\mathsf{A}_\alpha \in \Lambda_\alpha |$ A is closed and A is in $\beta\eta$-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha\downarrow_{\beta\eta})_{\alpha\in\mathcal{T}}$

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha \downarrow_{\beta\eta} := \{\mathsf{A}_\alpha \in \Lambda_\alpha |\, \mathsf{A}$ is closed and A is in $\beta\eta$-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha \downarrow_{\beta\eta})_{\alpha \in \mathcal{T}}$
- Let $@^{\beta\eta}_{\gamma\delta} : \mathsf{D}_{\gamma\delta} \times \mathsf{D}_\delta \longrightarrow \mathsf{D}_\gamma$ be defined by

$$\mathsf{F}_{\gamma\delta} @^{\beta\eta}_{\gamma\delta} \mathsf{G}_\delta = (\mathsf{F}\,\mathsf{G}) \downarrow$$

for all $\mathsf{F}_{\gamma\delta} \in \mathsf{D}_{\gamma\delta}$ and $\mathsf{G}_\delta \in \mathsf{D}_\delta$.

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha \downarrow_{\beta\eta} := \{\mathsf{A}_\alpha \in \Lambda_\alpha | \, \mathsf{A}$ is closed and A is in $\beta\eta$-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha \downarrow_{\beta\eta})_{\alpha \in \mathcal{T}}$
- Let $@^{\beta\eta}_{\gamma\delta} : \mathsf{D}_{\gamma\delta} \times \mathsf{D}_\delta \longrightarrow \mathsf{D}_\gamma$ be defined by

$$\mathsf{F}_{\gamma\delta} @^{\beta\eta}_{\gamma\delta} \mathsf{G}_\delta = (\mathsf{F}\,\mathsf{G}) \downarrow$$

  for all $\mathsf{F}_{\gamma\delta} \in \mathsf{D}_{\gamma\delta}$ and $\mathsf{G}_\delta \in \mathsf{D}_\delta$.
- $@^{\beta\eta} = (@^{\beta\eta}_{\gamma\delta})_{\gamma\delta \in \mathcal{T}}$

Ex.: Interesting Applicative Structures

- Let $\mathsf{D}_\alpha\downarrow_{\beta\eta} := \{\mathsf{A}_\alpha \in \Lambda_\alpha \mid \mathsf{A}$ is closed and A is in $\beta\eta$-normal form$\}$
- Let $\mathsf{D} := (\mathsf{D}_\alpha\downarrow_{\beta\eta})_{\alpha\in\mathcal{T}}$
- Let $@^{\beta\eta}_{\gamma\delta} : \mathsf{D}_{\gamma\delta} \times \mathsf{D}_\delta \longrightarrow \mathsf{D}_\gamma$ be defined by

$$\mathsf{F}_{\gamma\delta} @^{\beta\eta}_{\gamma\delta} \mathsf{G}_\delta = (\mathsf{F}\,\mathsf{G})\downarrow$$

  for all $\mathsf{F}_{\gamma\delta} \in \mathsf{D}_{\gamma\delta}$ and $\mathsf{G}_\delta \in \mathsf{D}_\delta$.
- $@^{\beta\eta} = (@^{\beta\eta}_{\gamma\delta})_{\gamma\delta\in\mathcal{T}}$

Claim: If $\mathcal{C}_\iota \neq \emptyset$ and $\mathcal{C}_o \neq \emptyset$ (i.e., at least one constant for each base type is given), then $(\mathsf{D}, @^{\beta\eta})$ is an applicative structure.

Ex.: Interesting Applicative Structures

Proof:

- . . . analogous . . .

Def.: Variable Assignment

Let $\mathcal{A} := (\mathcal{D}, @)$ be an applicative structure.

Def.: Variable Assignment

Let $\mathcal{A} := (\mathcal{D}, @)$ be an applicative structure.

A typed function $\varphi : \mathcal{V} \longrightarrow \mathcal{D} := (\varphi_\alpha : \mathcal{V}_\alpha \longrightarrow \mathcal{D}_\alpha)_{\alpha \in \mathcal{T}}$ is called a variable assignment into $\mathcal{A}$.

Def.: Variable Assignment

Let $\mathcal{A} := (\mathcal{D}, @)$ be an applicative structure.

A typed function $\varphi\colon \mathcal{V} \longrightarrow \mathcal{D} := (\varphi_\alpha\colon \mathcal{V}_\alpha \longrightarrow \mathcal{D}_\alpha)_{\alpha \in \mathcal{T}}$ is called a variable assignment into $\mathcal{A}$.

Given a variable assignment φ, variable X_α, and value $\mathsf{a} \in \mathcal{D}_\alpha$,

Def.: Variable Assignment

Let $\mathcal{A} := (\mathcal{D}, @)$ be an applicative structure.

A typed function $\varphi : \mathcal{V} \longrightarrow \mathcal{D} := (\varphi_\alpha : \mathcal{V}_\alpha \longrightarrow \mathcal{D}_\alpha)_{\alpha \in \mathcal{T}}$ is called a variable assignment into $\mathcal{A}$.

Given a variable assignment φ, variable X_α, and value $\mathsf{a} \in \mathcal{D}_\alpha$, we use $\varphi, [\mathsf{a}/\mathsf{X}]$ to denote the variable assignment with

Def.: Variable Assignment

Let $\mathcal{A} := (\mathcal{D}, @)$ be an applicative structure.

A typed function $\varphi{:}\,\mathcal{V} \longrightarrow \mathcal{D} := (\varphi_\alpha{:}\,\mathcal{V}_\alpha \longrightarrow \mathcal{D}_\alpha)_{\alpha\in\mathcal{T}}$ is called a variable assignment into $\mathcal{A}$.

Given a variable assignment φ, variable X_α, and value $\mathsf{a} \in \mathcal{D}_\alpha$, we use $\varphi, [\mathsf{a}/\mathsf{X}]$ to denote the variable assignment with

$$(\varphi, [\mathsf{a}/\mathsf{X}])(\mathsf{X}) = \mathsf{a}$$

Def.: Variable Assignment

Let $\mathcal{A} := (\mathcal{D}, @)$ be an applicative structure.

A typed function $\varphi\colon \mathcal{V} \longrightarrow \mathcal{D} := (\varphi_\alpha\colon \mathcal{V}_\alpha \longrightarrow \mathcal{D}_\alpha)_{\alpha\in\mathcal{T}}$ is called a variable assignment into $\mathcal{A}$.

Given a variable assignment φ, variable X_α, and value $\mathsf{a} \in \mathcal{D}_\alpha$, we use $\varphi, [\mathsf{a}/\mathsf{X}]$ to denote the variable assignment with

$$(\varphi, [\mathsf{a}/\mathsf{X}])(\mathsf{X}) = \mathsf{a}$$

and

$$(\varphi, [\mathsf{a}/\mathsf{X}])(\mathsf{Y}) = \varphi(\mathsf{Y})$$

for variables Y other than X.

Some Assumptions

From now on, we assume the signature $\Sigma_\alpha = (\mathcal{V}, \mathcal{C})$ to be infinite for each type α.

Some Assumptions

From now on, we assume the signature $\Sigma_\alpha = (\mathcal{V}, \mathcal{C})$ to be infinite for each type α. Furthermore, we assume there is a particular cardinal $\aleph_{\mathsf{s}}$ such that Σ_α has cardinality $\aleph_{\mathsf{s}}$ for every type α.

Some Assumptions

From now on, we assume the signature $\Sigma_\alpha = (\mathcal{V}, \mathcal{C})$ to be infinite for each type α. Furthermore, we assume there is a particular cardinal $\aleph_s$ such that Σ_α has cardinality $\aleph_s$ for every type α. Since $\mathcal{V}$ is countable, this implies $\mathit{wff}_\alpha(\Sigma) := \Lambda^\alpha$ and $\mathit{cwff}_\alpha(\Sigma) := \{\mathsf{A} \in \Lambda^\alpha | \mathsf{A}\ \text{closed}\}$ have cardinality $\aleph_s$ for each type α.

Some Assumptions

From now on, we assume the signature $\Sigma_\alpha = (\mathcal{V}, \mathcal{C})$ to be infinite for each type α. Furthermore, we assume there is a particular cardinal $\aleph_s$ such that Σ_α has cardinality $\aleph_s$ for every type α. Since $\mathcal{V}$ is countable, this implies $\mathit{wff}_\alpha(\Sigma) := \Lambda^\alpha$ and $\mathit{cwff}_\alpha(\Sigma) := \{\mathsf{A} \in \Lambda^\alpha | \mathsf{A}\,\text{closed}\}$ have cardinality $\aleph_s$ for each type α. Also, whether or not primitive equality is included in the signature, there can only be finitely many logical constants in Σ_α for each particular type α.

Some Assumptions

From now on, we assume the signature $\Sigma_\alpha = (\mathcal{V}, \mathcal{C})$ to be infinite for each type α. Furthermore, we assume there is a particular cardinal $\aleph_s$ such that Σ_α has cardinality $\aleph_s$ for every type α. Since $\mathcal{V}$ is countable, this implies $\mathit{wff}_\alpha(\Sigma) := \Lambda^\alpha$ and $\mathit{cwff}_\alpha(\Sigma) := \{A \in \Lambda^\alpha | A \text{ closed}\}$ have cardinality $\aleph_s$ for each type α. Also, whether or not primitive equality is included in the signature, there can only be finitely many logical constants in Σ_α for each particular type α. Thus, the cardinality of the set of parameters in Σ_α is also $\aleph_s$. In the countable case, $\aleph_s$ is $\aleph_0$.

Σ-Evaluations

Let Σ be a signature.

Σ-Evaluations

Let Σ be a signature. We build on the notion of applicative structures to define Σ-evaluations, where the evaluation function is assumed to respect application and β-conversion.

Σ-Evaluations

Let Σ be a signature. We build on the notion of applicative structures to define Σ-evaluations, where the evaluation function is assumed to respect application and β-conversion.

In such models, a function is not uniquely determined by its behavior on all possible arguments.

Σ-Evaluations

Let Σ be a signature. We build on the notion of applicative structures to define Σ-evaluations, where the evaluation function is assumed to respect application and β-conversion.

In such models, a function is not uniquely determined by its behavior on all possible arguments.

Such models can be constructed, for example, by labeling for functions (e.g., a green and a red version of a function f) in order to differentiate between them, even though they are functionally equivalent.

Σ-Evaluations

Let $\mathcal{E}\colon \mathcal{F}_{\mathcal{T}}(\mathcal{V};\mathcal{D}) \longrightarrow \mathcal{F}_{\mathcal{T}}(\mathit{wff}(\Sigma),\mathcal{D})$ be a total function, where $\mathcal{F}_{\mathcal{T}}(\mathcal{V};\mathcal{D})$ is the set of variable assignments and $\mathcal{F}_{\mathcal{T}}(\mathit{wff}(\Sigma),\mathcal{D})$ is the set of typed functions mapping terms into objects in $\mathcal{D}$.

Σ-Evaluations

Let $\mathcal{E}\colon \mathcal{F}_{\mathcal{T}}(\mathcal{V};\mathcal{D}) \longrightarrow \mathcal{F}_{\mathcal{T}}(\mathit{wff}(\Sigma),\mathcal{D})$ be a total function, where $\mathcal{F}_{\mathcal{T}}(\mathcal{V};\mathcal{D})$ is the set of variable assignments and $\mathcal{F}_{\mathcal{T}}(\mathit{wff}(\Sigma),\mathcal{D})$ is the set of typed functions mapping terms into objects in $\mathcal{D}$. We will write the argument of $\mathcal{E}$ as a subscript. So, for each assignment φ, we have a typed function

$$\mathcal{E}_{\varphi}\colon \mathit{wff}(\Sigma) \longrightarrow \mathcal{D}$$

Σ-Evaluations

Let $\mathcal{E}\colon \mathcal{F}_{\mathcal{T}}(\mathcal{V};\mathcal{D}) \longrightarrow \mathcal{F}_{\mathcal{T}}(\mathit{wff}(\Sigma),\mathcal{D})$ be a total function, where $\mathcal{F}_{\mathcal{T}}(\mathcal{V};\mathcal{D})$ is the set of variable assignments and $\mathcal{F}_{\mathcal{T}}(\mathit{wff}(\Sigma),\mathcal{D})$ is the set of typed functions mapping terms into objects in $\mathcal{D}$. We will write the argument of $\mathcal{E}$ as a subscript. So, for each assignment φ, we have a typed function

$$\mathcal{E}_{\varphi}\colon \mathit{wff}(\Sigma) \longrightarrow \mathcal{D}$$

What properties shall $\mathcal{E}$ fulfill?

Def.: Evaluation Function

$\mathcal{E}$ is called an evaluation function for an applicative structure $\mathcal{A} = (\mathcal{D}, @)$

Def.: Evaluation Function

$\mathcal{E}$ is called an evaluation function for an applicative structure $\mathcal{A} = (\mathcal{D}, @)$ if for any assignments φ and ψ into $\mathcal{A}$, we have

Def.: Evaluation Function

$\mathcal{E}$ is called an evaluation function for an applicative structure $\mathcal{A} = (\mathcal{D}, @)$ if for any assignments φ and ψ into $\mathcal{A}$, we have

1. $\mathcal{E}_\varphi|_\mathcal{V} = \varphi$

Def.: Evaluation Function

$\mathcal{E}$ is called an evaluation function for an applicative structure $\mathcal{A} = (\mathcal{D}, @)$ if for any assignments φ and ψ into $\mathcal{A}$, we have

1. $\mathcal{E}_\varphi|_\mathcal{V} = \varphi$
2. $\mathcal{E}_\varphi(\mathbf{FA}) = \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$ for any $\mathbf{F} \in \mathit{wff}_{\alpha\rightarrow\beta}(\Sigma)$ and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and types α and β.

Def.: Evaluation Function

$\mathcal{E}$ is called an evaluation function for an applicative structure $\mathcal{A} = (\mathcal{D}, @)$ if for any assignments φ and ψ into $\mathcal{A}$, we have

1. $\mathcal{E}_\varphi|_\mathcal{V} = \varphi$
2. $\mathcal{E}_\varphi(\mathbf{FA}) = \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$ for any $\mathbf{F} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and types α and β.
3. $\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\psi(\mathbf{A})$ for any type α and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$, whenever φ and ψ coincide on $\mathrm{FV}(\mathbf{A})$.

Def.: Evaluation Function

$\mathcal{E}$ is called an evaluation function for an applicative structure $\mathcal{A} = (\mathcal{D}, @)$ if for any assignments φ and ψ into $\mathcal{A}$, we have

1. $\mathcal{E}_\varphi|_\mathcal{V} = \varphi$
2. $\mathcal{E}_\varphi(\mathbf{FA}) = \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$ for any $\mathbf{F} \in \mathit{wff}_{\alpha\rightarrow\beta}(\Sigma)$ and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and types α and β.
3. $\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\psi(\mathbf{A})$ for any type α and $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$, whenever φ and ψ coincide on $\mathrm{FV}(\mathbf{A})$.
4. $\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{A}{\downarrow_\beta})$ for all $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$.

Def.: Σ-Evaluation

We call $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ a Σ-evaluation if $(\mathcal{D}, @)$ is an applicative structure and $\mathcal{E}$ is an evaluation function for $(\mathcal{D}, @)$. We call $\mathcal{E}_\varphi(\mathbf{A}_\alpha) \in \mathcal{D}_\alpha$ the denotation of $\mathbf{A}_\alpha$ in $\mathcal{J}$ for φ.

Def.: Σ-Evaluation

We call $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ a Σ-evaluation if $(\mathcal{D}, @)$ is an applicative structure and $\mathcal{E}$ is an evaluation function for $(\mathcal{D}, @)$. We call $\mathcal{E}_\varphi(\mathbf{A}_\alpha) \in \mathcal{D}_\alpha$ the denotation of $\mathbf{A}_\alpha$ in $\mathcal{J}$ for φ.

Remark: since $\mathcal{E}$ is a function, the denotation in $\mathcal{J}$ is unique. However, for a given applicative structure $\mathcal{A}$, there may be many possible evaluation functions.

Def.: Σ-Evaluation

We call $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ a Σ-evaluation if $(\mathcal{D}, @)$ is an applicative structure and $\mathcal{E}$ is an evaluation function for $(\mathcal{D}, @)$. We call $\mathcal{E}_\varphi(\mathbf{A}_\alpha) \in \mathcal{D}_\alpha$ the denotation of $\mathbf{A}_\alpha$ in $\mathcal{J}$ for φ.

Remark: since $\mathcal{E}$ is a function, the denotation in $\mathcal{J}$ is unique. However, for a given applicative structure $\mathcal{A}$, there may be many possible evaluation functions.

If $\mathbf{A}$ is a closed formula, then $\mathcal{E}_\varphi(\mathbf{A})$ is independent of φ, since $\mathbf{Free}(\mathbf{A}) = \emptyset$. In these cases we sometimes drop the reference to φ from $\mathcal{E}_\varphi(\mathbf{A})$ and simply write $\mathcal{E}(\mathbf{A})$.

Def.: Functional/Full/Standard Σ-Eval.

We call a Σ-evaluation $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ functional [full, standard] if the applicative structure $(\mathcal{D}, @)$ is functional [full, standard].

Def.: Functional/Full/Standard Σ-Eval.

We call a Σ-evaluation $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ functional [full, standard] if the applicative structure $(\mathcal{D}, @)$ is functional [full, standard].

We say $\mathcal{J}$ is a Σ-evaluation over a frame if $(\mathcal{D}, @)$ is a frame.

What is the Idea?

Σ-evaluations generalize Σ-evaluations over frames, which are the basis for Henkin models, to the non-functional case.

What is the Idea?

Σ-evaluations generalize Σ-evaluations over frames, which are the basis for Henkin models, to the non-functional case.

The existence of an evaluation function that meets the conditions as presented seems to be the weakest situation where one would like to speak of a model.

What is the Idea?

Σ-evaluations generalize Σ-evaluations over frames, which are the basis for Henkin models, to the non-functional case.

The existence of an evaluation function that meets the conditions as presented seems to be the weakest situation where one would like to speak of a model.

We cannot in general assume the evaluation function is uniquely determined by its values on constants as this requires functionality.

What is the Idea?

Σ-evaluations generalize Σ-evaluations over frames, which are the basis for Henkin models, to the non-functional case.

The existence of an evaluation function that meets the conditions as presented seems to be the weakest situation where one would like to speak of a model.

We cannot in general assume the evaluation function is uniquely determined by its values on constants as this requires functionality. Example: two evaluation functions $\mathcal{E}$ and $\mathcal{E}'$ on the same applicative structure may agree on all constants, but give a different value to the term $(\lambda X_\iota . X)$.

Lemma: Σ-Evaluations respect β-Equality

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation and $\mathbf{A} =_\beta \mathbf{B}$. For all assignments φ into $(\mathcal{D}, @)$, we have

.

Lemma: Σ-Evaluations respect β-Equality

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation and $\mathbf{A} =_\beta \mathbf{B}$. For all assignments φ into $(\mathcal{D}, @)$, we have

$$\mathcal{E}_\varphi(\mathbf{A}) = \qquad\qquad\qquad\qquad = \mathcal{E}_\varphi(\mathbf{B})$$

.

Lemma: Σ-Evaluations respect β-Equality

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation and $\mathbf{A} =_\beta \mathbf{B}$. For all assignments φ into $(\mathcal{D}, @)$, we have

$$\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{A}{\downarrow_\beta}) \quad \mathcal{E}_\varphi(\mathbf{B}{\downarrow_\beta}) = \mathcal{E}_\varphi(\mathbf{B})$$

.

Lemma: Σ-Evaluations respect β-Equality

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation and $\mathbf{A} =_\beta \mathbf{B}$. For all assignments φ into $(\mathcal{D}, @)$, we have

$$\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{A}{\downarrow_\beta}) = \mathcal{E}_\varphi(\mathbf{B}{\downarrow_\beta}) = \mathcal{E}_\varphi(\mathbf{B})$$

.

Thm.: Substitution-Value Lemma

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation and φ be an assignment into $\mathcal{J}$.

Thm.: Substitution-Value Lemma

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation and φ be an assignment into $\mathcal{J}$. For any types α and β, variables X_β, and formulae $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma)$, we have

Thm.: Substitution-Value Lemma

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation and φ be an assignment into $\mathcal{J}$. For any types α and β, variables X_β, and formulae $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma)$, we have

$$\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathrm{X}]}(\mathbf{A}) = \mathcal{E}_\varphi([\mathbf{B}/\mathrm{X}]\mathbf{A})$$

.

Prf.: Substitution-Value Lemma

Proof:

Prf.: Substitution-Value Lemma

Proof: Using the fact that $\mathcal{E}$ respects β-equality and the other properties of $\mathcal{E}$, we can compute

Prf.: Substitution-Value Lemma

Proof: Using the fact that $\mathcal{E}$ respects β-equality and the other properties of $\mathcal{E}$, we can compute

$$\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathrm{X}]}(\mathbf{A}) \quad =$$

Prf.: Substitution-Value Lemma

Proof: Using the fact that $\mathcal{E}$ respects β-equality and the other properties of $\mathcal{E}$, we can compute

$$\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathrm{X}]}(\mathbf{A}) = \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathrm{X}]}((\lambda \mathrm{X}.\mathbf{A})\mathrm{X})$$

.

Prf.: Substitution-Value Lemma

Proof: Using the fact that $\mathcal{E}$ respects β-equality and the other properties of $\mathcal{E}$, we can compute

$$\begin{aligned}
\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathbf{A}) &= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}((\lambda \mathsf{X}.\mathbf{A})\mathsf{X}) \\
&= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\lambda \mathsf{X}.\mathbf{A})@\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathsf{X})
\end{aligned}$$

.

Prf.: Substitution-Value Lemma

Proof: Using the fact that $\mathcal{E}$ respects β-equality and the other properties of $\mathcal{E}$, we can compute

$$\begin{aligned}
\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathbf{A}) &= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}((\lambda \mathsf{X}.\mathbf{A})\mathsf{X}) \\
&= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\lambda \mathsf{X}.\mathbf{A})@\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathsf{X}) \\
&= \mathcal{E}_\varphi(\lambda \mathsf{X}.\mathbf{A})@\mathcal{E}_\varphi(\mathbf{B})
\end{aligned}$$

.

Prf.: Substitution-Value Lemma

Proof: Using the fact that $\mathcal{E}$ respects β-equality and the other properties of $\mathcal{E}$, we can compute

$$\begin{aligned}
\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathbf{A}) &= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}((\lambda\mathsf{X}_{\bullet}\mathbf{A})\mathsf{X}) \\
&= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\lambda\mathsf{X}_{\bullet}\mathbf{A})@\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathsf{X}) \\
&= \mathcal{E}_\varphi(\lambda\mathsf{X}_{\bullet}\mathbf{A})@\mathcal{E}_\varphi(\mathbf{B}) \\
&= \mathcal{E}_\varphi((\lambda\mathsf{X}_{\bullet}\mathbf{A})\mathbf{B})
\end{aligned}$$

.

Prf.: Substitution-Value Lemma

Proof: Using the fact that $\mathcal{E}$ respects β-equality and the other properties of $\mathcal{E}$, we can compute

$$\begin{aligned}
\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathbf{A}) &= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}((\lambda \mathsf{X}.\mathbf{A})\mathsf{X}) \\
&= \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\lambda \mathsf{X}.\mathbf{A})@\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/\mathsf{X}]}(\mathsf{X}) \\
&= \mathcal{E}_\varphi(\lambda \mathsf{X}.\mathbf{A})@\mathcal{E}_\varphi(\mathbf{B}) \\
&= \mathcal{E}_\varphi((\lambda \mathsf{X}.\mathbf{A})\mathbf{B}) \\
&= \mathcal{E}_\varphi([\mathbf{B}/\mathsf{X}]\mathbf{A}).
\end{aligned}$$

Weaker Notions of Functionality

We will consider two weaker notions of functionality. These forms are often discussed in the literature (cf. [HindleySeldin86]).

Weaker Notions of Functionality

We will consider two weaker notions of functionality. These forms are often discussed in the literature (cf. [HindleySeldin86]).

- η-functionality simply means the evaluation respects η-conversion.

Weaker Notions of Functionality

We will consider two weaker notions of functionality. These forms are often discussed in the literature (cf. [HindleySeldin86]).

- η-functionality simply means the evaluation respects η-conversion.
- ξ-functionality means we have functionality (only) with respect to λ-abstractions.

Def.: η-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation.

Def.: η-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation.
We say $\mathcal{J}$ is η-functional if

Def.: η-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation.
We say $\mathcal{J}$ is η-functional if

$$\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{A}{\downarrow_{\beta\eta}})$$

for any type α, formula $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$, and assignment φ.

Def.: ξ-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation.

Def.: ξ-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation. We say $\mathcal{J}$ is ξ-functional if

Def.: ξ-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation. We say $\mathcal{J}$ is ξ-functional if for all $\alpha, \beta \in \mathcal{T}$, $\mathbf{M}, \mathbf{N} \in \mathit{wff}_\beta(\Sigma)$, assignments φ, and variables X_α,

Def.: ξ-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation. We say $\mathcal{J}$ is ξ-functional if for all $\alpha, \beta \in \mathcal{T}$, $\mathbf{M}, \mathbf{N} \in \mathit{wff}_\beta(\Sigma)$, assignments φ, and variables X_α,

$$\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha . \mathbf{M}_\beta) = \mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha . \mathbf{N}_\beta)$$

Def.: ξ-Functional

Let $\mathcal{J} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation. We say $\mathcal{J}$ is ξ-functional if for all $\alpha, \beta \in \mathcal{T}$, $\mathbf{M}, \mathbf{N} \in wff_\beta(\Sigma)$, assignments φ, and variables X_α,

$$\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha . \mathbf{M}_\beta) = \mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha . \mathbf{N}_\beta)$$

whenever

$$\mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{M}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{N})$$

for every $\mathsf{a} \in \mathcal{D}_\alpha$.

Lemma: Functionality and η

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-evaluation.

Lemma: Functionality and η

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-evaluation.

1. For any assignment φ into $\mathcal{J}$ and $\mathbf{F} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ where $\mathrm{X}_\alpha \notin \mathbf{Free}(\mathbf{F})$, we have

Lemma: Functionality and η

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-evaluation.

1. For any assignment φ into $\mathcal{J}$ and $\mathbf{F} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ where $\mathsf{X}_\alpha \notin \mathbf{Free}(\mathbf{F})$, we have

$$\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha \mathbf{.} \mathbf{F}\mathsf{X}) = \mathcal{E}_\varphi(\mathbf{F})$$

Lemma: Functionality and η

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-evaluation.

1. For any assignment φ into $\mathcal{J}$ and $\mathbf{F} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ where $\mathsf{X}_\alpha \notin \mathbf{Free}(\mathbf{F})$, we have

$$\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha \mathbf{.} \mathbf{F}\mathsf{X}) = \mathcal{E}_\varphi(\mathbf{F})$$

2. If a formula $\mathbf{A}$ η-reduces to $\mathbf{B}$ in one step, then for any assignment φ into $\mathcal{J}$, we have

Lemma: Functionality and η

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-evaluation.

1. For any assignment φ into $\mathcal{J}$ and $\mathbf{F} \in wff_{\alpha\to\beta}(\Sigma)$ where $\mathsf{X}_\alpha \notin \mathbf{Free}(\mathbf{F})$, we have

$$\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha . \mathbf{F}\mathsf{X}) = \mathcal{E}_\varphi(\mathbf{F})$$

2. If a formula $\mathbf{A}$ η-reduces to $\mathbf{B}$ in one step, then for any assignment φ into $\mathcal{J}$, we have

$$\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{B})$$

Lemma: Functionality and η

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-evaluation.

1. For any assignment φ into $\mathcal{J}$ and $\mathbf{F} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ where $\mathsf{X}_\alpha \notin \mathbf{Free}(\mathbf{F})$, we have

$$\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha\mathbf{.F}\mathsf{X}) = \mathcal{E}_\varphi(\mathbf{F})$$

2. If a formula $\mathbf{A}$ η-reduces to $\mathbf{B}$ in one step, then for any assignment φ into $\mathcal{J}$, we have

$$\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{B})$$

Proof: Exercise

Lemma: Functionality and η+ξ

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation.

Lemma: Functionality and η+ξ

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation. Then $\mathcal{J}$ is functional iff it is both η-functional and ξ-functional.

Lemma: Functionality and η+ξ

Let $\mathcal{J} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-evaluation. Then $\mathcal{J}$ is functional iff it is both η-functional and ξ-functional.

Proof: Exercise

Logical Constants in Signature

Let $\Sigma := (\mathcal{V}, \mathcal{C})$ be a signature.

Logical Constants in Signature

Let $\Sigma := (\mathcal{V}, \mathcal{C})$ be a signature.

The following logical constants may or may not be in the set $\mathcal{C}$ of constants:

Logical Constants in Signature

Let $\Sigma := (\mathcal{V}, \mathcal{C})$ be a signature.

The following logical constants may or may not be in the set $\mathcal{C}$ of constants:

$$\top_o,\ \bot_o,\ \neg_{oo},\ \vee_{ooo},\ \wedge_{ooo},\ \supset_{ooo},\ \Leftrightarrow_{ooo}$$

Logical Constants in Signature

Let $\Sigma := (\mathcal{V}, \mathcal{C})$ be a signature.

The following logical constants may or may not be in the set $\mathcal{C}$ of constants:

$$\top_{o},\ \bot_{o},\ \neg_{oo},\ \vee_{ooo},\ \wedge_{ooo},\ \supset_{ooo},\ \Leftrightarrow_{ooo}$$

$$\Pi^{\alpha}_{o(o\alpha)} (\Pi^{\alpha} F_{o\alpha} \sim \forall x_{\alpha} F x),\ \Sigma^{\alpha}_{o(o\alpha)} (\Sigma^{\alpha} F_{o\alpha} \sim \exists x_{\alpha} F x)$$

Logical Constants in Signature

Let $\Sigma := (\mathcal{V}, \mathcal{C})$ be a signature.

The following logical constants may or may not be in the set $\mathcal{C}$ of constants:

$$\top_{o},\ \bot_{o},\ \neg_{oo},\ \vee_{ooo},\ \wedge_{ooo},\ \supset_{ooo},\ \Leftrightarrow_{ooo}$$

$$\Pi^{\alpha}_{o(o\alpha)}(\Pi^{\alpha}F_{o\alpha}\ \sim\ \forall x_{\alpha}Fx),\ \Sigma^{\alpha}_{o(o\alpha)}(\Sigma^{\alpha}F_{o\alpha}\ \sim\ \exists x_{\alpha}Fx)$$

$$=^{\alpha}_{o\alpha\alpha}$$

Logical Constants in Signature

Let $\Sigma := (\mathcal{V}, \mathcal{C})$ be a signature.

The following logical constants may or may not be in the set $\mathcal{C}$ of constants:

$$\top_o, \ \bot_o, \ \neg_{oo}, \ \vee_{ooo}, \ \wedge_{ooo}, \ \supset_{ooo}, \ \Leftrightarrow_{ooo}$$

$$\Pi^\alpha_{o(o\alpha)} (\Pi^\alpha \mathsf{F}_{o\alpha} \ \sim \ \forall \mathsf{x}_\alpha \mathsf{F} \mathsf{x}), \ \Sigma^\alpha_{o(o\alpha)} (\Sigma^\alpha \mathsf{F}_{o\alpha} \ \sim \ \exists \mathsf{x}_\alpha \mathsf{F} \mathsf{x})$$

$$=^\alpha_{o\alpha\alpha}$$

for all $\alpha \in \mathcal{T}$

Once More: Cantor's Theorem

For any set A,

$$|A| < |\mathcal{P}(A)|$$

Once More: Cantor's Theorem

For any set A,

$$|A| < |\mathcal{P}(A)|$$

i.e., $\neg\exists g : A \longrightarrow \mathcal{P}(A)$ with g surjective.

Once More: Cantor's Theorem

Assume the set A is associated with ι.

Once More: Cantor's Theorem

Assume the set A is associated with ι. Then $\mathcal{P}(\mathrm{A})$ has type $o\iota$, i.e. the type of "sets" (or characteristic functions) over ι.

Once More: Cantor's Theorem

Assume the set A is associated with ι. Then $\mathcal{P}(A)$ has type $o\iota$, i.e. the type of "sets" (or characteristic functions) over ι.

$$D_{o\iota}$$

Once More: Cantor's Theorem

Assume the set A is associated with ι. Then $\mathcal{P}(A)$ has type $o\iota$, i.e. the type of "sets" (or characteristic functions) over ι.

$$D_{o\iota} = D_o^{D_\iota}$$

Once More: Cantor's Theorem

Assume the set A is associated with ι. Then $\mathcal{P}(\mathsf{A})$ has type $o\iota$, i.e. the type of "sets" (or characteristic functions) over ι.

$$\begin{aligned} \mathsf{D}_{o\iota} &= \mathsf{D}_o^{\mathsf{D}_\iota} \\ &= \{\bot, \top\}^{\mathsf{D}_\iota} \end{aligned}$$

Once More: Cantor's Theorem

Assume the set A is associated with ι. Then $\mathcal{P}(A)$ has type $o\iota$, i.e. the type of "sets" (or characteristic functions) over ι.

$$
\begin{aligned}
D_{o\iota} &= D_o^{D_\iota} \\
&= \{\bot, \top\}^{D_\iota} \\
&= \{f |\ f : D_\iota \to \{\bot, \top\}\ \}
\end{aligned}
$$

Once More: Cantor's Theorem

Assume the set A is associated with ι. Then $\mathcal{P}(A)$ has type $o\iota$, i.e. the type of "sets" (or characteristic functions) over ι.

$$
\begin{aligned}
D_{o\iota} &= D_o^{D_\iota} \\
&= \{\bot, \top\}^{D_\iota} \\
&= \{f \mid f : D_\iota \longrightarrow \{\bot, \top\}\ \} \\
&\cong \{\ X \mid X \subseteq D_\iota\ \}
\end{aligned}
$$

Once More: Cantor's Theorem

Assume the set A is associated with ι. Then $\mathcal{P}(A)$ has type $o\iota$, i.e. the type of "sets" (or characteristic functions) over ι.

$$\begin{aligned}
D_{o\iota} &= D_o^{D_\iota} \\
&= \{\bot, \top\}^{D_\iota} \\
&= \{f \mid f : D_\iota \longrightarrow \{\bot, \top\}\ \} \\
&\cong \{\ X \mid X \subseteq D_\iota\ \} \\
&= \mathcal{P}(D_\iota)
\end{aligned}$$

Once More: Cantor's Theorem

We can now formulate Cantor's Theorem with typed terms (as seen before):

Once More: Cantor's Theorem

We can now formulate Cantor's Theorem with typed terms (as seen before):

$$\neg\exists g_{o\iota\iota}\forall f_{o\iota}\exists x_{\iota} : gx = f$$

Once More: Cantor's Theorem

We can now formulate Cantor's Theorem with typed terms (as seen before):

$$\neg\exists g_{o\iota\iota}\forall f_{o\iota}\exists x_{\iota} : gx = f$$

which is shorthand for:

Once More: Cantor's Theorem

We can now formulate Cantor's Theorem with typed terms (as seen before):

$$\neg\exists g_{o\iota\iota}\forall f_{o\iota}\exists x_\iota : gx = f$$

which is shorthand for:

$$\neg_{oo}\Sigma^{o\iota\iota}_{o(o(o\iota\iota))}\left(\lambda g_{o\iota\iota}.\Pi^{o\iota}_{o(o(o\iota))}\left(\lambda f_{o\iota}.\Sigma^{\iota}_{o(o\iota)}\left(\lambda x_\iota. =^{o\iota}_{o(o\iota)(o\iota)} (gx)\, f\right)\right)\right)$$

Once More: Cantor's Theorem

We can now formulate Cantor's Theorem with typed terms (as seen before):

$$\neg\exists g_{o\iota\iota}\forall f_{o\iota}\exists x_\iota : gx = f$$

which is shorthand for:

$$\neg_{oo}\Sigma^{o\iota\iota}_{o(o(o\iota\iota))}\left(\lambda g_{o\iota\iota}.\Pi^{o\iota}_{o(o(o\iota))}\left(\lambda f_{o\iota}.\Sigma^{\iota}_{o(o\iota)}\left(\lambda x_\iota. =^{o\iota}_{o(o\iota)(o\iota)}(gx)\, f\right)\right)\right)$$

Note: for this term to be in the set $cwff_\alpha(\Sigma)$, the constants $\neg_{oo}$, $\Sigma^{o\iota\iota}_{o(o(o\iota\iota))}$, $\Pi^{o\iota}_{o(o(o\iota))}$, Σ^{ι} and $=^{o\iota}$ have to be in the set $\mathcal{C}$.

Once More: Cantor's Theorem

Proof:

Once More: Cantor's Theorem

Proof: Assume such a function g exists.

Once More: Cantor's Theorem

Proof: Assume such a function g exists.

Let $f = \{x \mid x \notin gx\}$ that is $f = (\lambda x_\iota . \neg gxx)$.

Once More: Cantor's Theorem

Proof: Assume such a function g exists.

Let $f = \{x \mid x \notin gx\}$ that is $f = (\lambda x_\iota . \neg gxx)$.

g is surjective,

Once More: Cantor's Theorem

Proof: Assume such a function g exists.
Let $\mathsf{f} = \{\mathsf{x} \mid \mathsf{x} \notin \mathsf{gx}\}$ that is $\mathsf{f} = (\lambda \mathsf{x}_\iota . \neg \mathsf{gxx})$.
g is surjective, hence

$$(\exists \mathsf{y}_\iota : \mathsf{gy} = [\lambda \mathsf{x}. \neg \mathsf{gxx}])$$

Once More: Cantor's Theorem

Proof: Assume such a function g exists.

Let $f = \{x \mid x \notin gx\}$ that is $f = (\lambda x_\iota . \neg gxx)$.

g is surjective, hence

$$(\exists y_\iota : gy = [\lambda x . \neg gxx])$$

hence

$$(gyy \Leftrightarrow \neg gyy)$$

Once More: Cantor's Theorem

Proof: Assume such a function g exists.
Let $f = \{x \mid x \notin gx\}$ that is $f = (\lambda x_\iota . \neg gxx)$.
g is surjective, hence

$$(\exists y_\iota : gy = [\lambda x . \neg gxx])$$

hence

$$(gyy \Leftrightarrow \neg gyy)$$

Contradiction!

Once More: Cantor's Theorem

Proof: Assume such a function g exists.
Let $f = \{x \mid x \notin gx\}$ that is $f = (\lambda x_\iota.\neg gxx)$.
g is surjective, hence

$$(\exists y_\iota : gy = [\lambda x.\neg gxx])$$

hence

$$(gyy \Leftrightarrow \neg gyy)$$

Contradiction!

Note that the proof uses $\neg$.

Semantics: Σ-Models

Def.: Properties of Logical Constants

Let $(\mathrm{D}, @)$ be an applicative structure and let $\mathtt{v} : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$).

Def.: Properties of Logical Constants

Let $(\mathsf{D}, @)$ be an applicative structure and let $\mathsf{v} : \mathcal{D}_{\mathsf{o}} \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $\mathsf{a} \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_{\mathsf{c}}(()\mathsf{a})$ with respect to v:

Def.: Properties of Logical Constants

Let $(\mathrm{D}, @)$ be an applicative structure and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $\mathsf{a} \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_{\mathsf{c}}(()\mathsf{a})$ with respect to v:

c		

Def.: Properties of Logical Constants

Let $(\mathcal{D}, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c((\,)a)$ with respect to v:

c	β	

Def.: Properties of Logical Constants

Let $(\mathsf{D}, @)$ be an applicative structure and let $\mathsf{v} : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $\mathsf{a} \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_\mathsf{c}((\,)\mathsf{a})$ with respect to v:

c	β	$\mathfrak{L}_\mathsf{c}((\,)\mathsf{a})$ holds when

Def.: Properties of Logical Constants

Let $(\mathcal{D}, @)$ be an applicative structure and let $v: \mathcal{D}_o \rightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$		

Def.: Properties of Logical Constants

Let $(\mathsf{D}, @)$ be an applicative structure and let $\mathsf{v} : \mathcal{D}_{\mathsf{o}} \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $\mathsf{a} \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_{\mathsf{c}}(()\mathsf{a})$ with respect to v:

c	β	$\mathfrak{L}_{\mathsf{c}}(()\mathsf{a})$ holds when
$\top$	o	

Def.: Properties of Logical Constants

Let $(\mathsf{D}, @)$ be an applicative structure and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $\mathsf{a} \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_\mathsf{c}(()\mathsf{a})$ with respect to v:

c	β	$\mathfrak{L}_\mathsf{c}(()\mathsf{a})$ holds when
$\top$	o	$\mathsf{v}(\mathsf{a}) = \mathtt{T}$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$

Def.: Properties of Logical Constants

Let $(\mathsf{D}, @)$ be an applicative structure and let $\mathsf{v} : \mathcal{D}_{\mathsf{o}} \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $\mathsf{a} \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_{\mathsf{c}}(()\mathsf{a})$ with respect to v:

c	β	$\mathfrak{L}_{\mathsf{c}}(()\mathsf{a})$ holds when
$\top$	o	$\mathsf{v}(\mathsf{a}) = \mathtt{T}$
$\bot$	o	$\mathsf{v}(\mathsf{a}) = \mathtt{F}$
$\neg$		

Def.: Properties of Logical Constants

Let $(\mathcal{D}, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$
$\neg$	oo	

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$
$\neg$	oo	$v(a@b) = \mathtt{T}$ iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \rightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$
$\neg$	oo	$v(a@b) = \mathtt{T}$ iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$
$\neg$	oo	$v(a@b) = \mathtt{T}$ iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$ iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$
$\neg$	oo	$v(a@b) = \mathtt{T}$ iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$ iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$
$\neg$	oo	$v(a@b) = \mathtt{T}$ iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$ iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when
$\top$	o	$v(a) = \mathtt{T}$
$\bot$	o	$v(a) = \mathtt{F}$
$\neg$	oo	$v(a@b) = \mathtt{T}$ iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$ iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$ iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\perp$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$			

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$			

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$	$v(a@b@c) = \mathtt{T}$	iff $b = c$ $\forall b, c \in \mathcal{D}_o$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$	$v(a@b@c) = \mathtt{T}$	iff $b = c$ $\forall b, c \in \mathcal{D}_o$
Π^α			

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$	$v(a@b@c) = \mathtt{T}$	iff $b = c$ $\forall b, c \in \mathcal{D}_o$
Π^α	$o(o\alpha)$		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$	$v(a@b@c) = \mathtt{T}$	iff $b = c$ $\forall b, c \in \mathcal{D}_o$
Π^α	$o(o\alpha)$	$v(a@f) = \mathtt{T}$	iff $\forall b \in \mathcal{D}_\alpha : v(f@b) = \mathtt{T}$ $\forall f \in \mathcal{D}_{o\alpha}$

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$	$v(a@b@c) = \mathtt{T}$	iff $b = c$ $\forall b, c \in \mathcal{D}_o$
Π^α	$o(o\alpha)$	$v(a@f) = \mathtt{T}$	iff $\forall b \in \mathcal{D}_\alpha :\ v(f@b) = \mathtt{T}$ $\forall f \in \mathcal{D}_{o\alpha}$
Σ^α			

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \rightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$	$v(a@b@c) = \mathtt{T}$	iff $b = c$ $\forall b, c \in \mathcal{D}_o$
Π^α	$o(o\alpha)$	$v(a@f) = \mathtt{T}$	iff $\forall b \in \mathcal{D}_\alpha : v(f@b) = \mathtt{T}$ $\forall f \in \mathcal{D}_{o\alpha}$
Σ^α	$o(o\alpha)$		

Def.: Properties of Logical Constants

Let $(D, @)$ be an applicative structure and let $v : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a function (for given $\mathtt{T} \neq \mathtt{F}$). For each logical constant c_β and for $a \in \mathcal{D}_\beta$, we define the proposition $\mathfrak{L}_c(()a)$ with respect to v:

c	β	$\mathfrak{L}_c(()a)$ holds when	
$\top$	o	$v(a) = \mathtt{T}$	
$\bot$	o	$v(a) = \mathtt{F}$	
$\neg$	oo	$v(a@b) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ $\forall b \in \mathcal{D}_o$
$\vee$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\wedge$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{T}$ and $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\supset$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = \mathtt{F}$ or $v(c) = \mathtt{T}$ $\forall b, c \in \mathcal{D}_o$
$\Leftrightarrow$	ooo	$v(a@b@c) = \mathtt{T}$	iff $v(b) = v(c)$ $\forall b, c \in \mathcal{D}_o$
$=^\alpha$	$o\alpha\alpha$	$v(a@b@c) = \mathtt{T}$	iff $b = c$ $\forall b, c \in \mathcal{D}_o$
Π^α	$o(o\alpha)$	$v(a@f) = \mathtt{T}$	iff $\forall b \in \mathcal{D}_\alpha : v(f@b) = \mathtt{T}$ $\forall f \in \mathcal{D}_{o\alpha}$
Σ^α	$o(o\alpha)$	$v(a@f) = \mathtt{T}$	iff $\exists b \in \mathcal{D}_\alpha : v(f@b) = \mathtt{T}$ $\forall f \in \mathcal{D}_{o\alpha}$

Def.: Σ-Valuation

Let $\mathcal{J} := (\mathrm{D}, @, \mathcal{E})$ be a Σ-evaluation and $\mathsf{v} : \mathcal{D}_{\mathsf{o}} \longrightarrow \{\mathtt{T}, \mathtt{F}\}$.

Def.: Σ-Valuation

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and $\mathsf{v} : \mathcal{D}_o \rightarrow \{\mathtt{T}, \mathtt{F}\}$. We say v is a Σ-valuation w.r.t $\mathcal{J}$ if

Def.: Σ-Valuation

Let $\mathcal{J} := (\mathrm{D}, @, \mathcal{E})$ be a Σ-evaluation and $\mathsf{v} : \mathcal{D}_{\mathsf{o}} \longrightarrow \{\mathtt{T}, \mathtt{F}\}$. We say v is a Σ-valuation w.r.t $\mathcal{J}$ if $\mathfrak{L}_{\mathsf{c}}((\mathcal{E}(\mathsf{c})))$ holds w.r.t v for each logical constant $\mathsf{c} \in \Sigma$.

Def.: Σ-Model

Let $\mathcal{J} := (\mathrm{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \rightarrow \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

Def.: Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

Def.: Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

If $(\mathsf{D}, @, \mathcal{E})$ is functional (full, standard), we say $\mathcal{M}$ is functional (full, standard).

Def.: Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

If $(\mathsf{D}, @, \mathcal{E})$ is functional (full, standard), we say $\mathcal{M}$ is functional (full, standard).

If $(\mathsf{D}, @, \mathcal{E})$ is η-functional, we say $\mathcal{M}$ is η-functional.

Def.: Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

If $(\mathsf{D}, @, \mathcal{E})$ is functional (full, standard), we say $\mathcal{M}$ is functional (full, standard).

If $(\mathsf{D}, @, \mathcal{E})$ is η-functional, we say $\mathcal{M}$ is η-functional.

If $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional, we say $\mathcal{M}$ is ξ-functional.

Some Conventions: Equality

Some important conventions:

- $=$ denotes primitive equality

Some Conventions: Equality

Some important conventions:

- $=$ denotes primitive equality
- $\doteq$ denotes Leibniz equality: $\mathbf{A}_\alpha \doteq^\alpha \mathbf{B}_\alpha := \forall \mathrm{P}_{o\alpha}.(\mathrm{P}\mathbf{A}) \Rightarrow (\mathrm{P}\mathbf{B})$

Some Conventions: Equality

Some important conventions:

- $=$ denotes primitive equality
- $\doteq$ denotes Leibniz equality: $\mathbf{A}_\alpha \doteq^\alpha \mathbf{B}_\alpha := \forall \mathrm{P}_{o\alpha} (\mathrm{P}\mathbf{A}) \Rightarrow (\mathrm{P}\mathbf{B})$
- $\ddot{=}$ . . . other definition of equality (e.g., see [Andrews02])

Some Conventions: Equality

Some important conventions:

- $=$ denotes primitive equality
- $\doteq$ denotes Leibniz equality: $\mathbf{A}_\alpha \doteq^\alpha \mathbf{B}_\alpha := \forall \mathrm{P}_{o\alpha}.(\mathrm{P}\mathbf{A}) \Rightarrow (\mathrm{P}\mathbf{B})$
- $\ddot{=}$ ... other definition of equality (e.g., see [Andrews02])

We use $\stackrel{*}{=}$ in the following to refer to any of the above

Def.: Properties $\mathfrak{f}, \mathfrak{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

Def.: Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)

Def.: Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)

Def.: Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- f if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)

Def.: Properties $\mathfrak{f}, \mathfrak{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- $\mathfrak{f}$ if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)
- $\mathfrak{b}$ if v is injective.

Def.: Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- f if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)
- b if v is injective.

  Note: In the [JSC04]-paper, b is defined as $\mathsf{D}_o = \{\mathtt{T}, \mathtt{F}\}$, but here we are using the injectivity criterion, because we are varying the signature. If the signature is too sparse, we could have a D_o with two elements which both valuate via v to $\mathtt{T}$. Another ill case would be D_o with just one element.

Def.: Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- f if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)
- b if v is injective.
- q if for all $\alpha \in \mathcal{T}$ there is some $\mathsf{q} \in \mathsf{D}_{\mathsf{o}\alpha\alpha}$ such that $\mathfrak{L}_{=^\alpha}(\mathsf{q})$.

Def.: Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- f if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)
- b if v is injective.
- q if for all $\alpha \in \mathcal{T}$ there is some $\mathsf{q} \in \mathsf{D}_{o\alpha\alpha}$ such that $\mathfrak{L}_{=^\alpha}(\mathsf{q})$.

  Note: This basically says that for each type α the identity relation over α is already present in the model. If we require $=_{o\alpha\alpha} \in \mathcal{C}$ with $\mathfrak{L}_{=^\alpha}(\mathcal{E}_\varphi(=_{o\alpha\alpha}))$, then this property is automatically ensured, but not for weaker signatures. See [Andrew71] for a detailed discussion of property q. Andrews constructs a Henkin model where Leibniz equality $\doteq$ does not evaluate to the intended identity relation. This is resolved by property q.

Lemma: Surjective v

Let $\mathcal{C}$ be a signature and $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model.

Lemma: Surjective v

Let $\mathcal{C}$ be a signature and $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model.
If $\mathtt{T}, \mathtt{F} \in \mathcal{C}$ or $\neg \in \mathcal{C}$ then v is surjective.

Lemma: Surjective v

Let $\mathcal{C}$ be a signature and $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model.
If $\mathtt{T}, \mathtt{F} \in \mathcal{C}$ or $\neg \in \mathcal{C}$ then v is surjective.

Proof: Exercise.

Thm.: Property $\mathfrak{b}$

Let $\mathcal{C}$ be a signature and $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model.

Thm.: Property $\mathfrak{b}$

Let $\mathcal{C}$ be a signature and $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model.
Suppose $\mathtt{T}, \mathtt{F} \in \mathcal{C}$ or $\neg \in \mathcal{C}$.

Thm.: Property $\mathfrak{b}$

Let $\mathcal{C}$ be a signature and $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model.
Suppose $\mathtt{T}, \mathtt{F} \in \mathcal{C}$ or $\neg \in \mathcal{C}$.
Then $\mathcal{M}$ satisfies property $\mathfrak{b}$ iff $|\mathsf{D}_o| = 2$.

Thm.: Property $\mathfrak{b}$

Let $\mathcal{C}$ be a signature and $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model.
Suppose $\mathtt{T}, \mathtt{F} \in \mathcal{C}$ or $\neg \in \mathcal{C}$.
Then $\mathcal{M}$ satisfies property $\mathfrak{b}$ iff $|\mathsf{D}_o| = 2$.

Proof: Exercise.

Semantics: HOL-CUBE

Def. (Reminder): Σ-Model

Let $\mathcal{J} := (\mathrm{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

Def. (Reminder): Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

Def. (Reminder): Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

If $(\mathsf{D}, @, \mathcal{E})$ is functional (full, standard), we say $\mathcal{M}$ is functional (full, standard).

Def. (Reminder): Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

If $(\mathsf{D}, @, \mathcal{E})$ is functional (full, standard), we say $\mathcal{M}$ is functional (full, standard).

If $(\mathsf{D}, @, \mathcal{E})$ is η-functional, we say $\mathcal{M}$ is η-functional.

Def. (Reminder): Σ-Model

Let $\mathcal{J} := (\mathsf{D}, @, \mathcal{E})$ be a Σ-evaluation and let $\mathsf{v} : \mathcal{D}_o \rightarrow \{\mathtt{T}, \mathtt{F}\}$ be a Σ-valuation w.r.t $\mathcal{J}$

We say $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ is a Σ-model.

If $(\mathsf{D}, @, \mathcal{E})$ is functional (full, standard), we say $\mathcal{M}$ is functional (full, standard).

If $(\mathsf{D}, @, \mathcal{E})$ is η-functional, we say $\mathcal{M}$ is η-functional.

If $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional, we say $\mathcal{M}$ is ξ-functional.

Def. (Reminder): Properties $\mathfrak{f}, \mathfrak{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

Def. (Reminder): Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)

Def. (Reminder): Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)

Def. (Reminder): Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- f if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)

Def. (Reminder): Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- f if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)
- b if v is injective.

Def. (Reminder): Properties $\mathsf{f}, \mathsf{b}, \eta, \xi$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a $\mathcal{C}$-model. We say, M has property

- η if M is η-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is η-functional)
- ξ if M is ξ-functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is ξ-functional)
- f if M is functional (respectively $(\mathsf{D}, @, \mathcal{E})$ is functional)
- b if v is injective.
- q if for all $\alpha \in \mathcal{T}$ there is some $\mathsf{q} \in \mathsf{D}_{o\alpha\alpha}$ such that $\mathfrak{L}_{=^\alpha}(\mathsf{q})$.

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$.

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathsf{f}, \mathsf{b}$.

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathsf{f}, \mathsf{b}$. Thus we obtain

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathsf{f}, \mathsf{b}$. Thus we obtain

- $\mathfrak{M}_{\beta\eta}(\Sigma)$

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathsf{f}, \mathsf{b}$. Thus we obtain

- $\mathfrak{M}_{\beta\eta}(\Sigma)$
- $\mathfrak{M}_{\beta\xi}(\Sigma)$

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathsf{f}, \mathsf{b}$. Thus we obtain

- $\mathfrak{M}_{\beta\eta}(\Sigma)$
- $\mathfrak{M}_{\beta\xi}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma)$

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathfrak{f}, \mathfrak{b}$. Thus we obtain

- $\mathfrak{M}_{\beta\eta}(\Sigma)$
- $\mathfrak{M}_{\beta\xi}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{b}}(\Sigma)$

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathfrak{f}, \mathfrak{b}$. Thus we obtain

- $\mathfrak{M}_{\beta\eta}(\Sigma)$
- $\mathfrak{M}_{\beta\xi}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{b}}(\Sigma)$
- $\mathfrak{M}_{\beta\eta\mathfrak{b}}(\Sigma)$

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathsf{f}, \mathsf{b}$. Thus we obtain

- $\mathfrak{M}_{\beta\eta}(\Sigma)$
- $\mathfrak{M}_{\beta\xi}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{b}}(\Sigma)$
- $\mathfrak{M}_{\beta\eta\mathfrak{b}}(\Sigma)$
- $\mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma)$

Def. (Reminder): Different Model Classes

We denote the class of $\mathcal{C}$-models by $\mathfrak{M}_\beta(\Sigma)$. We obtain a hierarchy of subclasses of $\mathfrak{M}_\beta(\Sigma)$ by adding the properties $\xi, \eta, \mathfrak{f}, \mathfrak{b}$. Thus we obtain

- $\mathfrak{M}_{\beta\eta}(\Sigma)$
- $\mathfrak{M}_{\beta\xi}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{b}}(\Sigma)$
- $\mathfrak{M}_{\beta\eta\mathfrak{b}}(\Sigma)$
- $\mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma)$
- $\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma)$

Def.: Satisfies, models, and $\models$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a Σ-model and let φ be an assignment into $\mathcal{M}$.

Def.: Satisfies, models, and $\models$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a Σ-model and let φ be an assignment into $\mathcal{M}$.

We say φ satisfies a formula $\mathbf{A} \in \mathit{wff}_{o}(\Sigma)$ in $\mathcal{M}$ (we write $\mathcal{M} \models_{\varphi} \mathbf{A}$) if $\upsilon(\mathcal{E}_{\varphi}(\mathbf{A})) = \mathtt{T}$.

Def.: Satisfies, models, and $\models$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a Σ-model and let φ be an assignment into $\mathcal{M}$.

We say φ satisfies a formula $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ in $\mathcal{M}$ (we write $\mathcal{M} \models_\varphi \mathbf{A}$) if $\upsilon(\mathcal{E}_\varphi(\mathbf{A})) = \mathtt{T}$.

We say that $\mathbf{A}$ is valid in $\mathcal{M}$ (and write $\mathcal{M} \models \mathbf{A}$) if $\mathcal{M} \models_\varphi \mathbf{A}$ for all assignments φ.

Def.: Satisfies, models, and $\models$

Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a Σ-model and let φ be an assignment into $\mathcal{M}$.

We say φ satisfies a formula $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ in $\mathcal{M}$ (we write $\mathcal{M} \models_\varphi \mathbf{A}$) if $\upsilon(\mathcal{E}_\varphi(\mathbf{A})) = \mathtt{T}$.

We say that $\mathbf{A}$ is valid in $\mathcal{M}$ (and write $\mathcal{M} \models \mathbf{A}$) if $\mathcal{M} \models_\varphi \mathbf{A}$ for all assignments φ. When $\mathbf{A} \in \mathit{cwff}_o(\Sigma)$, we drop the reference to the assignment and use the notation $\mathcal{M} \models \mathbf{A}$.

Def.: Satisfies, models, and $\models$

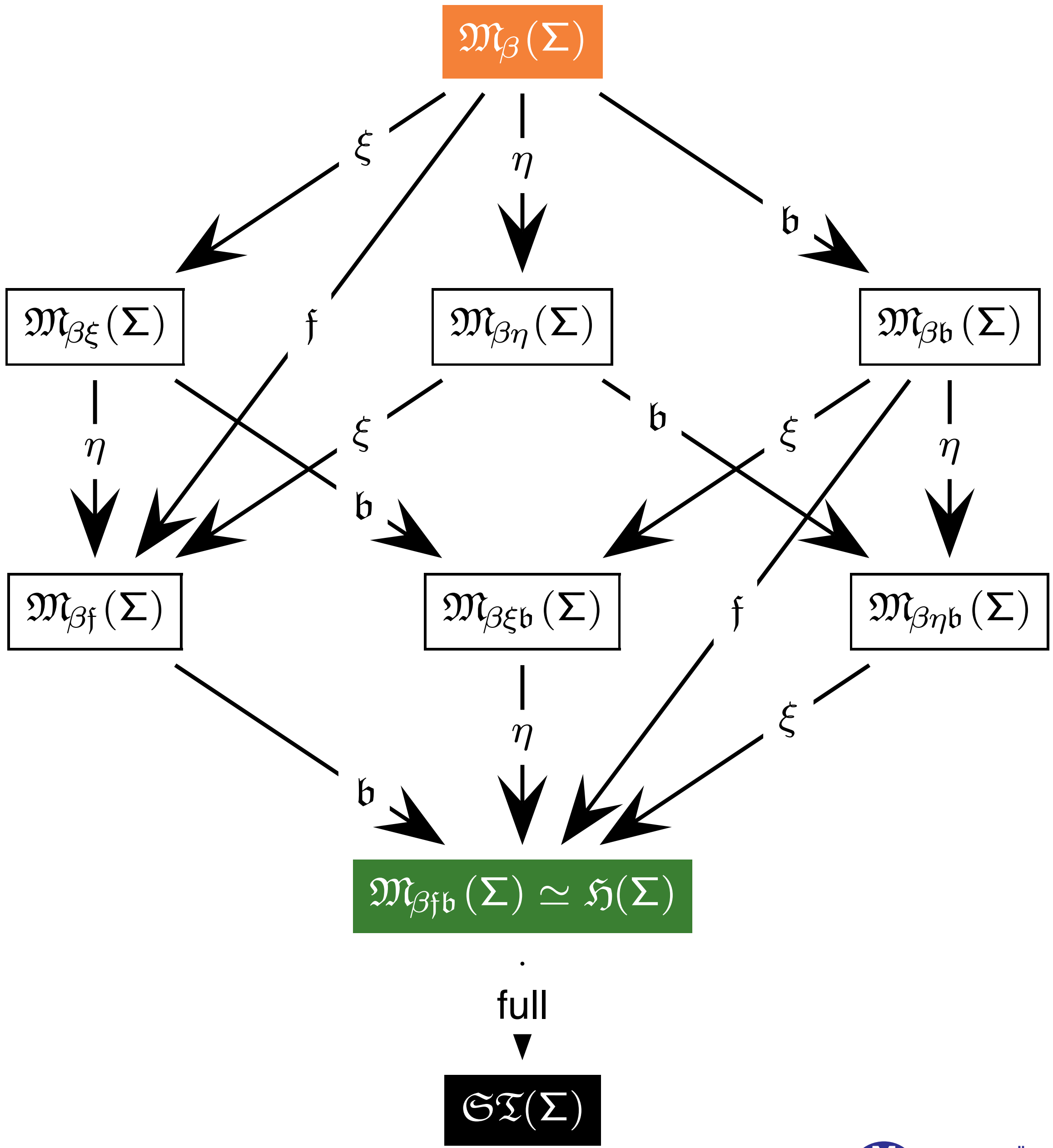
Let $\mathcal{M} = (\mathsf{D}, @, \mathcal{E}, \mathsf{v})$ be a Σ-model and let φ be an assignment into $\mathcal{M}$.

We say φ satisfies a formula $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ in $\mathcal{M}$ (we write $\mathcal{M} \models_\varphi \mathbf{A}$) if $v(\mathcal{E}_\varphi(\mathbf{A})) = \mathtt{T}$.

We say that $\mathbf{A}$ is valid in $\mathcal{M}$ (and write $\mathcal{M} \models \mathbf{A}$) if $\mathcal{M} \models_\varphi \mathbf{A}$ for all assignments φ. When $\mathbf{A} \in \mathit{cwff}_o(\Sigma)$, we drop the reference to the assignment and use the notation $\mathcal{M} \models \mathbf{A}$.

Finally, we say that $\mathcal{M}$ is a Σ-model for a set $\Phi \subseteq \mathit{cwff}_o(\Sigma)$ (we write $\mathcal{M} \models \Phi$) if $\mathcal{M} \models \mathbf{A}$ for all $\mathbf{A} \in \Phi$.

Semantics: HOL-CUBE



Landscape of HOL model classes

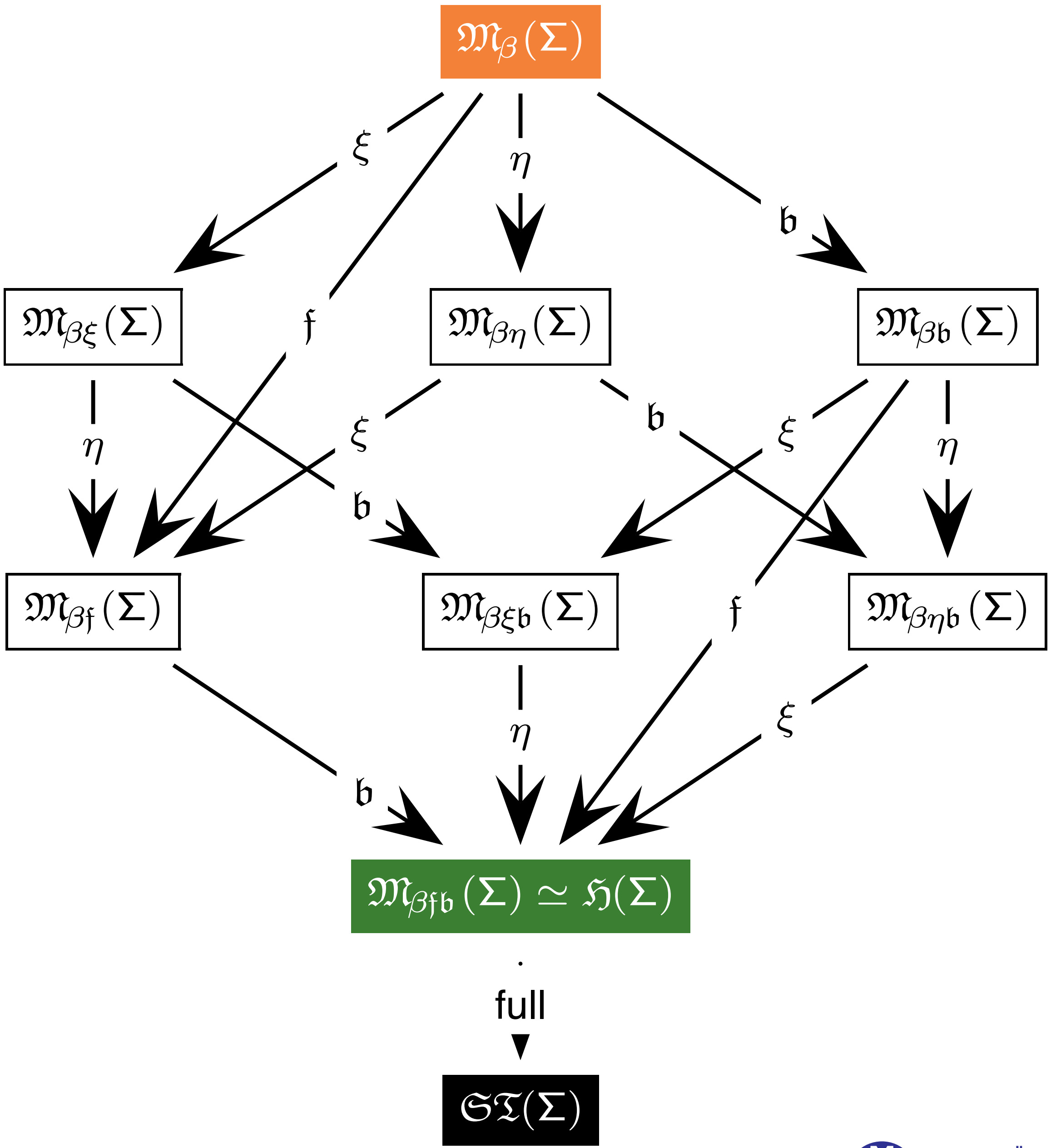
[Kohlhase-PhD-94]

[Benzmüller-PhD-99]

[Brown-PhD-04]

[Benzm.BrownKohlhase-JSL-04]

Semantics: HOL-CUBE



Landscape of HOL model classes

[Kohlhase-PhD-94]

[Benzmüller-PhD-99]

[Brown-PhD-04]

[Benzm.BrownKohlhase-JSL-04]

$\mathfrak{M}_{\beta}(\Sigma)$ model class for Σ-fragment of elementary type theory

Semantics: HOL-CUBE

$$\begin{tikzcd}
 & \mathfrak{M}_{\beta}(\Sigma) \arrow[dl, "\xi"'] \arrow[d, "\eta"] \arrow[dr, "\mathfrak{b}"] \arrow[ddl, "\mathfrak{f}"] & \\
\mathfrak{M}_{\beta\xi}(\Sigma) \arrow[d, "\eta"] \arrow[dr, "\mathfrak{b}"] & \mathfrak{M}_{\beta\eta}(\Sigma) \arrow[dl, "\xi"] \arrow[dr, "\mathfrak{b}"] & \mathfrak{M}_{\beta\mathfrak{b}}(\Sigma) \arrow[dl, "\xi"] \arrow[d, "\eta"] \arrow[ddl, "\mathfrak{f}"] \\
\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma) \arrow[dr, "\mathfrak{b}"] & \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma) \arrow[d, "\eta"] & \mathfrak{M}_{\beta\eta\mathfrak{b}}(\Sigma) \arrow[dl, "\xi"] \\
 & \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma) \simeq \mathfrak{H}(\Sigma) \arrow[d, dotted, "\text{full}"] & \\
 & \mathfrak{S}\mathfrak{T}(\Sigma) &
\end{tikzcd}$$

Landscape of HOL model classes

[Kohlhase-PhD-94]

[Benzmüller-PhD-99]

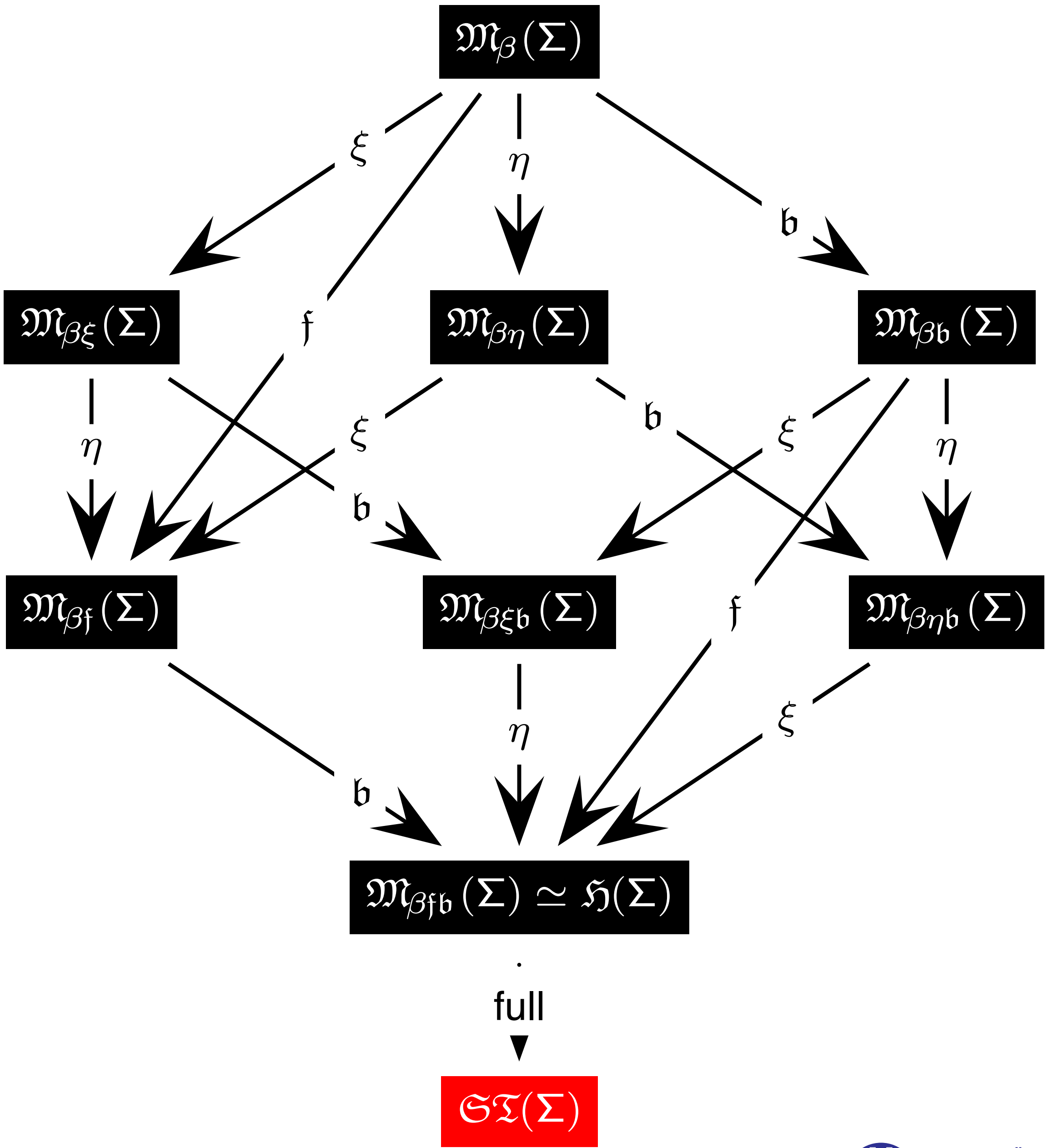
[Brown-PhD-04]

[Benzm.BrownKohlhase-JSL-04]

$\mathfrak{M}_{\beta}(\Sigma)$ model class for Σ-fragment of elementary type theory

$\mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma)$ model class for Σ-fragment of extensional type theory (Henkin models)

Semantics: HOL-CUBE

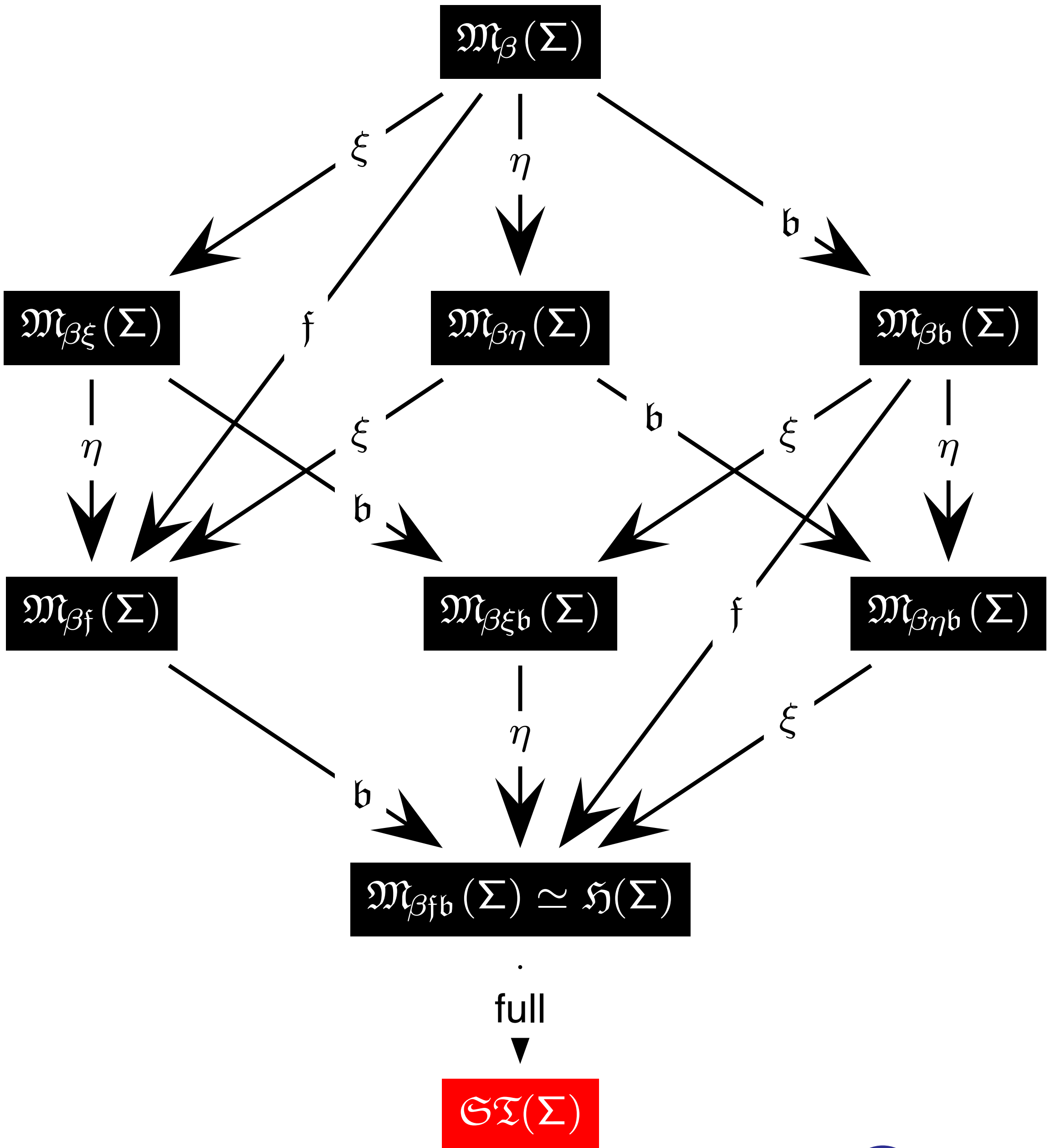


β: models support β-equality

$\mathfrak{q}$: models provide identity relations

$\forall\alpha : \mathsf{id} \in \mathcal{D}_{\alpha\to\alpha\to o}$

Semantics: HOL-CUBE



β: models support β-equality

$\mathfrak{q}$: models provide identity relations

$\forall\alpha : \mathsf{id} \in \mathcal{D}_{\alpha\to\alpha\to o}$

- [Andrews72]: without property $\mathfrak{q}$ Leibniz equality $\doteq$ not necessarily evaluates to identity relation even in Henkin semantics ($\mathfrak{H}(\Sigma)$)

Standard Models and Henkin Models

Leon Henkin generalized the class of admissible domains for functional types.

Standard Models and Henkin Models

Leon Henkin generalized the class of admissible domains for functional types.

Instead of requiring $\mathcal{D}_{\alpha\beta}$ (and thus in particular, $\mathcal{D}_{o\iota}$) to be the full set of functions (predicates), it is sufficient to require that $\mathcal{D}_{\alpha\beta}$ has enough members that any well-formed formula can be evaluated (in other words, the domains of function types are rich enough to satisfy comprehension).

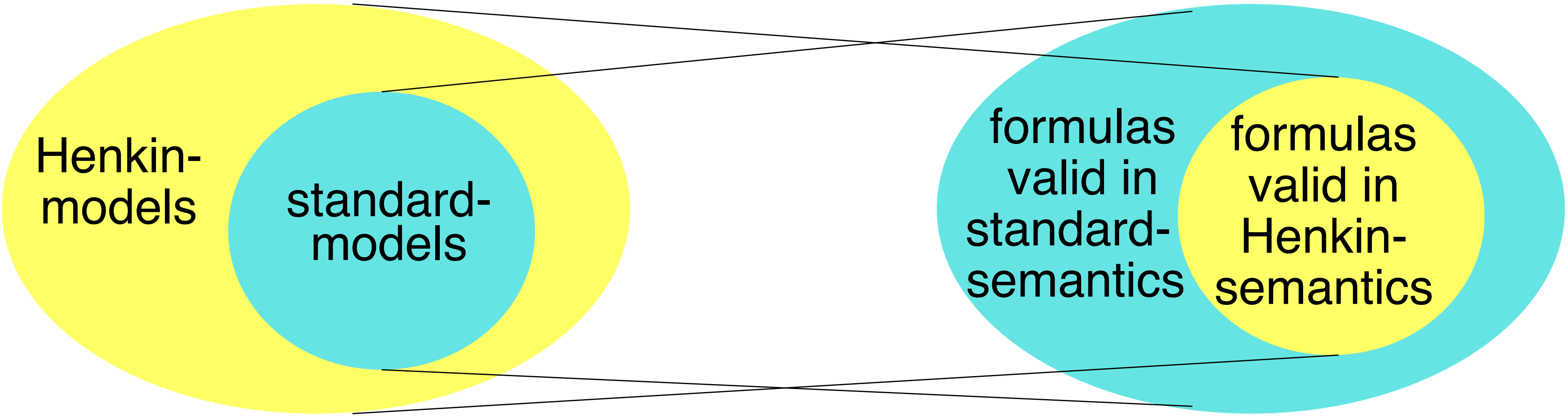
Standard Models and Henkin Models

Leon Henkin generalized the class of admissible domains for functional types.

Instead of requiring $\mathcal{D}_{\alpha\beta}$ (and thus in particular, $\mathcal{D}_{o\iota}$) to be the full set of functions (predicates), it is sufficient to require that $\mathcal{D}_{\alpha\beta}$ has enough members that any well-formed formula can be evaluated (in other words, the domains of function types are rich enough to satisfy comprehension).

Note that with this generalized notion of a model, there are fewer formulae that are valid in all models (intuitively, for any given formula there are more possibilities for counter-models).

Standard Models and Henkin Models



Standard Models and Henkin Models

The generalization to Henkin models restricts the set of valid formulae sufficiently so that all of them can be proven by a Hilbert-style calculus [Henkin50].

Standard Models and Henkin Models

The generalization to Henkin models restricts the set of valid formulae sufficiently so that all of them can be proven by a Hilbert-style calculus [Henkin50].

Of course our HOL-CUBE is not complete here; we can axiomatically require the existence of particular (classes of) functions, e.g., by assuming the description or choice operators.

Standard Models and Henkin Models

The generalization to Henkin models restricts the set of valid formulae sufficiently so that all of them can be proven by a Hilbert-style calculus [Henkin50].

Of course our HOL-CUBE is not complete here; we can axiomatically require the existence of particular (classes of) functions, e.g., by assuming the description or choice operators.

We will not pursue this here; for a detailed discussion of the semantic issues raised by the presence of these logical constants see [Andrews72].

Standard Models and Henkin Models

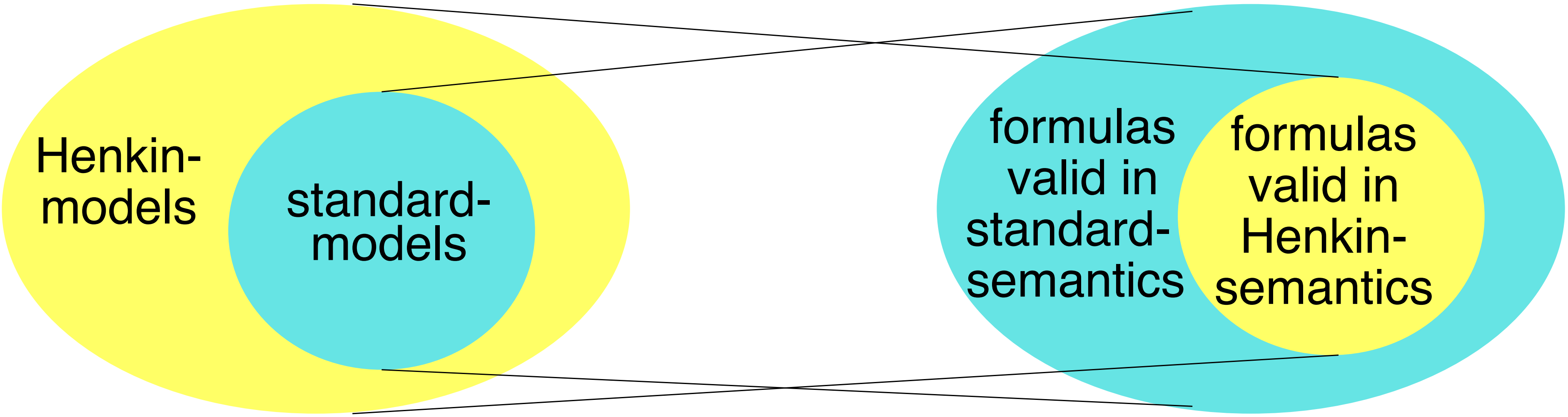
The generalization to Henkin models restricts the set of valid formulae sufficiently so that all of them can be proven by a Hilbert-style calculus [Henkin50].

Of course our HOL-CUBE is not complete here; we can axiomatically require the existence of particular (classes of) functions, e.g., by assuming the description or choice operators.

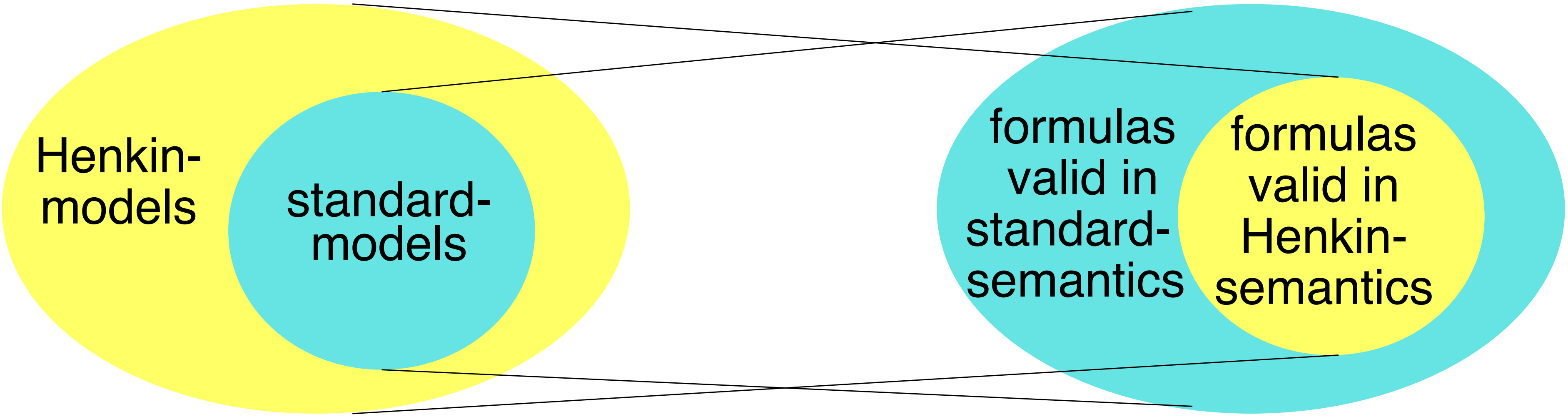
We will not pursue this here; for a detailed discussion of the semantic issues raised by the presence of these logical constants see [Andrews72].

Note that even though we can consider model classes with richer and richer function spaces, we can never reach standard models where function spaces are full while maintaining complete (recursively axiomatizable) calculi.

Standard Models and Henkin Models



Standard Models and Henkin Models



What has been our motivation for further generalization of Henkin semantics with respect to Boolean and functional extensionality?

Models without Functional Extensionality

Motivation: modeling programs as (higher-order) functions

- We might be interested in intensional properties like run-time complexity.

Models without Functional Extensionality

Motivation: modeling programs as (higher-order) functions

- We might be interested in intensional properties like run-time complexity.
- $\mathbf{I} := \lambda X.X$ and $\mathbf{L} := \lambda X.\mathsf{rev}(\mathsf{rev}(X))$, where rev is the self-inverse function.

Models without Functional Extensionality

Motivation: modeling programs as (higher-order) functions

- We might be interested in intensional properties like run-time complexity.
- $\mathbf{I} := \lambda X.X$ and $\mathbf{L} := \lambda X.\mathsf{rev}(\mathsf{rev}(X))$, where rev is the self-inverse function.
- The identity function has constant complexity, the function rev is linear in the length of its argument.

Models without Functional Extensionality

How do we account for models without functional extensionality?

- We have generalized the notion of domains at function types and evaluation functions.

Models without Functional Extensionality

How do we account for models without functional extensionality?

- We have generalized the notion of domains at function types and evaluation functions.
- The usual construction already uses sets of (extensional) functions for the domains of function type and the property of functionality to construct values for λ-terms.

Models without Functional Extensionality

How do we account for models without functional extensionality?

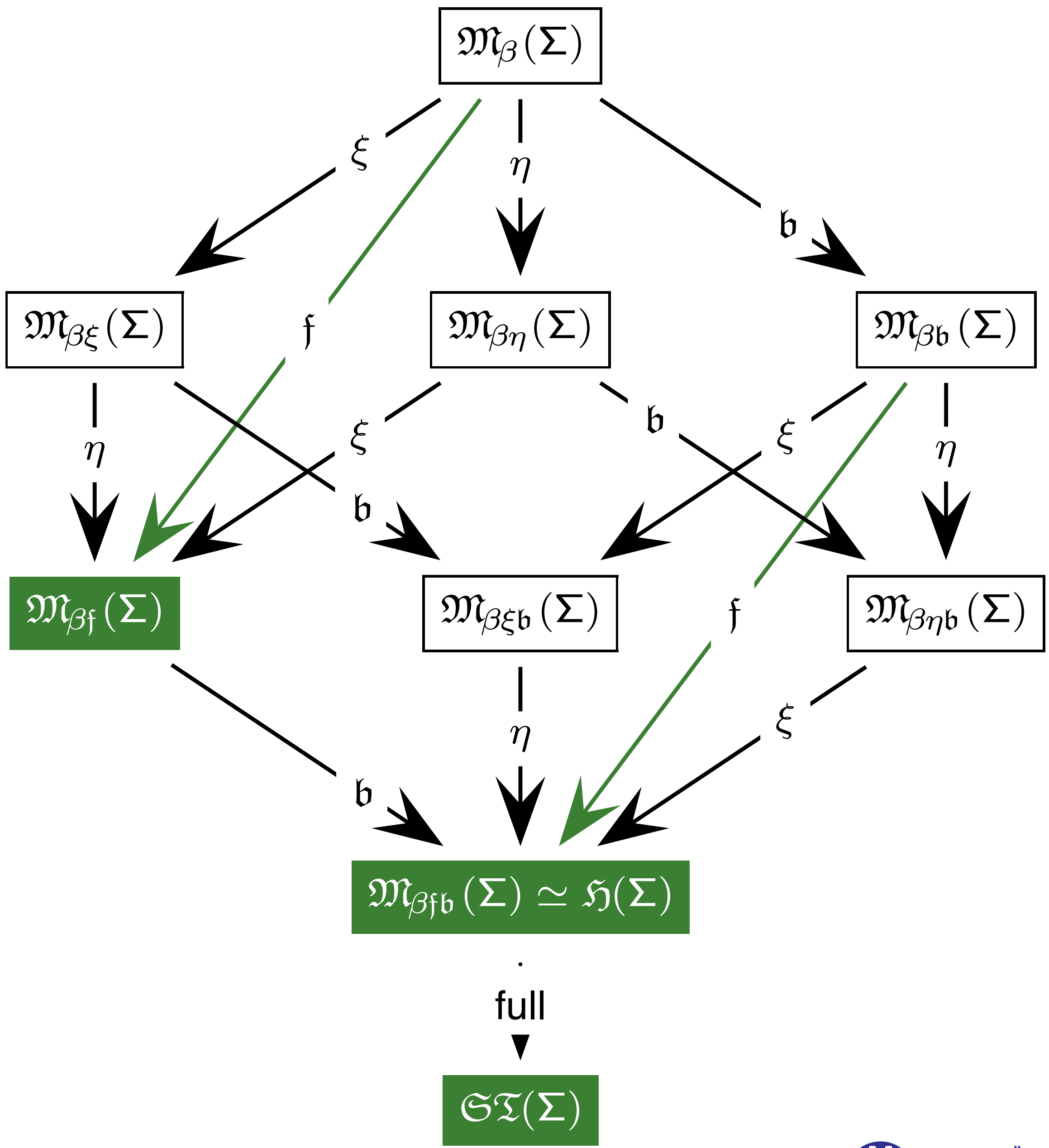
- We have generalized the notion of domains at function types and evaluation functions.
- The usual construction already uses sets of (extensional) functions for the domains of function type and the property of functionality to construct values for λ-terms.
- We build on the notion of applicative structures to define Σ-evaluations, where the evaluation function is assumed to respect application and β-conversion.

Models without Functional Extensionality

How do we account for models without functional extensionality?

- We have generalized the notion of domains at function types and evaluation functions.
- The usual construction already uses sets of (extensional) functions for the domains of function type and the property of functionality to construct values for λ-terms.
- We build on the notion of applicative structures to define Σ-evaluations, where the evaluation function is assumed to respect application and β-conversion.
- In such models, a function is not uniquely determined by its behavior on all possible arguments.

Semantics: HOL-CUBE



$\mathfrak{f}$: models are functional

$$\forall \mathsf{f}, \mathsf{g} \in \mathcal{D}_{\beta\alpha} :$$
$$\mathsf{f} = \mathsf{g} \text{ iff } \mathsf{f}@\mathsf{a} = \mathsf{g}@\mathsf{a} \ (\forall \mathsf{a} \in \mathcal{D}_{\alpha})$$

Models without η- or ξ-Functionality

Motivation: in standard literature functional extensionality is often is discussed in terms of

Models without η- or ξ-Functionality

Motivation: in standard literature functional extensionality is often is discussed in terms of

- ξ-functionality

Models without η- or ξ-Functionality

Motivation: in standard literature functional extensionality is often is discussed in terms of

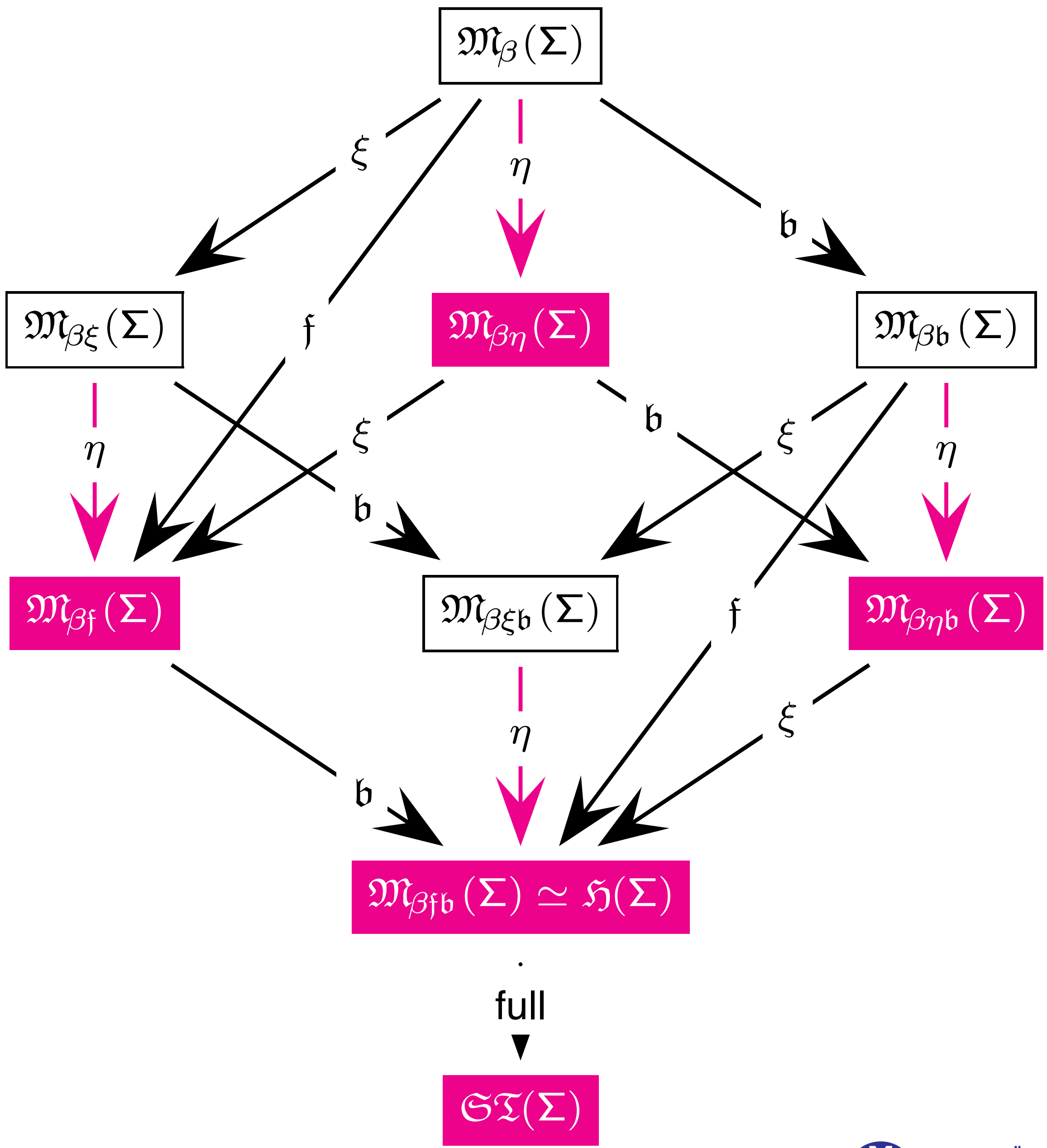
- ξ-functionality
- η-functionality

Models without η- or ξ-Functionality

Motivation: in standard literature functional extensionality is often is discussed in terms of

- ξ-functionality
- η-functionality
- Therefore, we integrated these two cases in our landscape.

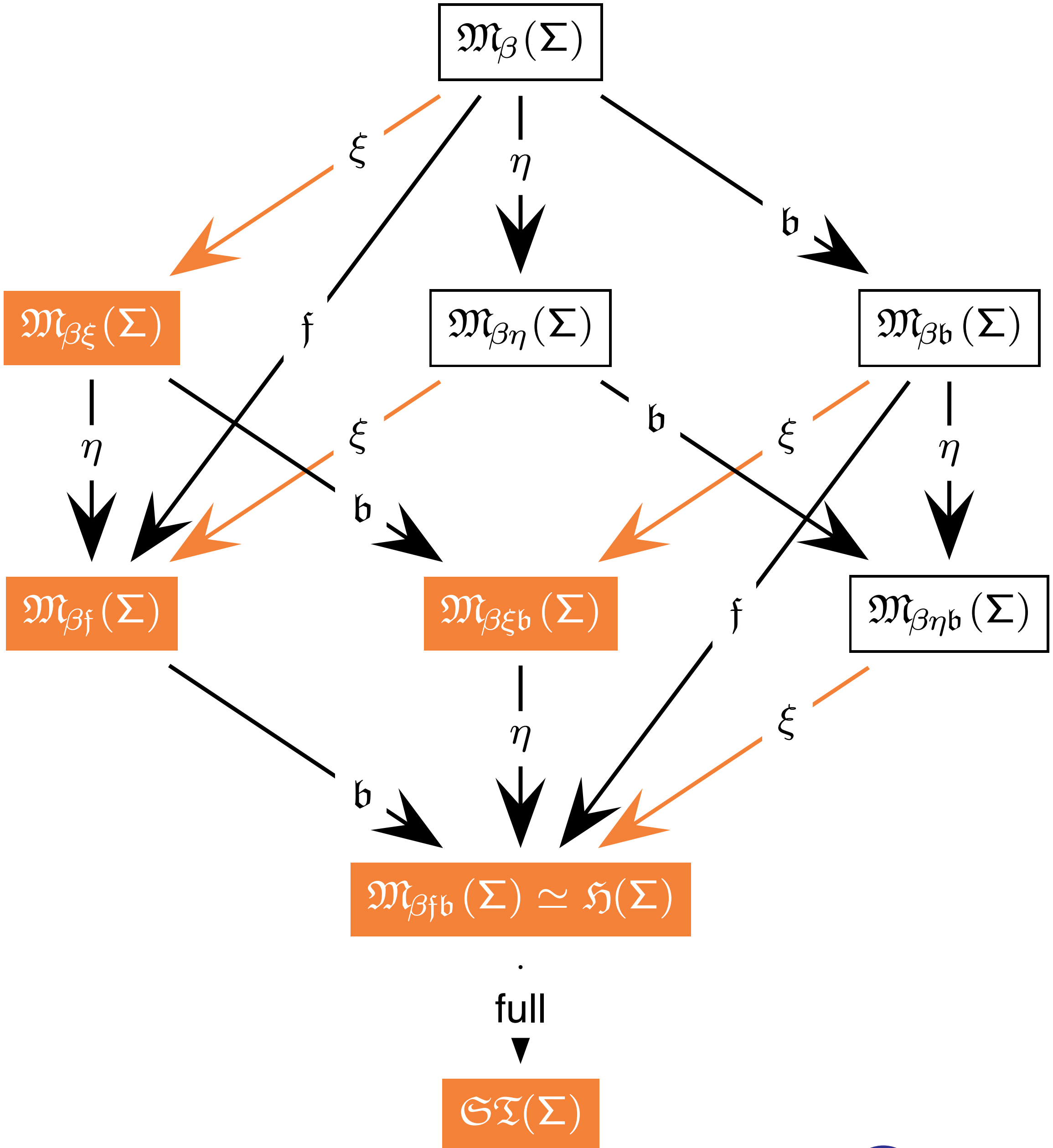
Semantics: HOL-CUBE



η: models are η-functional

$$\mathcal{E}_\varphi(\mathrm{A}) = \mathcal{E}_\varphi(\mathrm{A}\downarrow_{\beta\eta})$$

Semantics: HOL-CUBE



ξ: models are ξ-functional

$\mathcal{E}_\varphi(\lambda X_\alpha . M_\beta) = \mathcal{E}_\varphi(\lambda X_\alpha . N_\beta)$ iff
$\mathcal{E}_{\varphi,[a/X]}(M) = \mathcal{E}_{\varphi,[a/X]}(N)$ $(\forall a \in \mathcal{D}_\alpha)$

Models without Boolean Extensionality

Motivation: Semantics of natural language

Models without Boolean Extensionality

Motivation: Semantics of natural language

- We may not want to commit to a logic where the sentence
  "John believes that Phil is a woodchuck"

Models without Boolean Extensionality

Motivation: Semantics of natural language

- We may not want to commit to a logic where the sentence "John believes that Phil is a woodchuck" automatically entails "John believes that Phil is a groundhog"

Models without Boolean Extensionality

Motivation: Semantics of natural language

- We may not want to commit to a logic where the sentence "John believes that Phil is a woodchuck" automatically entails "John believes that Phil is a groundhog" since John might not know that "woodchuck" is just another word for "groundhog".

Models without Boolean Extensionality

Motivation: Semantics of natural language

- We may not want to commit to a logic where the sentence "John believes that Phil is a woodchuck" automatically entails "John believes that Phil is a groundhog" since John might not know that "woodchuck" is just another word for "groundhog".
- However, Boolean extensionality does just that: whenever two propositions are equivalent, they must be equal, and can be substituted for each other.

Models without Boolean Extensionality

Motivation: Semantics of natural language

- We may not want to commit to a logic where the sentence "John believes that Phil is a woodchuck" automatically entails "John believes that Phil is a groundhog" since John might not know that "woodchuck" is just another word for "groundhog".
- However, Boolean extensionality does just that: whenever two propositions are equivalent, they must be equal, and can be substituted for each other.
- Another example: $\mathsf{obvious}(\mathbf{O})$ and $\mathsf{obvious}(\mathbf{F})$ where $\mathbf{O} := 2 + 2 = 4$ and $\mathbf{F} := \forall n > 2.x^n + y^n = z^n \Rightarrow x = y = z = 0$ should not be equivalent, even if their arguments are.

Models without Boolean Extensionality

Motivation: Semantics of natural language

- We may not want to commit to a logic where the sentence "John believes that Phil is a woodchuck" automatically entails "John believes that Phil is a groundhog" since John might not know that "woodchuck" is just another word for "groundhog".
- However, Boolean extensionality does just that: whenever two propositions are equivalent, they must be equal, and can be substituted for each other.
- Another example: $\mathsf{obvious}(\mathbf{O})$ and $\mathsf{obvious}(\mathbf{F})$ where $\mathbf{O} := 2 + 2 = 4$ and $\mathbf{F} := \forall n > 2.x^n + y^n = z^n \Rightarrow x = y = z = 0$ should not be equivalent, even if their arguments are.
- Such phenomena have been studied under the heading of "hyper-intensional semantics" in theoretical semantics.

Models without Boolean Extensionality

How do we account for models without Boolean extensionality?

- We have weakened the assumption that $\mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$, since this entails that the values of $\mathbf{O}$ and $\mathbf{F}$ are identical.

Models without Boolean Extensionality

How do we account for models without Boolean extensionality?

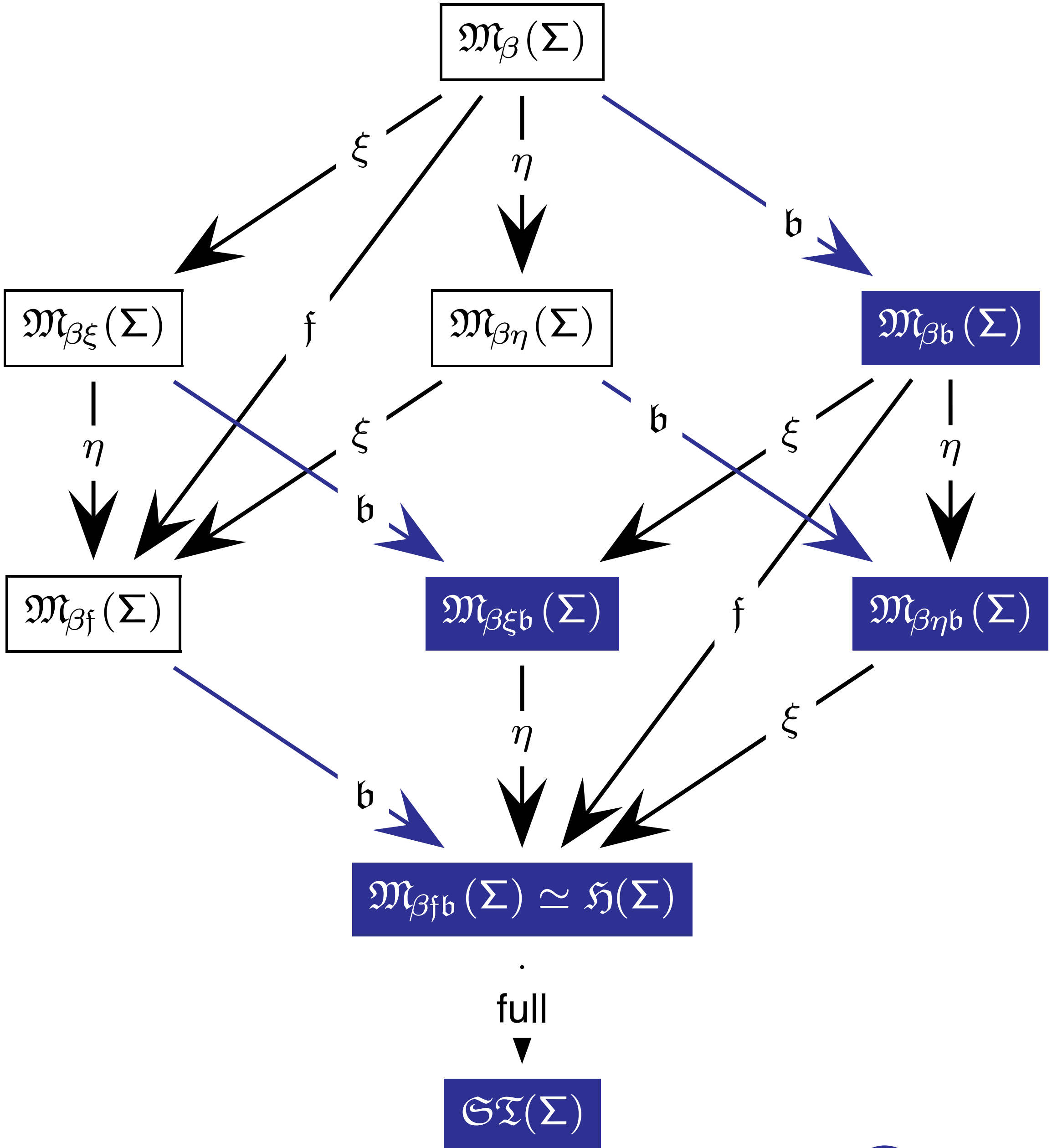
- We have weakened the assumption that $\mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$, since this entails that the values of $\mathbf{O}$ and $\mathbf{F}$ are identical.
- In our Σ-models without property $\mathfrak{b}$ we only insist that there is a division of the truth values into "good" and "bad" ones, which we express by insisting on the existence of a valuation υ of $\mathcal{D}_o$, i.e., a function $\upsilon : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ that is coordinated with the interpretations of the logical constants $\neg$, $\vee$, and Π^α (for each type α).

Models without Boolean Extensionality

How do we account for models without Boolean extensionality?

- We have weakened the assumption that $\mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$, since this entails that the values of $\mathbf{O}$ and $\mathbf{F}$ are identical.
- In our Σ-models without property $\mathfrak{b}$ we only insist that there is a division of the truth values into "good" and "bad" ones, which we express by insisting on the existence of a valuation υ of $\mathcal{D}_o$, i.e., a function $\upsilon: \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ that is coordinated with the interpretations of the logical constants $\neg$, $\vee$, and Π^α (for each type α).
- Notion of validity: we call a sentence $\mathbf{A}$ valid in such a model if $\upsilon(\mathsf{a}) = \mathtt{T}$, where $\mathsf{a} \in \mathcal{D}_o$ is the denotation of the sentence $\mathbf{A}$.

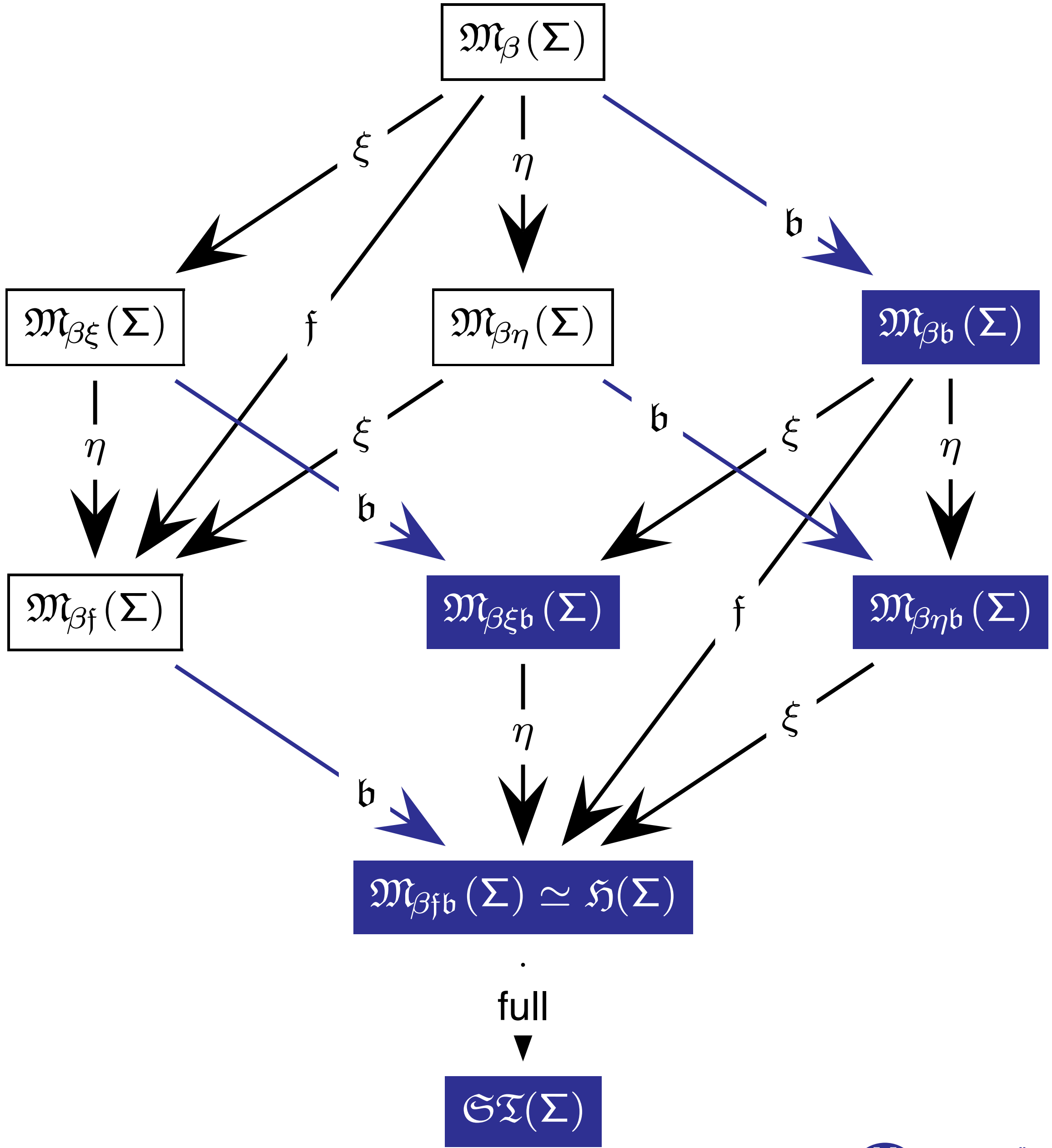
Semantics: HOL-CUBE



$\mathfrak{b}$: models are Boolean extensional

v is injective

Semantics: HOL-CUBE



$\mathfrak{b}$: models are Boolean extensional

v is injective

If Σ contains sufficiently many logical constants:

$\mathcal{D}_o = \{\bot, \top\}$

Semantics and Theorem Proving: Test Problems for Theorem Provers

Test Problems for Theorem Provers

- Test problems for FOL theorem provers

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.
  - TPTP [PelletierSutcliffeSuttner02]

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.
  - TPTP [PelletierSutcliffeSuttner02]
  - significantly fostered the development of FOL ATPs

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.
  - TPTP [PelletierSutcliffeSuttner02]
  - significantly fostered the development of FOL ATPs
- Test problems for HOL theorem provers

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.
  - TPTP [PelletierSutcliffeSuttner02]
  - significantly fostered the development of FOL ATPs
- Test problems for HOL theorem provers
  - common library missing

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.
  - TPTP [PelletierSutcliffeSuttner02]
  - significantly fostered the development of FOL ATPs
- Test problems for HOL theorem provers
  - common library missing
- Following slides: example problems from our paper [TPHOLS-05]

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.
  - TPTP [PelletierSutcliffeSuttner02]
  - significantly fostered the development of FOL ATPs
- Test problems for HOL theorem provers
  - common library missing
- Following slides: example problems from our paper [TPHOLS-05]
- Are we proposing challenging HOL benchmark problems?

Test Problems for Theorem Provers

- Test problems for FOL theorem provers
  - [McCharenOverbeekWos76], [WilsonMinker79], [Pelletier86], etc.
  - TPTP [PelletierSutcliffeSuttner02]
  - significantly fostered the development of FOL ATPs
- Test problems for HOL theorem provers
  - common library missing
- Following slides: example problems from our paper [TPHOLS-05]
- Are we proposing challenging HOL benchmark problems?
  - No!!!

Test Problems for Theorem Provers

- Examples are simple

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point
  - easy to understand and easy to encode

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point
  - easy to understand and easy to encode
  - relevant for both: automated and interactive TP

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point
  - easy to understand and easy to encode
  - relevant for both: automated and interactive TP
- Examples are structured

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point
  - easy to understand and easy to encode
  - relevant for both: automated and interactive TP
- Examples are structured
  - quick indicators for completeness and soundness wrt to HOL model classes from [Benzm.BrownKohlhase-JSL-04]

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point
  - easy to understand and easy to encode
  - relevant for both: automated and interactive TP
- Examples are structured
  - quick indicators for completeness and soundness wrt to HOL model classes from [Benzm.BrownKohlhase-JSL-04]
  - shall precede formal soundness / completeness analysis

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point
  - easy to understand and easy to encode
  - relevant for both: automated and interactive TP
- Examples are structured
  - quick indicators for completeness and soundness wrt to HOL model classes from [Benzm.BrownKohlhase-JSL-04]
  - shall precede formal soundness / completeness analysis
  - many are collected from experience with LEO and TPS

Test Problems for Theorem Provers

- Examples are simple
  - highlight the essence of some semantical or technical point
  - easy to understand and easy to encode
  - relevant for both: automated and interactive TP
- Examples are structured
  - quick indicators for completeness and soundness wrt to HOL model classes from [Benzm.BrownKohlhase-JSL-04]
  - shall precede formal soundness / completeness analysis
  - many are collected from experience with LEO and TPS
- (Some more challenging examples are also added in [TPHOLS-05])

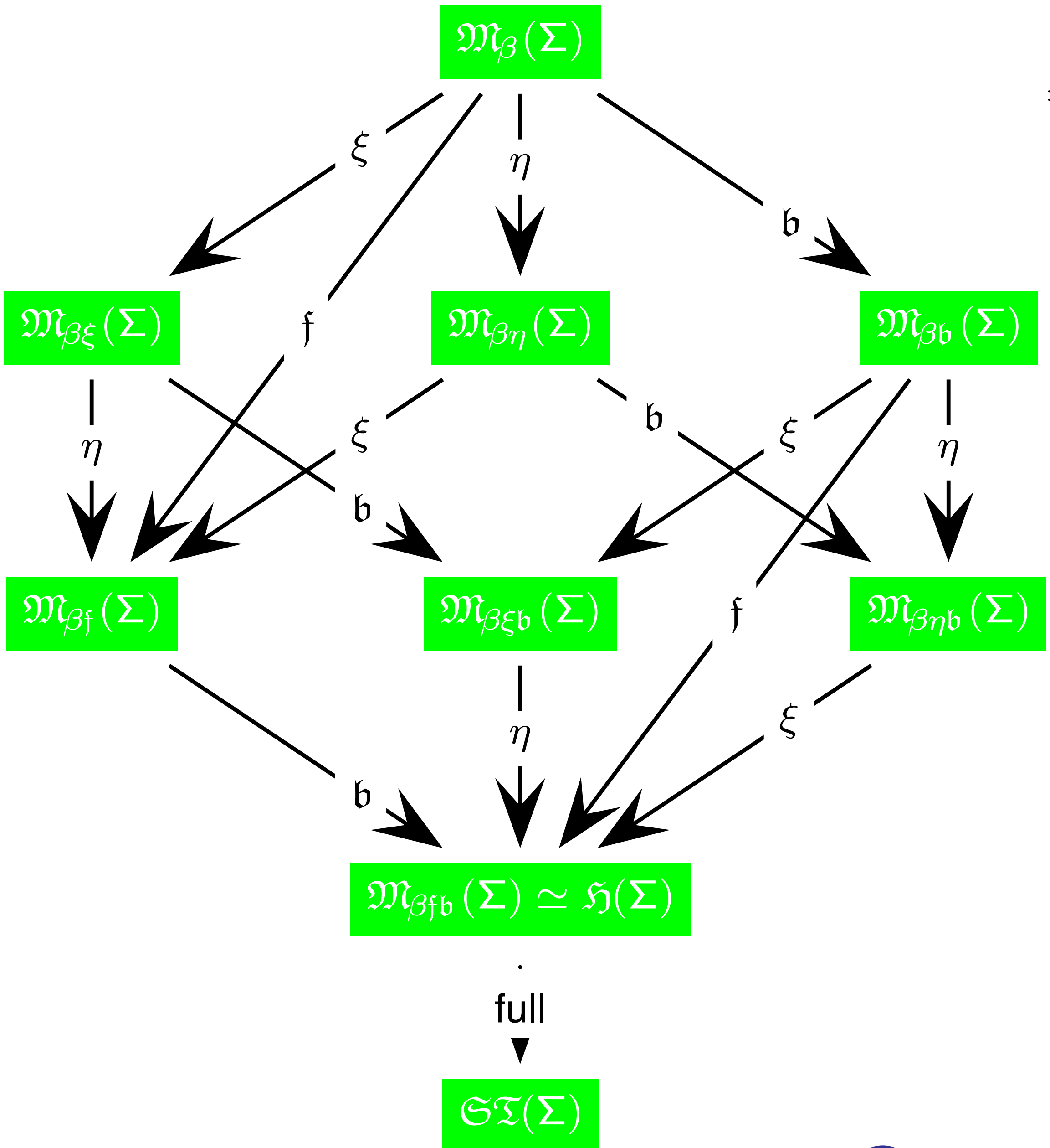
Remark: Signature

Unless stated otherwise we assume on the following slides that our signature Σ contains the following logical connectives:

$$\{\top, \bot, \neg, \wedge, \vee, \supset, \Leftrightarrow\} \cup \{\Pi^\alpha, \Sigma^\alpha, =^\alpha\}$$

(less logical connectives are possible)

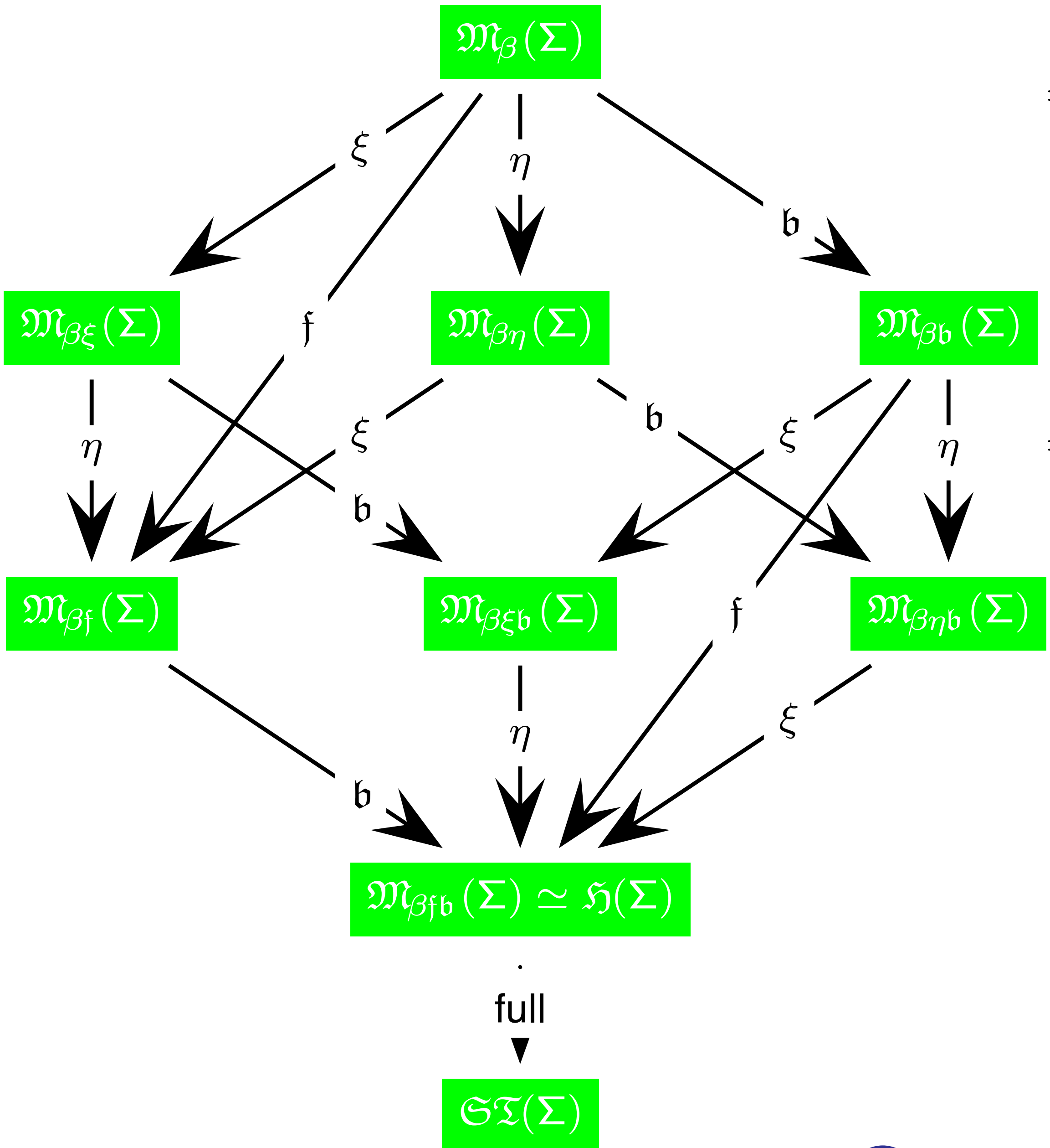
HOL-Problems: β



$\stackrel{*}{\doteq}$ **is equivalence relation**

- $\forall X_\alpha. X \stackrel{*}{\doteq} X$
- $\forall X_\alpha, Y_\alpha. X \stackrel{*}{\doteq} Y \supset Y \stackrel{*}{\doteq} X$
- $\forall X_\alpha, Y_\alpha, Z_\alpha. (X \stackrel{*}{\doteq} Y \wedge Y \stackrel{*}{\doteq} Z) \supset X \stackrel{*}{\doteq} Z$

HOL-Problems: β



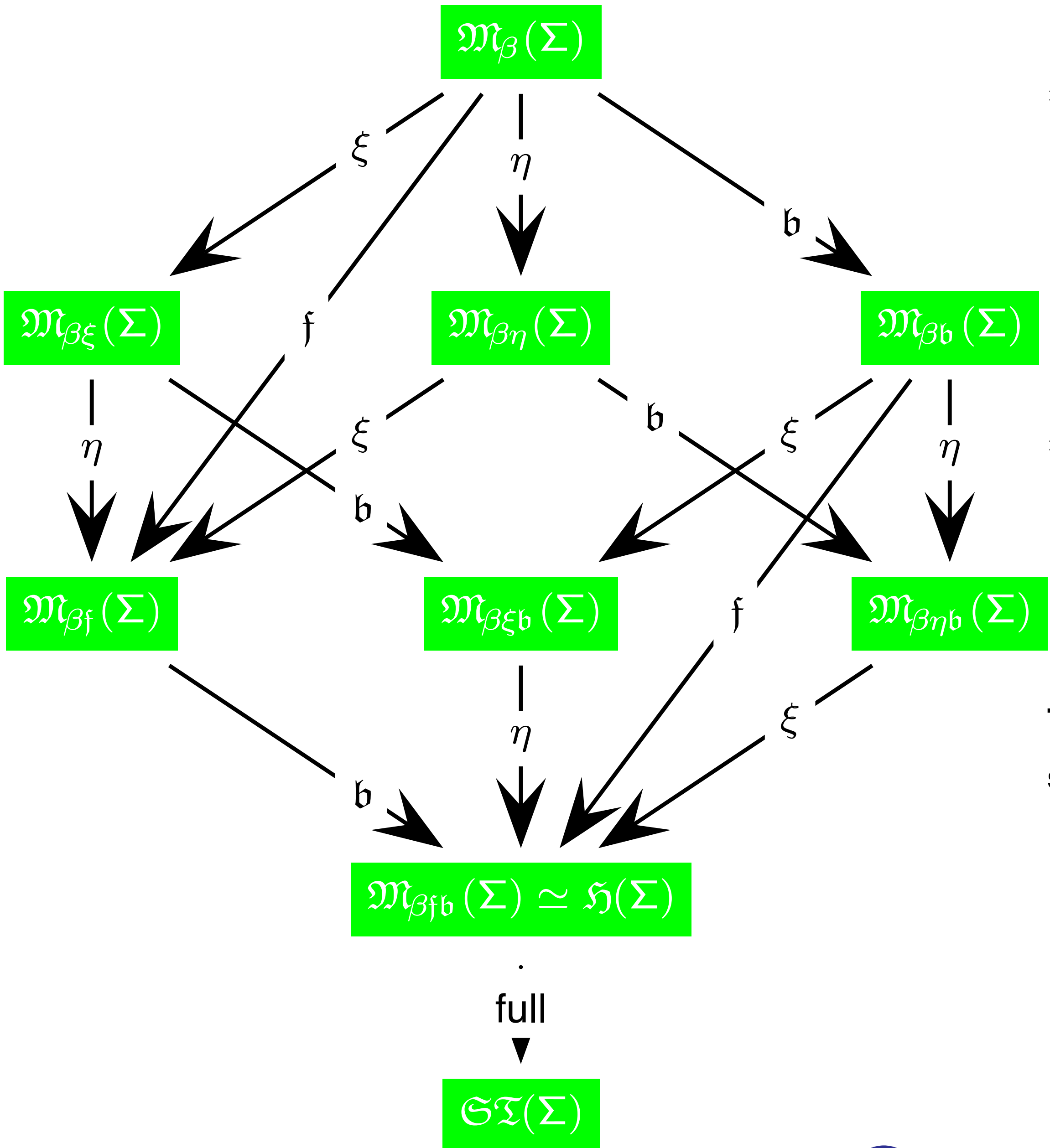
$\stackrel{*}{=}$ is equivalence relation

- $\forall X_\alpha. X \stackrel{*}{=} X$
- $\forall X_\alpha, Y_\alpha. X \stackrel{*}{=} Y \supset Y \stackrel{*}{=} X$
- $\forall X_\alpha, Y_\alpha, Z_\alpha. (X \stackrel{*}{=} Y \wedge Y \stackrel{*}{=} Z) \supset X \stackrel{*}{=} Z$

$\stackrel{*}{=}$ is congruence relation

- $\forall X_\alpha, Y_\alpha, F_{\alpha\alpha}. X \stackrel{*}{=} Y \supset (FX) \stackrel{*}{=} (FY)$
- $\forall X_\alpha, Y_\alpha, P_{o\alpha}. X \stackrel{*}{=} Y \wedge (PX) \supset (PY)$

HOL-Problems: β



$\stackrel{*}{\doteq}$ is equivalence relation

- $\forall X_\alpha . X \stackrel{*}{\doteq} X$
- $\forall X_\alpha, Y_\alpha . X \stackrel{*}{\doteq} Y \supset Y \stackrel{*}{\doteq} X$
- $\forall X_\alpha, Y_\alpha, Z_\alpha . (X \stackrel{*}{\doteq} Y \wedge Y \stackrel{*}{\doteq} Z) \supset X \stackrel{*}{\doteq}$ Z

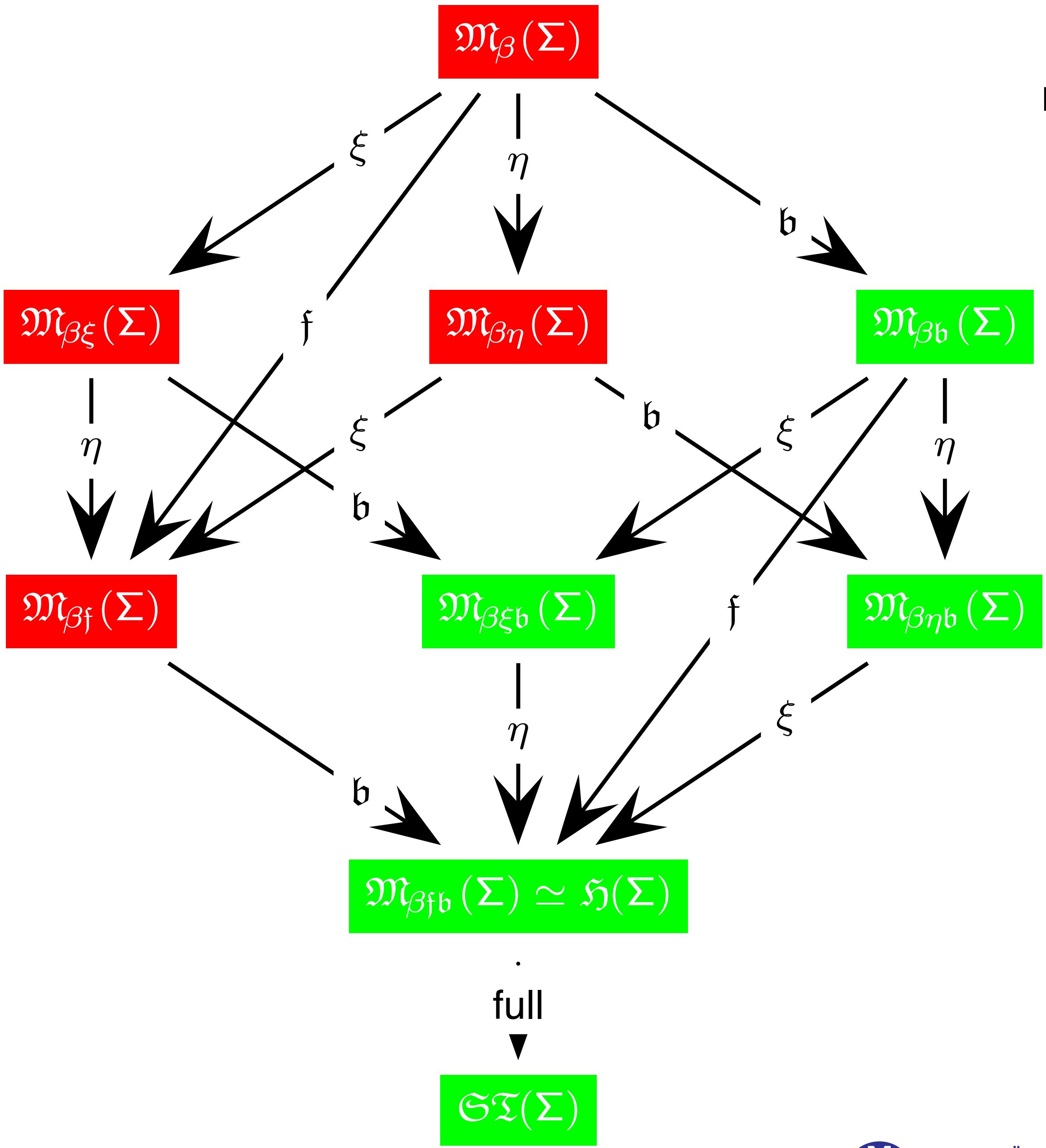
$\stackrel{*}{\doteq}$ is congruence relation

- $\forall X_\alpha, Y_\alpha, F_{\alpha\alpha} . X \stackrel{*}{\doteq} Y \supset (FX) \stackrel{*}{\doteq} (FY)$
- $\forall X_\alpha, Y_\alpha, P_{o\alpha} . X \stackrel{*}{\doteq} Y \wedge (PX) \supset (PY)$

Trivial directions of Boolean and functional extensionality

- $\forall A_o, B_o . A \stackrel{*}{\doteq} B \supset (A \Leftrightarrow B)$
- $\forall F_{\beta\alpha}, G_{\beta\alpha} . F \stackrel{*}{\doteq} G \supset (\forall X_\alpha . FX \stackrel{*}{\doteq} GX)$

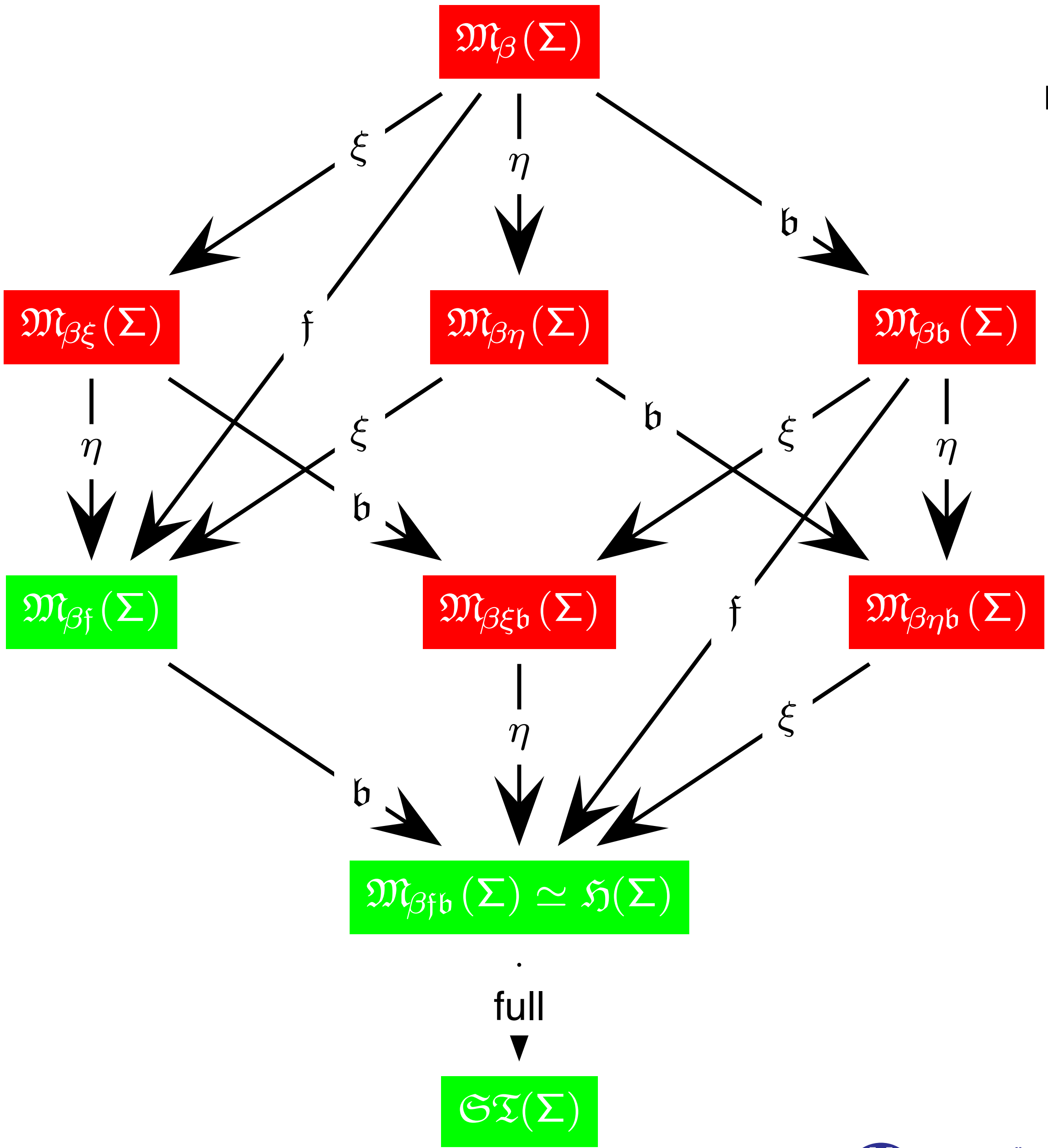
HOL-Problems: $\mathfrak{b}$



Non-trivial direction of Boolean extensionality

- $\forall A_o, B_o . (A \Leftrightarrow B) \supset A \stackrel{*}{=} B$

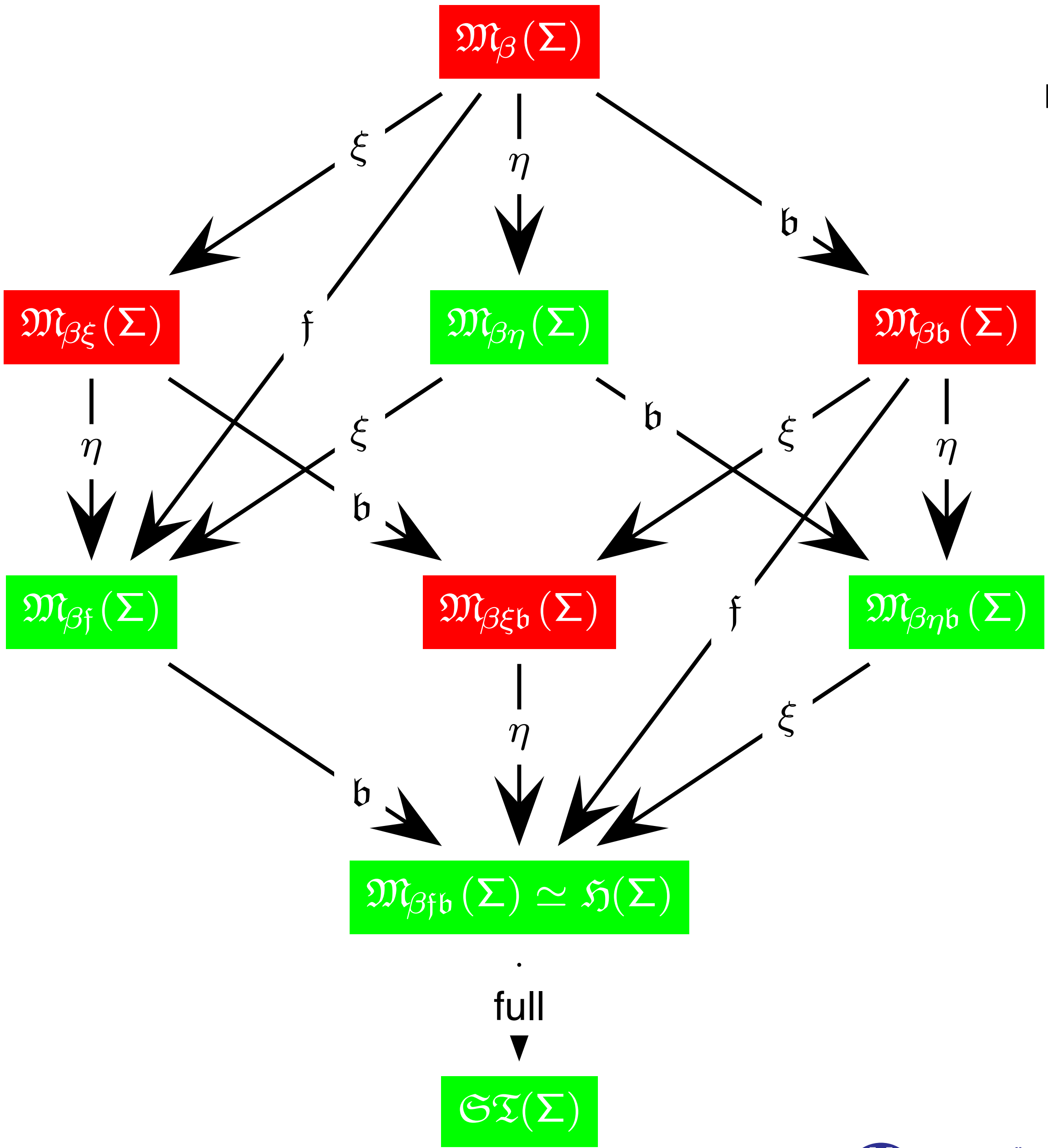
HOL-Problems: $\mathfrak{f}$



Non-trivial direct. of functional extensionality

- $\forall F_{\beta\alpha}, G_{\beta\alpha}.(\forall X_\alpha.FX \stackrel{*}{\doteq} GX) \supset F \stackrel{*}{\doteq} G$

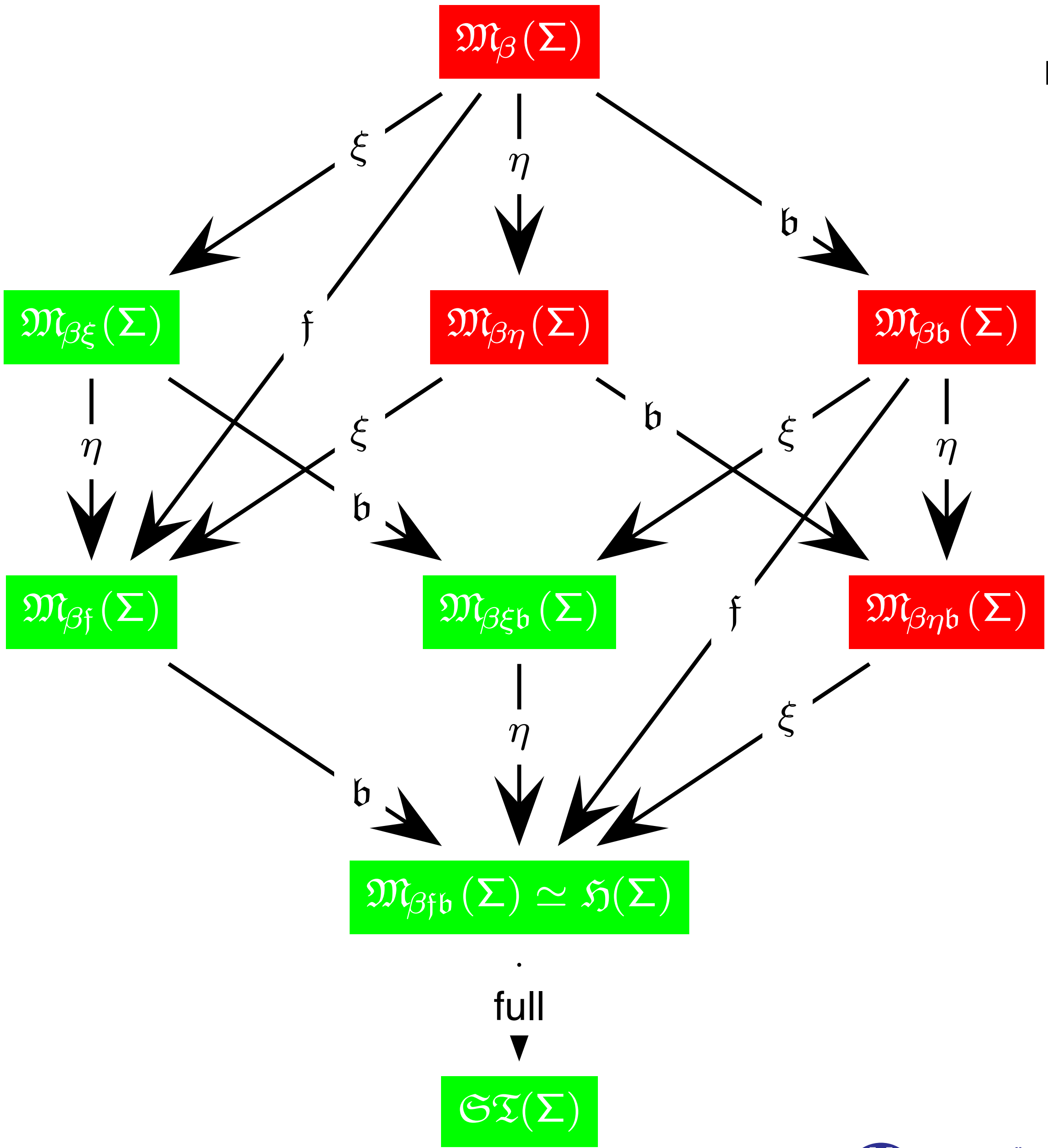
HOL-Problems: η



Example requiring property η

- $(p_{o(\iota\iota)}(\lambda X_\iota . f_{\iota\iota} X)) \supset (p\ f)$

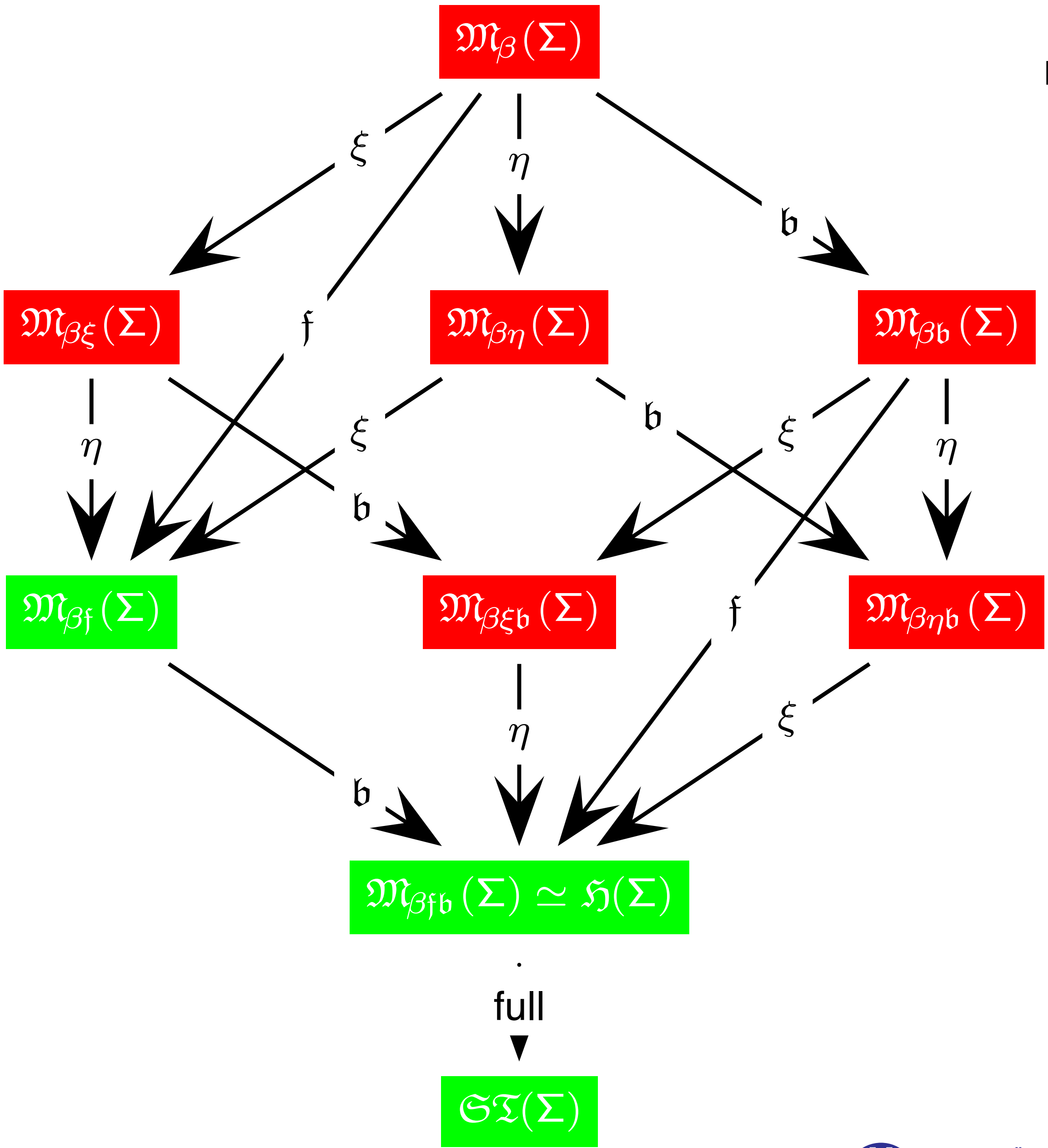
HOL-Problems: ξ



Example requiring property ξ (and $\mathfrak{q}$!)

- $(\forall X_\iota . (f_{\iota\iota} X) \stackrel{*}{\doteq} X) \wedge p_{o(\iota\iota)}(\lambda X_\iota . X) \supset p(\lambda X_\iota . f X)$

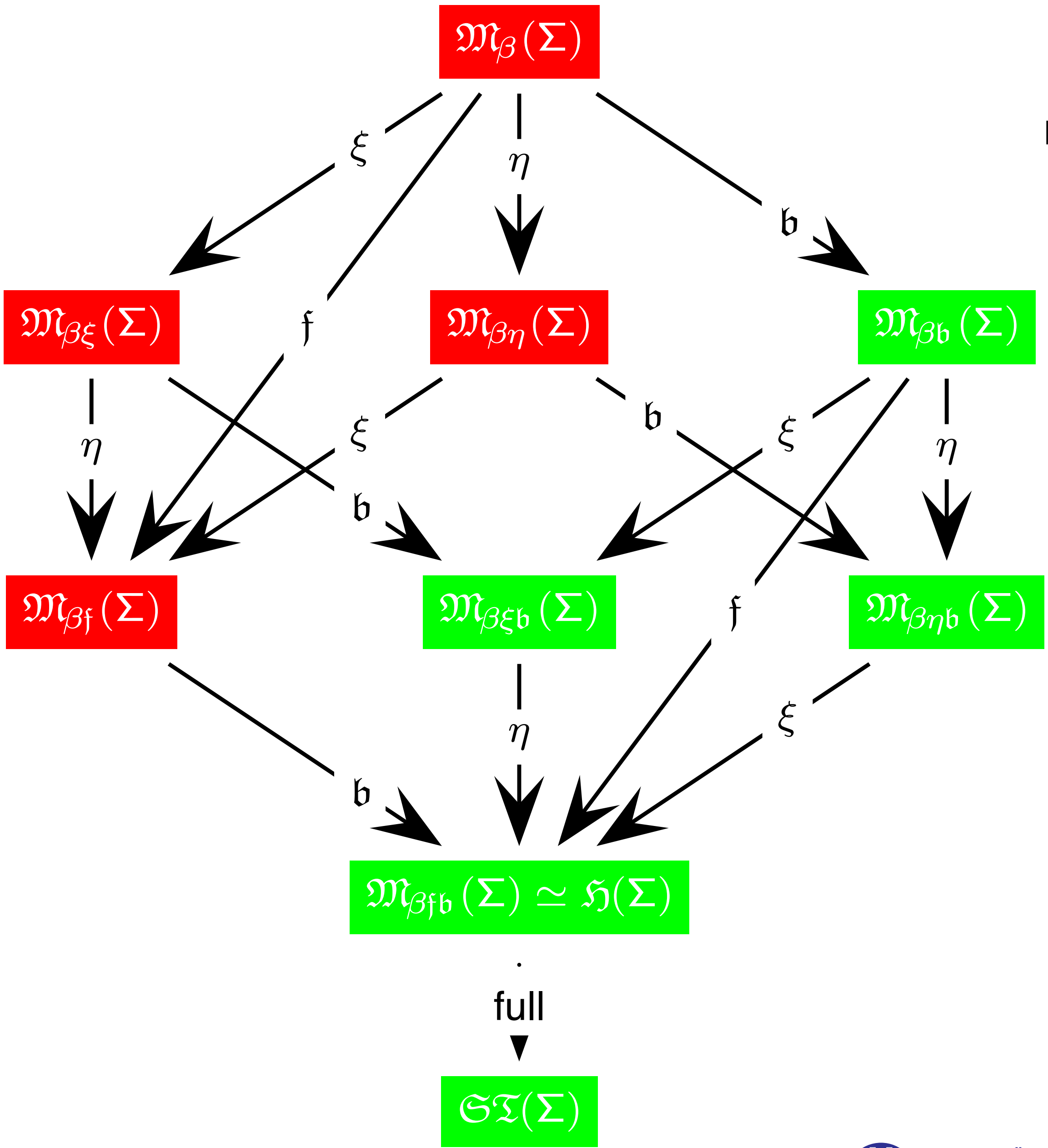
HOL-Problems: $\mathfrak{f}$



Example requiring property $\mathfrak{f}$ (and $\mathfrak{q}$!)

- $(\forall X_\iota . (f_{\iota\iota} X) \stackrel{*}{\doteq} X) \wedge p_{o(\iota\iota)}(\lambda X_\iota . X) \supset (p\ f)$

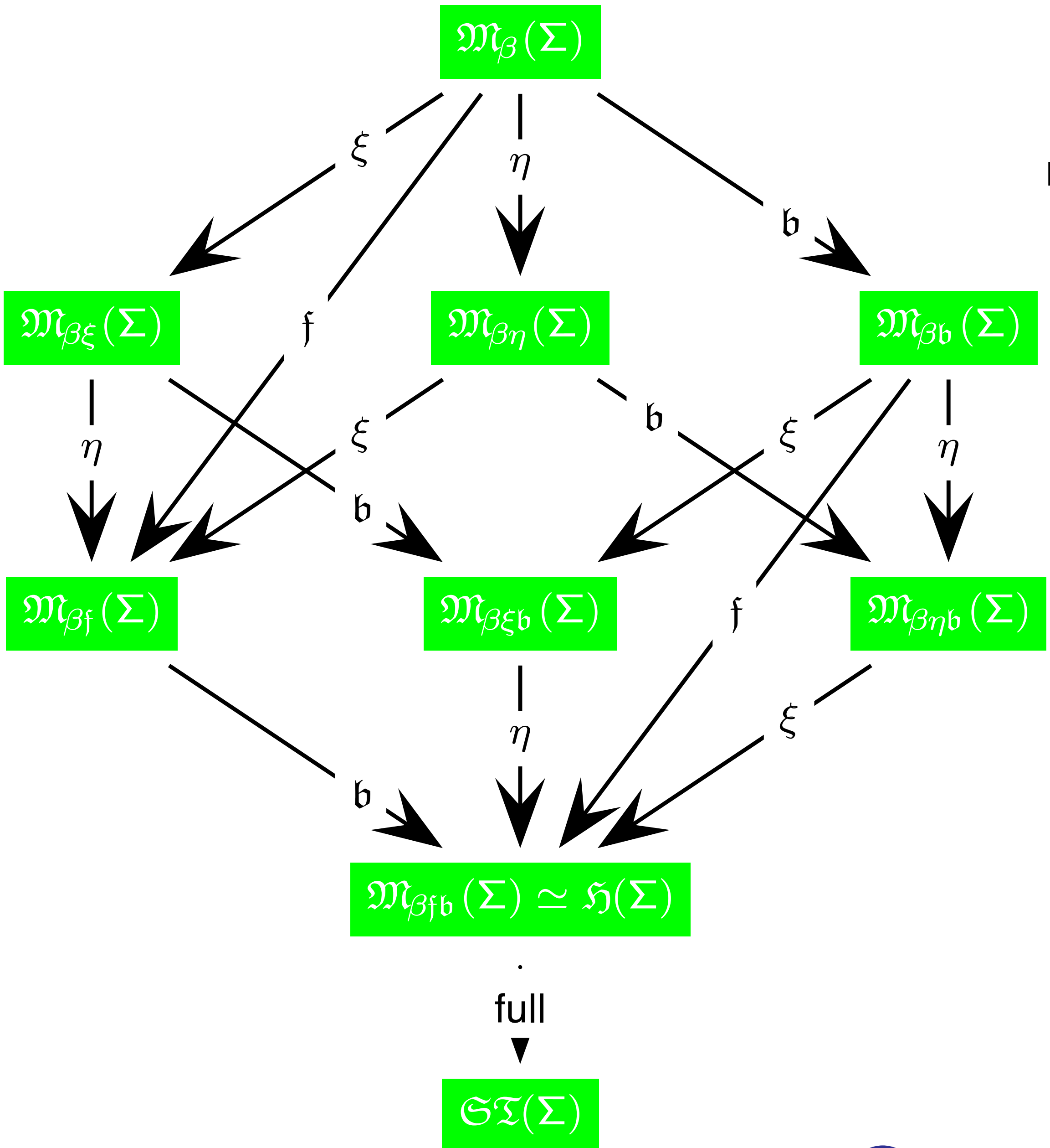
HOL-Problems: $\mathfrak{b}$



Examples requiring property $\mathfrak{b}$

- $(p_{oo}\ a_o) \wedge (p\ b_o) \Rightarrow (p\ (a \wedge b))$
- $\neg(a \stackrel{*}{\doteq} \neg a)$ (in particular $\neg(a = \neg a)$)
- $(h_{\iota o}((h\top) \stackrel{*}{\doteq} (h\bot))) \stackrel{*}{\doteq} (h\bot)$

HOL-Problems: Other Examples

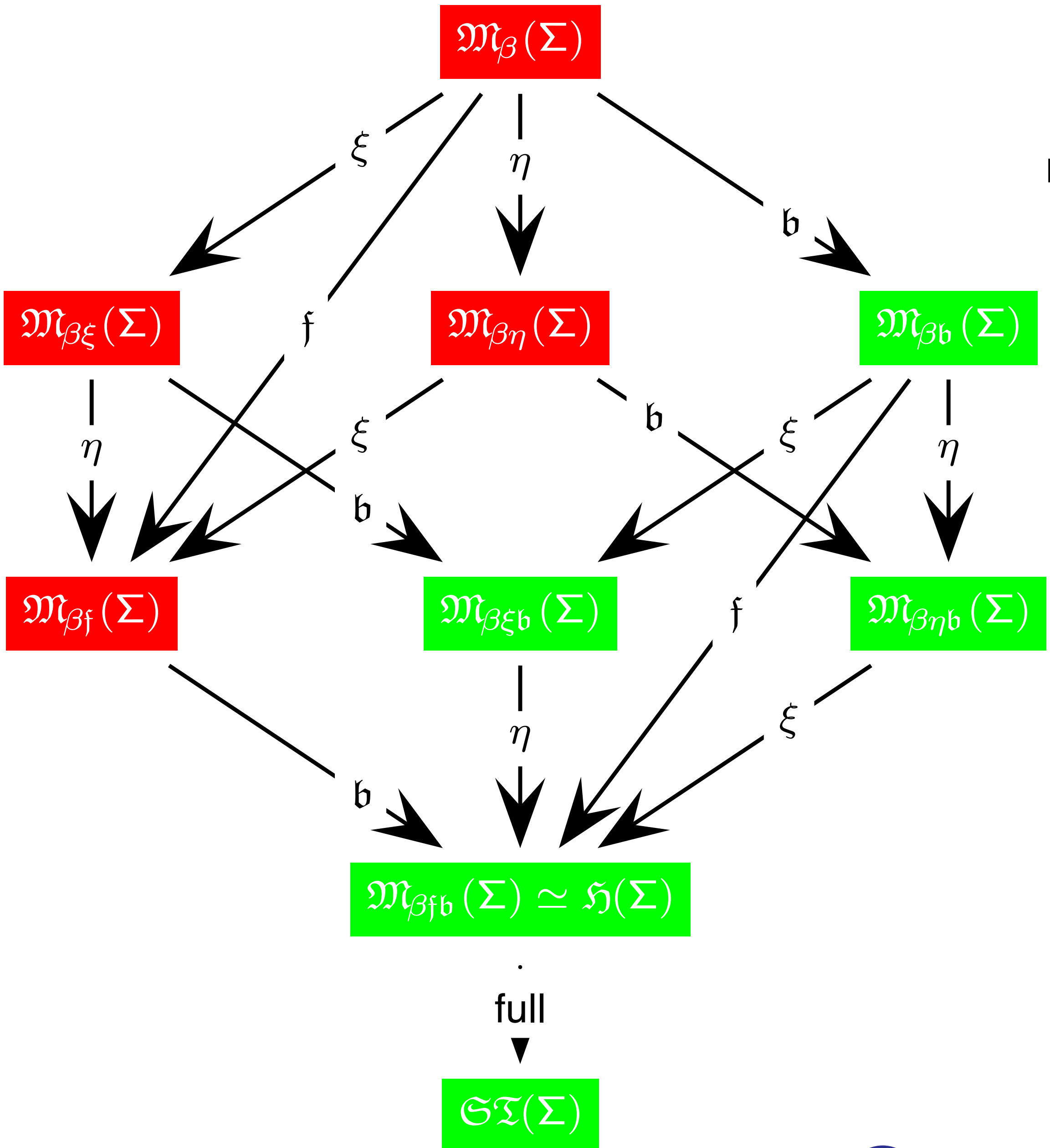


Playing with DeMorgan's Law:

- $\forall X, Y. X \wedge Y \Leftrightarrow \neg(\neg X \vee \neg Y)$

'Ok' for all model classes

HOL-Problems: DeMorgan's Law

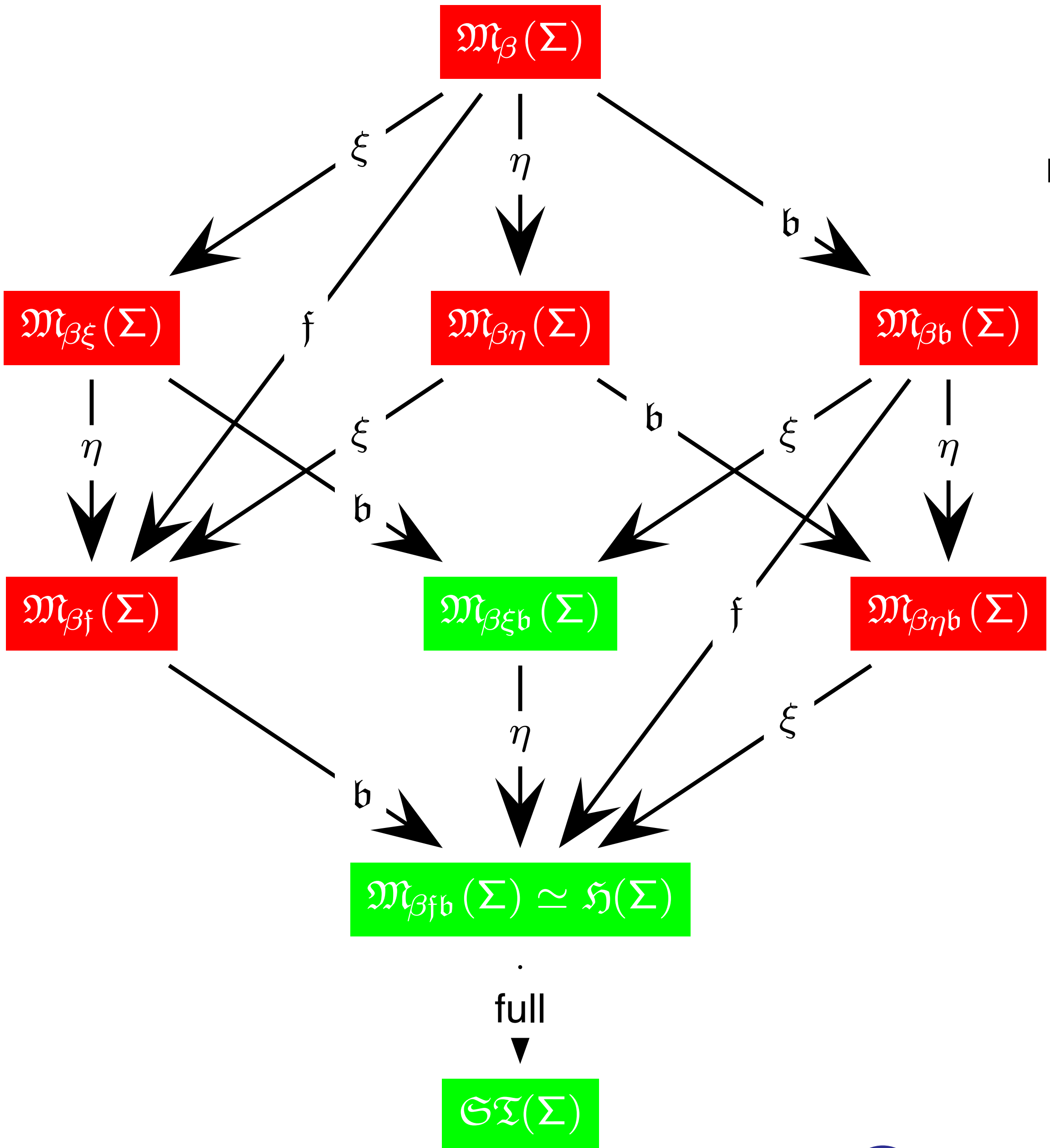


Playing with DeMorgan's Law:

- $\forall X, Y.X \land Y \Leftrightarrow \neg(\neg X \lor \neg Y)$
- $\forall X, Y.X \land Y \stackrel{*}{=} \neg(\neg X \lor \neg Y)$

requires $\mathfrak{b}$

HOL-Problems: DeMorgan's Law

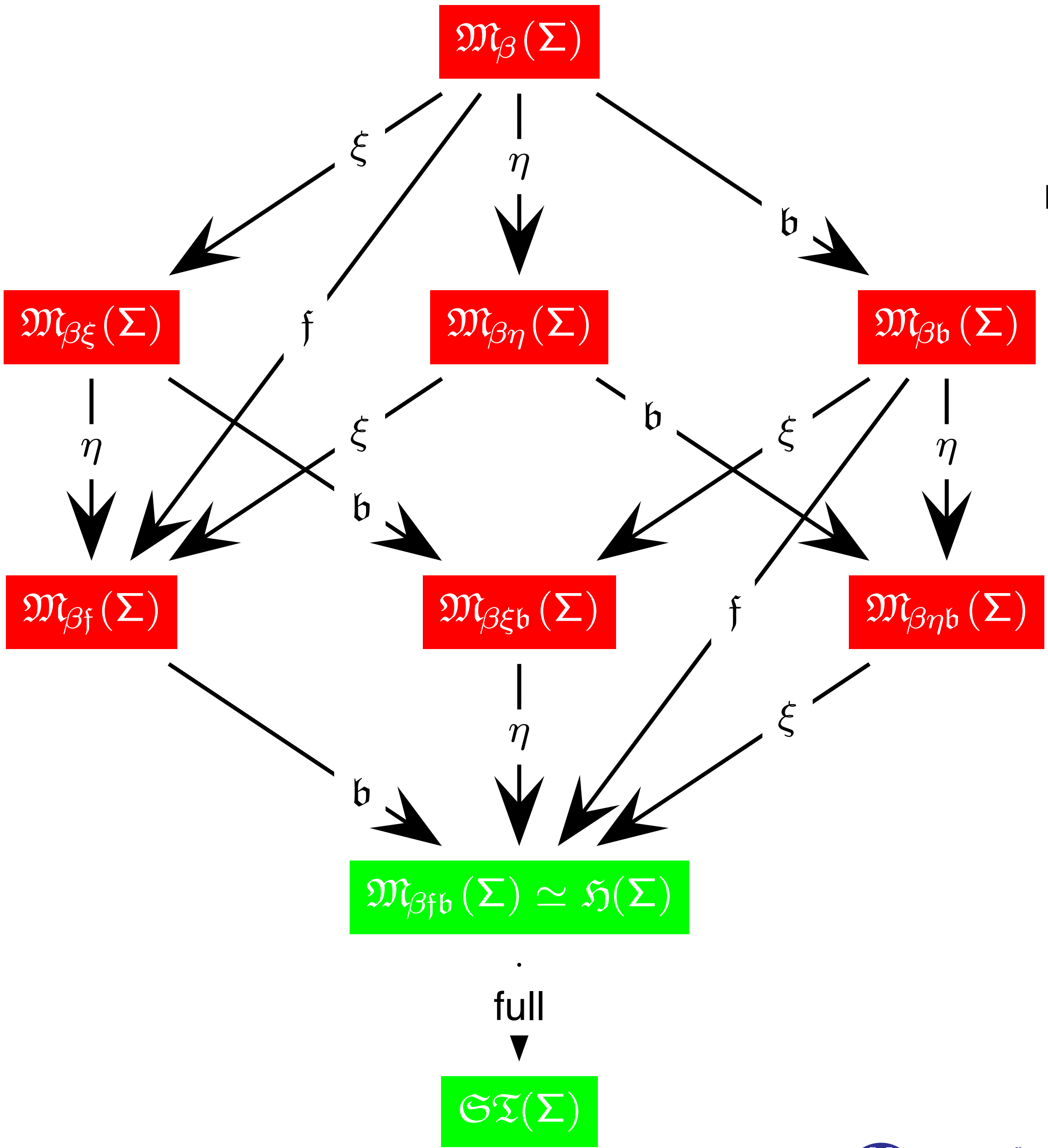


Playing with DeMorgan's Law:

- $\forall X, Y.X \wedge Y \Leftrightarrow \neg(\neg X \vee \neg Y)$
- $\forall X, Y.X \wedge Y \stackrel{*}{=} \neg(\neg X \vee \neg Y)$
- $(\lambda U \lambda V.U \wedge V) \stackrel{*}{=} (\lambda X \lambda Y.\neg(\neg X \vee \neg Y))$

$\boxed{\text{requires } \mathfrak{b} \text{ and } \xi}$

HOL-Problems: DeMorgan's Law

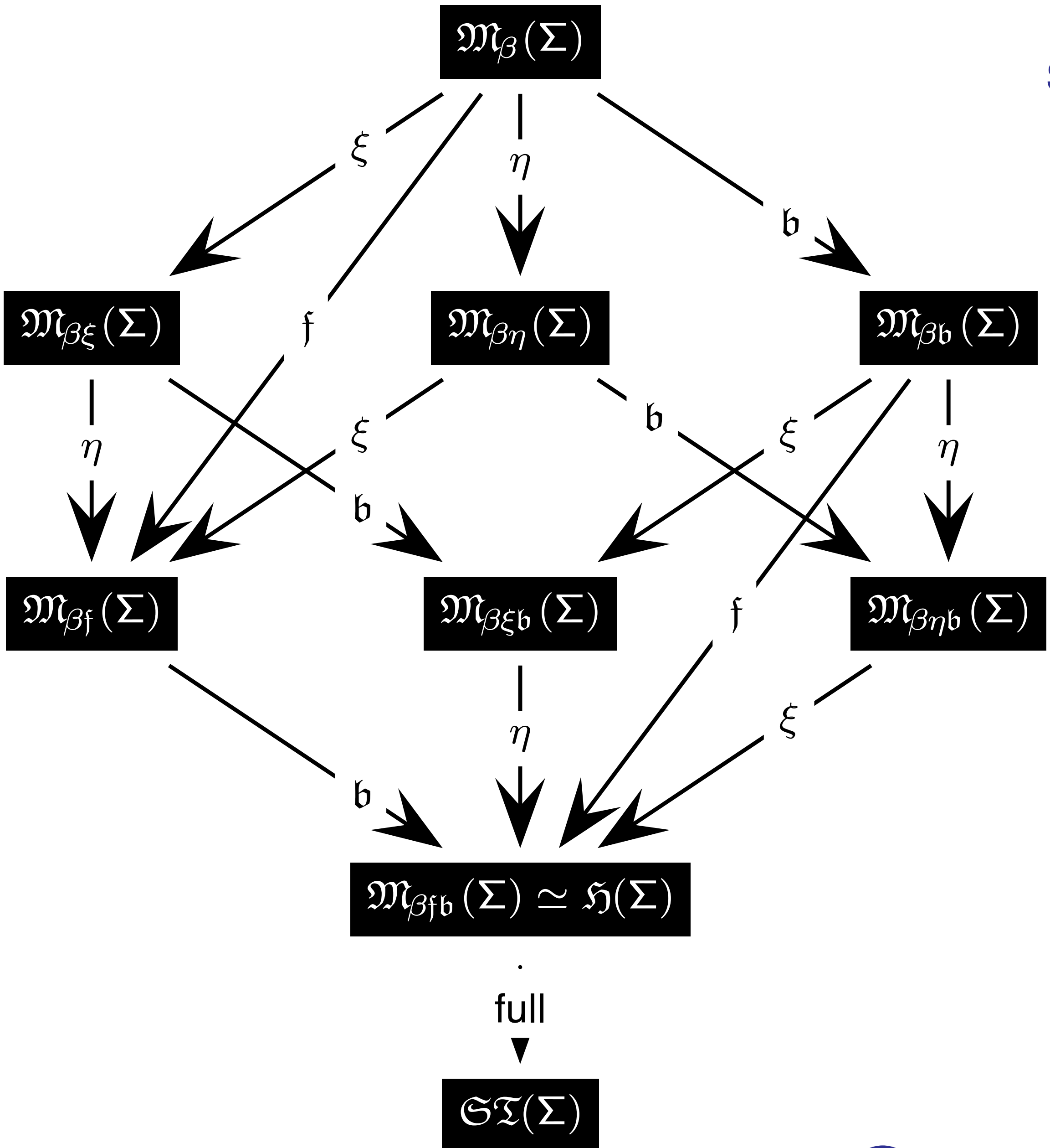


Playing with DeMorgan's Law:

- $\forall X, Y. X \wedge Y \Leftrightarrow \neg(\neg X \vee \neg Y)$
- $\forall X, Y. X \wedge Y \stackrel{*}{\doteq} \neg(\neg X \vee \neg Y)$
- $(\lambda U \lambda V. U \wedge V) \stackrel{*}{\doteq} (\lambda X \lambda Y. \neg(\neg X \vee \neg Y))$
- $\wedge \stackrel{*}{\doteq} (\lambda X \lambda Y. \neg(\neg X \vee \neg Y))$

requires $\mathfrak{b}$ and $\mathfrak{f}$

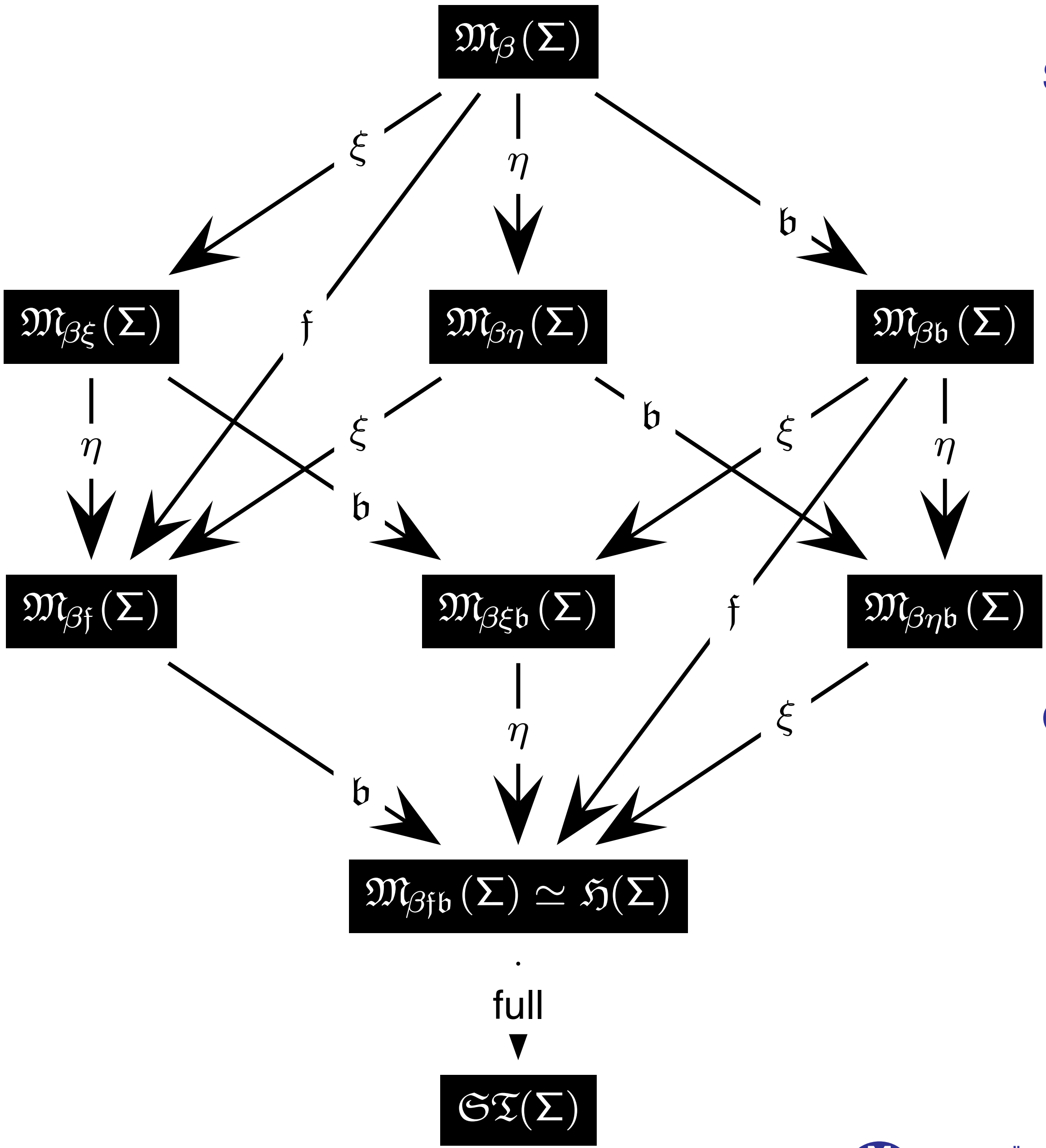
HOL-Problems: Set Comprehension



Set comprehension

- big challenge for automation
- [Benzm.BrownKohlhase-Draft-05] set instantiations can be used to simulate cut-rule if one of the following axioms is given: comprehension, induction, extensionality, choice, description
- dependend on logical constants in Σ

HOL-Problems: Set Comprehension



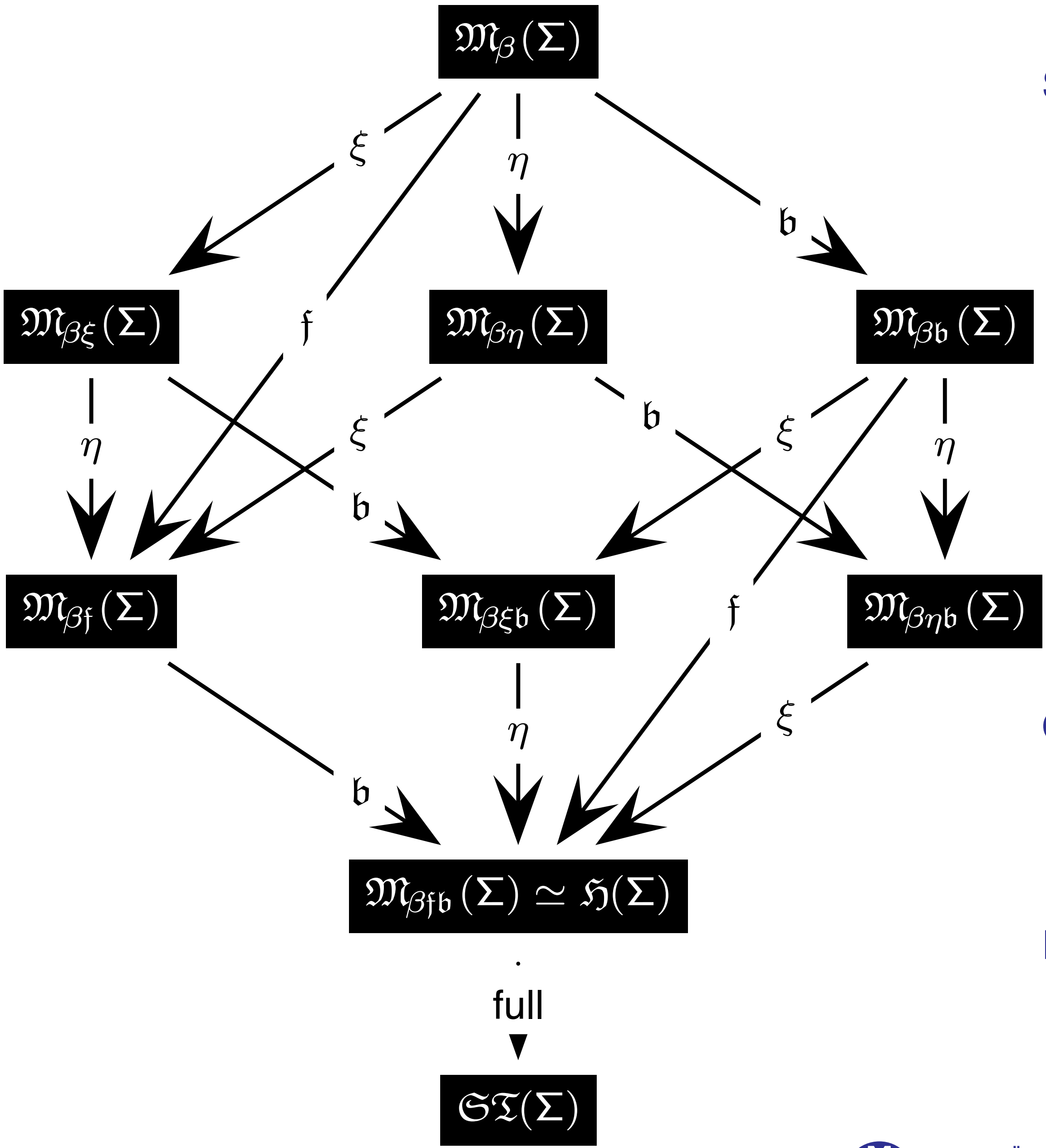
Set comprehension

- big challenge for automation
- [Benzm.BrownKohlhase-Draft-05] set instantiations can be used to simulate cut-rule if one of the following axioms is given: comprehension, induction, extensionality, choice, description
- dependend on logical constants in Σ

On the following slides emphasis on:

- signature Σ varying
- no property $\mathfrak{q}$ assumed

HOL-Problems: Set Comprehension



Set comprehension

- big challenge for automation
- [Benzm.BrownKohlhase-Draft-05] set instantiations can be used to simulate cut-rule if one of the following axioms is given: comprehension, induction, extensionality, choice, description
- dependend on logical constants in Σ

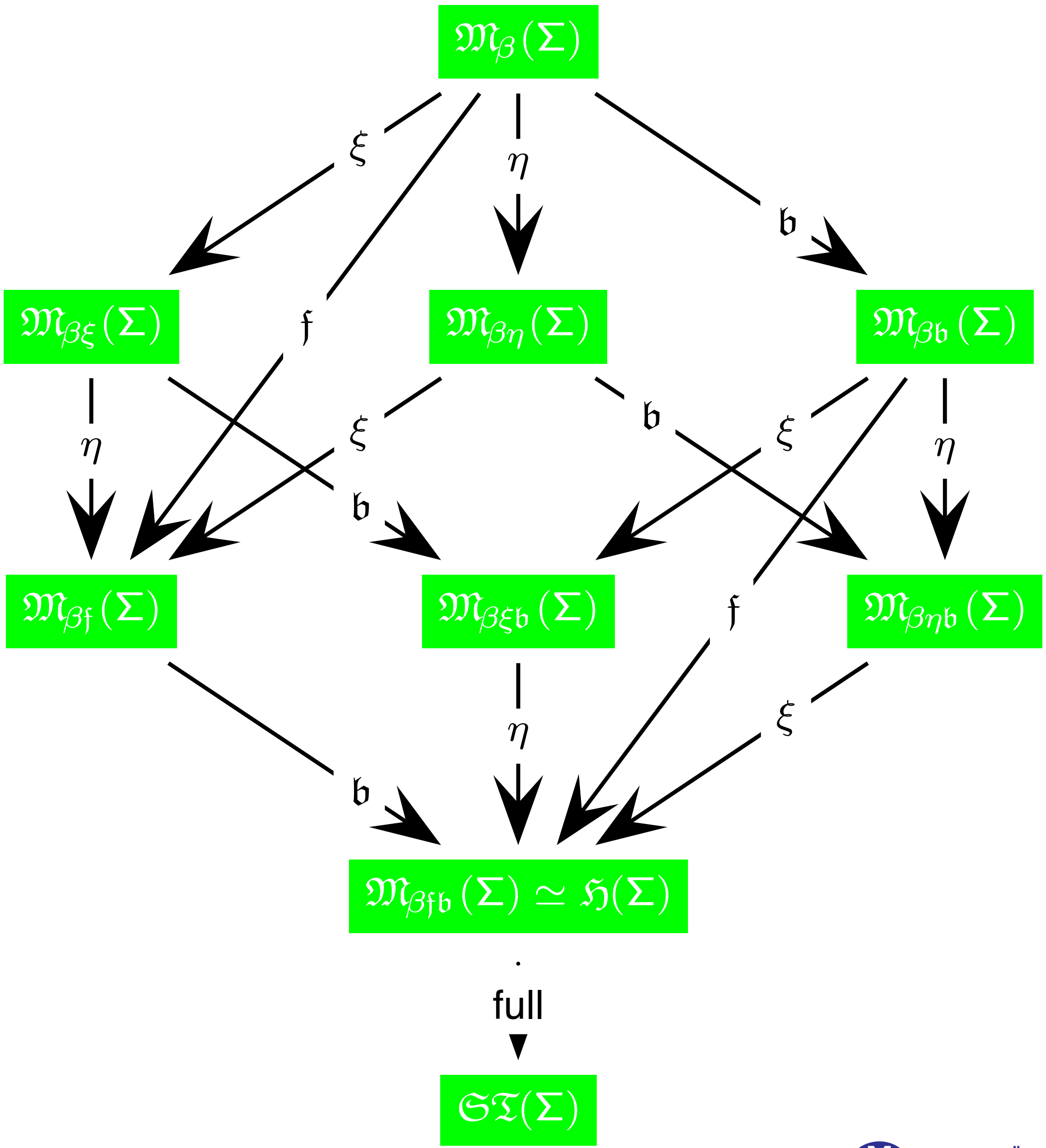
On the following slides emphasis on:

- signature Σ varying
- no property $\mathfrak{q}$ assumed

External vs. internal logical constants

- if $\neg \notin \Sigma$:
  $\neg$ refers to 'external' symbol
  $\mathcal{M} \models \neg A$ means $\mathcal{M} \not\models A$

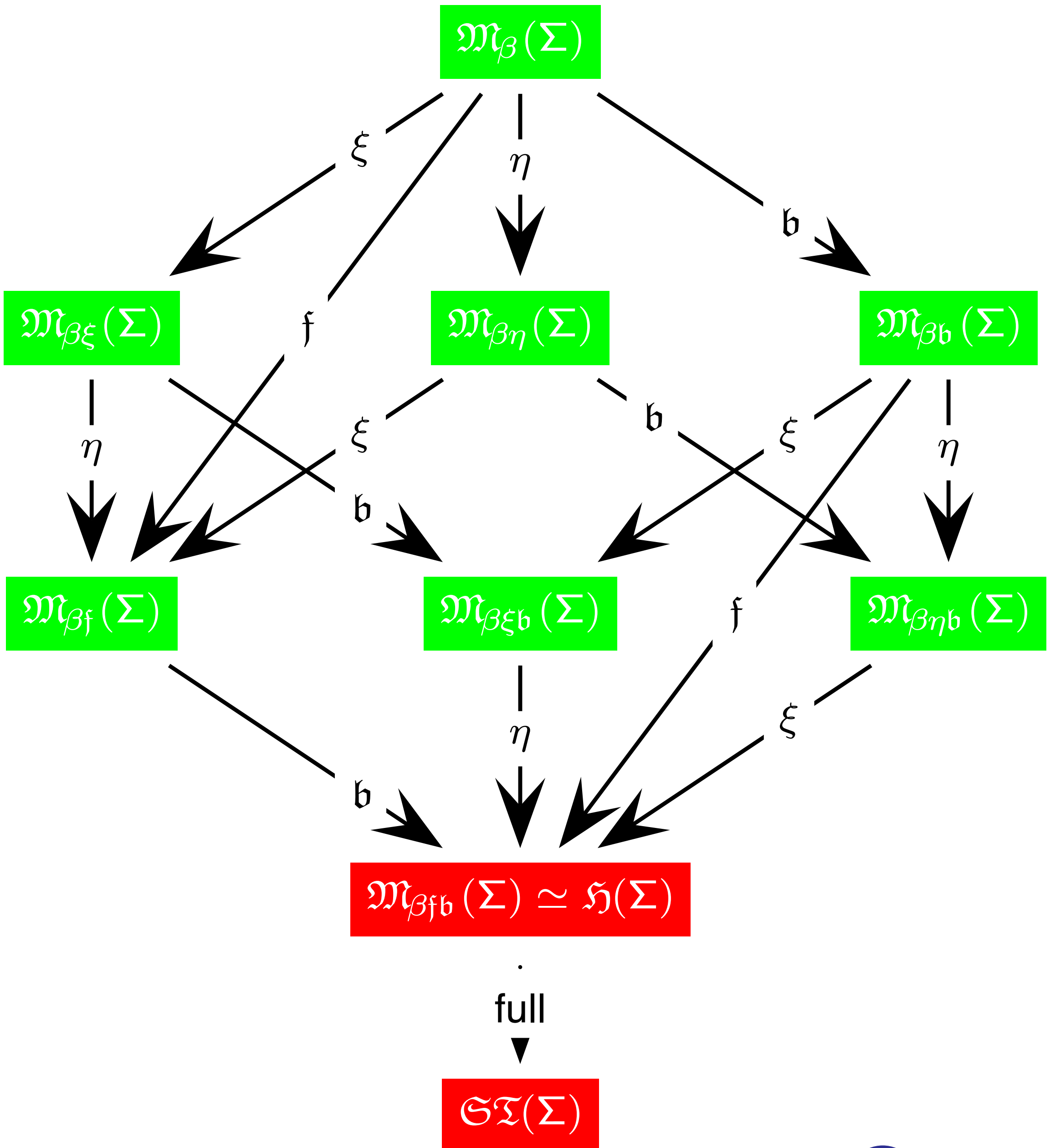
HOL-Problems: Set Comprehension



Set comprehension

- $\exists N_{oo} \forall P_o . NP \Leftrightarrow \neg P$
  - if $\neg \in \Sigma$ or $\{\bot, \supset\} \subseteq \Sigma$ or $\{\bot, \Leftrightarrow\} \subseteq \Sigma$
  - e.g.: $N_{oo} \longleftarrow \lambda X_o . \neg X$
    e.g.: $N_{oo} \longleftarrow \lambda X_o . X \supset \bot$

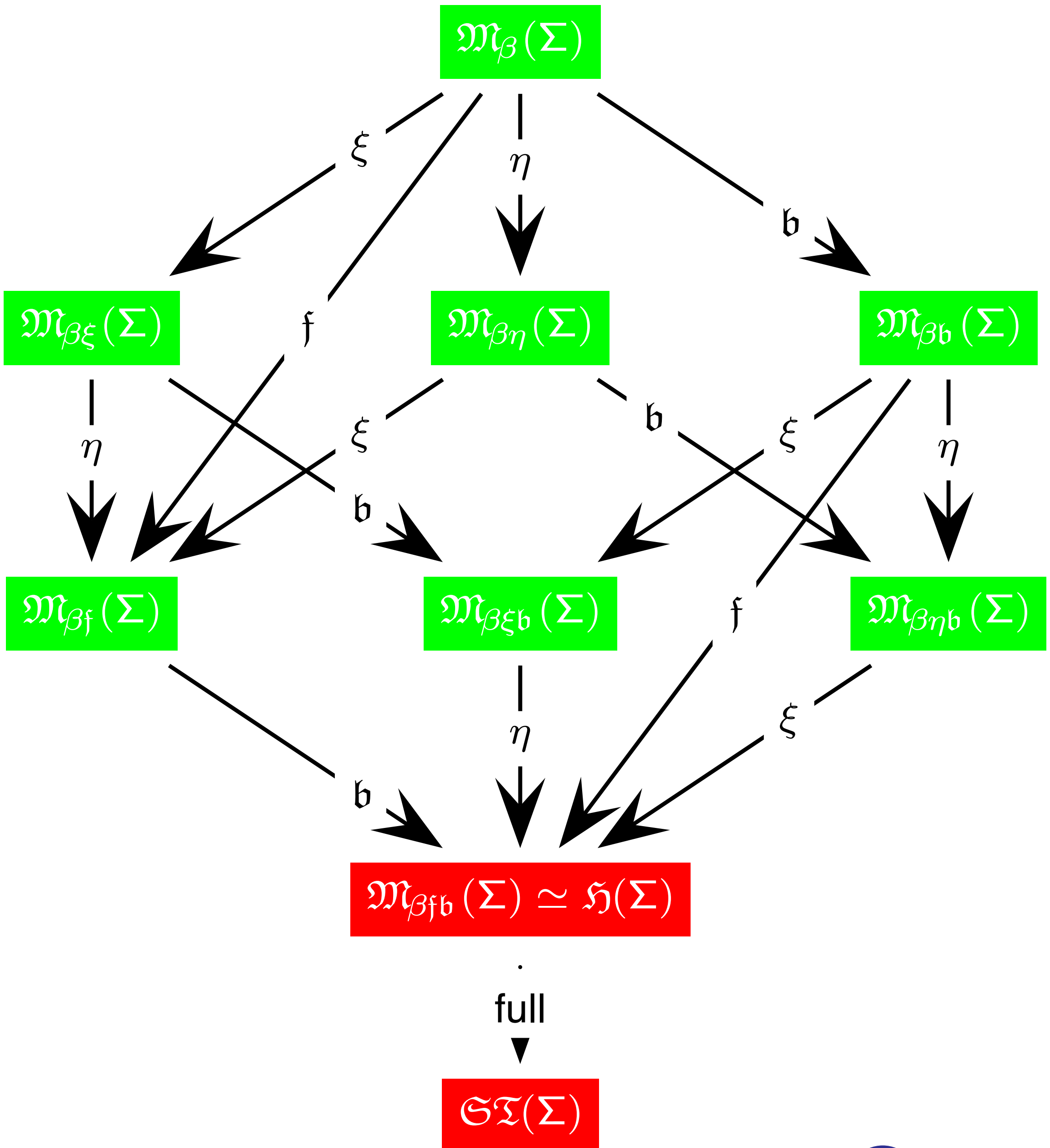
HOL-Problems: Set Comprehension



Set comprehension

- $\exists N_{oo} \forall P_o . N P \Leftrightarrow \neg P$
  - if $\neg \notin \Sigma$ and $\{\bot, \supset\} \not\subseteq \Sigma$ or $\{\bot, \Leftrightarrow\} \not\subseteq \Sigma$

HOL-Problems: Set Comprehension



Set comprehension

- $\exists N_{oo} \forall P_o . NP \Leftrightarrow \neg P$
  - if $\neg \notin \Sigma$ and $\{\bot, \supset\} \not\subseteq \Sigma$ or $\{\bot, \Leftrightarrow\} \not\subseteq \Sigma$

Other examples from [Brown-PhD-04]

- Surjective Cantor Theorem
- Injective Cantor Theorem

Semantics: Examples of

Σ-Models

Examples of Σ-Models

We now sketch the construction of models in the model classes $\mathfrak{M}_*(\Sigma)$ to demonstrate concretely how properties for Boolean, strong and weak functional extensionality can fail.

Examples of Σ-Models

We now sketch the construction of models in the model classes $\mathfrak{M}_*(\Sigma)$ to demonstrate concretely how properties for Boolean, strong and weak functional extensionality can fail.

We need this to show that the inclusions of the model classes in our landscape are proper, and we indeed need all of them.

Ex.: Singleton Model

- We choose $(\mathcal{D}, @)$ as the full frame with $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ and $\mathcal{D}_\iota := \{*\}$.

Ex.: Singleton Model

- We choose $(\mathcal{D}, @)$ as the full frame with $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ and $\mathcal{D}_\iota := \{*\}$.
- Easy to define an evaluation function $\mathcal{E}$ for this frame by induction on terms, using functions to interpret λ-abstractions.

Ex.: Singleton Model

- We choose $(\mathcal{D}, @)$ as the full frame with $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ and $\mathcal{D}_\iota := \{*\}$.
- Easy to define an evaluation function $\mathcal{E}$ for this frame by induction on terms, using functions to interpret λ-abstractions.
- The identity function $\upsilon : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ is a valuation, assuming the logical constants are interpreted in the standard way.

Ex.: Singleton Model

- We choose $(\mathcal{D}, @)$ as the full frame with $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ and $\mathcal{D}_\iota := \{*\}$.
- Easy to define an evaluation function $\mathcal{E}$ for this frame by induction on terms, using functions to interpret λ-abstractions.
- The identity function $\upsilon : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ is a valuation, assuming the logical constants are interpreted in the standard way.
- Thus, $\mathcal{M}^{\beta\mathfrak{fb}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ defines a Σ-model.

Ex.: Singleton Model

- We choose $(\mathcal{D}, @)$ as the full frame with $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ and $\mathcal{D}_\iota := \{*\}$.
- Easy to define an evaluation function $\mathcal{E}$ for this frame by induction on terms, using functions to interpret λ-abstractions.
- The identity function $\upsilon \colon \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ is a valuation, assuming the logical constants are interpreted in the standard way.
- Thus, $\mathcal{M}^{\beta\mathfrak{f}\mathfrak{b}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ defines a Σ-model.
- This model satisfies properties $\mathfrak{b}$, $\mathfrak{f}$ (hence η and ξ) and $\mathfrak{q}$ (since the frame is full).

Ex.: Singleton Model

- We choose $(\mathcal{D}, @)$ as the full frame with $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ and $\mathcal{D}_\iota := \{*\}$.
- Easy to define an evaluation function $\mathcal{E}$ for this frame by induction on terms, using functions to interpret λ-abstractions.
- The identity function $\upsilon : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ is a valuation, assuming the logical constants are interpreted in the standard way.
- Thus, $\mathcal{M}^{\beta\mathfrak{fb}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ defines a Σ-model.
- This model satisfies properties $\mathfrak{b}$, $\mathfrak{f}$ (hence η and ξ) and $\mathfrak{q}$ (since the frame is full).
- So, $\mathcal{M}^{\beta\mathfrak{fb}} \in \mathfrak{ST}(\Sigma) \subseteq \mathfrak{H}(\Sigma) \subseteq \mathfrak{M}_{\beta\mathfrak{fb}}(\Sigma) \subseteq \ldots$.

UNIVERSITÄT DES SAARLANDES

Ex.: Singleton Model

$$
\begin{tikzcd}
 & \mathfrak{M}_{\beta}(\Sigma) \arrow[dl, "\xi"'] \arrow[d, "\eta"] \arrow[dr, "\mathfrak{b}"] \arrow[ddl, "\mathfrak{f}"] & \\
\mathfrak{M}_{\beta\xi}(\Sigma) \arrow[d, "\eta"] \arrow[dr, "\mathfrak{b}"] & \mathfrak{M}_{\beta\eta}(\Sigma) \arrow[dl, "\xi"'] \arrow[dr, "\mathfrak{b}"] & \mathfrak{M}_{\beta\mathfrak{b}}(\Sigma) \arrow[dl, "\xi"'] \arrow[d, "\eta"] \arrow[ddl, "\mathfrak{f}"] \\
\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma) \arrow[dr, "\mathfrak{b}"] & \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma) \arrow[d, "\eta"] & \mathfrak{M}_{\beta\eta\mathfrak{b}}(\Sigma) \arrow[dl, "\xi"'] \\
 & \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma) \simeq \mathfrak{H}(\Sigma) \arrow[d, "\text{full}"] & \\
 & \mathfrak{S}\mathfrak{T}(\Sigma) &
\end{tikzcd}
$$

Ex.: Model without Boolean Extensionality

- Assume Σ contains only the connectives $\neg, \vee, \Pi^\alpha$; other connectives defined as usual, e.g., $\forall X, Y. X \wedge Y \Leftrightarrow \neg(\neg X \vee \neg Y)$.

Ex.: Model without Boolean Extensionality

- Assume Σ contains only the connectives $\neg, \vee, \Pi^\alpha$; other connectives defined as usual, e.g., $\forall X, Y. X \wedge Y \Leftrightarrow \neg(\neg X \vee \neg Y)$.
- Choose $(\mathcal{D}, @)$ as full frame with $\mathcal{D}_o = \{a, b, c\}$ and $\mathcal{D}_\iota = \{0, 1\}$.

Ex.: Model without Boolean Extensionality

- Assume Σ contains only the connectives $\neg, \vee, \Pi^\alpha$; other connectives defined as usual, e.g., $\forall X, Y. X \wedge Y \Leftrightarrow \neg(\neg X \vee \neg Y)$.
- Choose $(\mathcal{D}, @)$ as full frame with $\mathcal{D}_o = \{a, b, c\}$ and $\mathcal{D}_\iota = \{0, 1\}$.
- We define evaluation function $\mathcal{E}$ for this frame by defining $\mathcal{E}(\neg)$, $\mathcal{E}(\vee)$, and $\mathcal{E}(\Pi^\alpha)$:

$\mathcal{E}(\neg)$	a	b	c
	c	c	a

$\mathcal{E}(\vee)$	a	b	c
a	a	a	a
b	a	a	a
c	a	a	c

$$\mathcal{E}(\Pi^\alpha)@f = \begin{cases} a, & \text{if } f@g \in \{a, b\} \text{ for all } g \in \mathcal{D}_\alpha \\ c, & \text{if } f@g = c \text{ for some } g \in \mathcal{D}_\alpha \end{cases}$$

Ex.: Model without Boolean Extensionality

- Assume Σ contains only the connectives $\neg, \vee, \Pi^\alpha$; other connectives defined as usual, e.g., $\forall X, Y.X \wedge Y \Leftrightarrow \neg(\neg X \vee \neg Y)$.
- Choose $(\mathcal{D}, @)$ as full frame with $\mathcal{D}_o = \{a, b, c\}$ and $\mathcal{D}_\iota = \{0, 1\}$.
- We define evaluation function $\mathcal{E}$ for this frame by defining $\mathcal{E}(\neg)$, $\mathcal{E}(\vee)$, and $\mathcal{E}(\Pi^\alpha)$:

$\mathcal{E}(\neg)$	a	b	c
	c	c	a

$\mathcal{E}(\vee)$	a	b	c
a	a	a	a
b	a	a	a
c	a	a	c

$$\mathcal{E}(\Pi^\alpha)@f = \begin{cases} a, & \text{if } f@g \in \{a, b\} \text{ for all } g \in \mathcal{D}_\alpha \\ c, & \text{if } f@g = c \text{ for some } g \in \mathcal{D}_\alpha \end{cases}$$

- We can choose $\mathcal{E}(w)$ to be arbitrary for parameters $w \in \Sigma$.

Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.

Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.
- Since the frame is full, we are guaranteed that there will be enough functions to interpret λ-abstractions.

Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.
- Since the frame is full, we are guaranteed that there will be enough functions to interpret λ-abstractions.
- Let $\upsilon : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be defined by $\upsilon(\mathsf{a}) := \mathtt{T}$, $\upsilon(\mathsf{b}) := \mathtt{T}$ and $\upsilon(\mathsf{c}) := \mathtt{F}$.

Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.
- Since the frame is full, we are guaranteed that there will be enough functions to interpret λ-abstractions.
- Let $\upsilon\colon \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be defined by $\upsilon(\mathsf{a}) := \mathtt{T}$, $\upsilon(\mathsf{b}) := \mathtt{T}$ and $\upsilon(\mathsf{c}) := \mathtt{F}$.
- Easy to check that $\mathcal{M}^{\beta\mathfrak{f}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is indeed a Σ-model.

Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.
- Since the frame is full, we are guaranteed that there will be enough functions to interpret λ-abstractions.
- Let $\upsilon : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be defined by $\upsilon(\mathsf{a}) := \mathtt{T}$, $\upsilon(\mathsf{b}) := \mathtt{T}$ and $\upsilon(\mathsf{c}) := \mathtt{F}$.
- Easy to check that $\mathcal{M}^{\beta\mathfrak{f}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is indeed a Σ-model.
- Since $\mathcal{M}^{\beta\mathfrak{f}}$ is a model over a frame it satisfies property $\mathfrak{f}$.

Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.
- Since the frame is full, we are guaranteed that there will be enough functions to interpret λ-abstractions.
- Let $\upsilon \colon \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be defined by $\upsilon(\mathsf{a}) := \mathtt{T}$, $\upsilon(\mathsf{b}) := \mathtt{T}$ and $\upsilon(\mathsf{c}) := \mathtt{F}$.
- Easy to check that $\mathcal{M}^{\beta\mathfrak{f}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is indeed a Σ-model.
- Since $\mathcal{M}^{\beta\mathfrak{f}}$ is a model over a frame it satisfies property $\mathfrak{f}$.
- Since this frame is full, we know property $\mathfrak{q}$ holds.

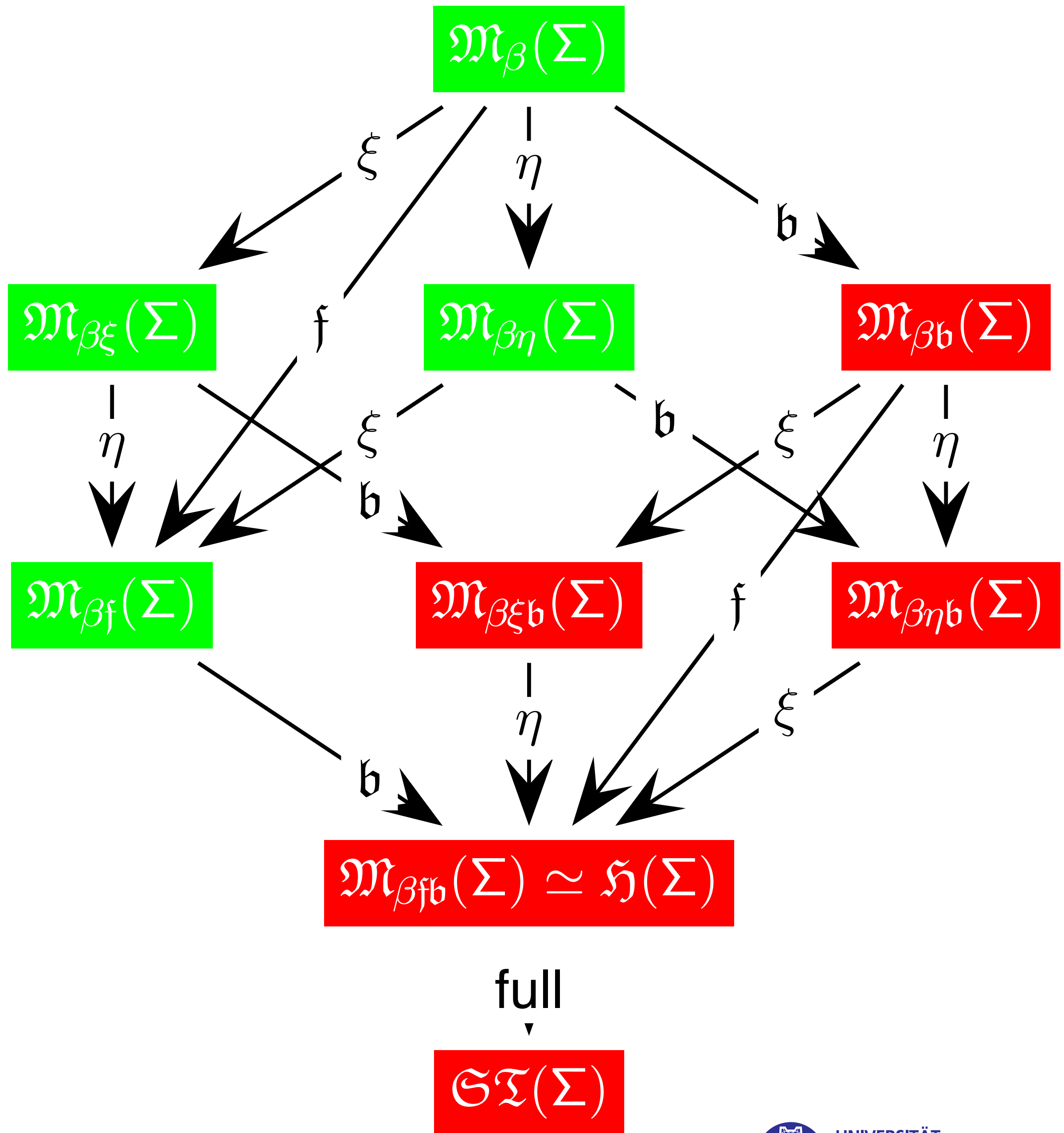
Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.
- Since the frame is full, we are guaranteed that there will be enough functions to interpret λ-abstractions.
- Let $\upsilon\colon \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be defined by $\upsilon(\mathsf{a}) := \mathtt{T}$, $\upsilon(\mathsf{b}) := \mathtt{T}$ and $\upsilon(\mathsf{c}) := \mathtt{F}$.
- Easy to check that $\mathcal{M}^{\beta\mathfrak{f}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is indeed a Σ-model.
- Since $\mathcal{M}^{\beta\mathfrak{f}}$ is a model over a frame it satisfies property $\mathfrak{f}$.
- Since this frame is full, we know property $\mathfrak{q}$ holds.
- Clearly property $\mathfrak{b}$ fails.

Ex.: Model without Boolean Extensionality

- Since $(\mathcal{D}, @)$ is a frame, hence functional, this uniquely determines $\mathcal{E}$ on all formulae.
- Since the frame is full, we are guaranteed that there will be enough functions to interpret λ-abstractions.
- Let $\upsilon : \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be defined by $\upsilon(\mathsf{a}) := \mathtt{T}$, $\upsilon(\mathsf{b}) := \mathtt{T}$ and $\upsilon(\mathsf{c}) := \mathtt{F}$.
- Easy to check that $\mathcal{M}^{\beta\mathfrak{f}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is indeed a Σ-model.
- Since $\mathcal{M}^{\beta\mathfrak{f}}$ is a model over a frame it satisfies property $\mathfrak{f}$.
- Since this frame is full, we know property $\mathfrak{q}$ holds.
- Clearly property $\mathfrak{b}$ fails.
- So, $\mathcal{M}^{\beta\mathfrak{f}} \in \mathfrak{M}_{\beta\mathfrak{f}}(\Sigma) \setminus \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma)$.

Ex.: Model without Boolean Extensionality



Ex.: Model without Boolean Extensionality

In the previous model one can easily verify, if $d := \mathcal{E}_\varphi(\mathbf{D}_o)$ and $e := \mathcal{E}_\varphi(\mathbf{E}_o)$, then the values $\mathcal{E}_\varphi(\mathbf{D} \land \mathbf{E})$, $\mathcal{E}_\varphi(\mathbf{D} \Rightarrow \mathbf{E})$, and $\mathcal{E}_\varphi(\mathbf{D} \Leftrightarrow \mathbf{E})$ are given by the following tables:

		e:		
$\mathcal{E}(\mathbf{D} \land \mathbf{E})$		a	b	c
d:	a	a	a	c
	b	a	a	c
	c	c	c	c

		e:		
$\mathcal{E}(\mathbf{D} \Rightarrow \mathbf{E})$		a	b	c
d:	a	a	a	c
	b	a	a	c
	c	a	a	a

		e:		
$\mathcal{E}(\mathbf{D} \Leftrightarrow \mathbf{E})$		a	b	c
d:	a	a	a	c
	b	a	a	c
	c	c	c	a

Now we show that one can properly model the woodchuck/groundhog example.

Ex.: Groundhogs and Woodchucks

- Let $\mathcal{M}^{\beta\mathfrak{f}}$ be given as above and suppose $\mathsf{woodchuck}_{\iota\rightarrow o}$, $\mathsf{groundhog}_{\iota\rightarrow o}$, john_{ι}, and phil_{ι} are in the signature Σ.

Ex.: Groundhogs and Woodchucks

- Let $\mathcal{M}^{\beta\mathfrak{f}}$ be given as above and suppose $\mathsf{woodchuck}_{\iota\to o}$, $\mathsf{groundhog}_{\iota\to o}$, john_{ι}, and phil_{ι} are in the signature Σ.
- Let $\mathcal{E}(\mathsf{phil}) := 0$ and $\mathcal{E}(\mathsf{john}) := 1$.

Ex.: Groundhogs and Woodchucks

- Let $\mathcal{M}^{\beta\mathfrak{f}}$ be given as above and suppose $\mathsf{woodchuck}_{\iota\rightarrow o}$, $\mathsf{groundhog}_{\iota\rightarrow o}$, john_{ι}, and phil_{ι} are in the signature Σ.
- Let $\mathcal{E}(\mathsf{phil}) := 0$ and $\mathcal{E}(\mathsf{john}) := 1$.
- Let $\mathcal{E}(\mathsf{woodchuck})$ be the function $\mathsf{w} \in \mathcal{D}_{\iota\rightarrow o}$ with $\mathsf{w}(0) = \mathsf{b}$ and $\mathsf{w}(1) = \mathsf{c}$.

Ex.: Groundhogs and Woodchucks

- Let $\mathcal{M}^{\beta\mathfrak{f}}$ be given as above and suppose $\mathsf{woodchuck}_{\iota\to o}$, $\mathsf{groundhog}_{\iota\to o}$, john_{ι}, and phil_{ι} are in the signature Σ.
- Let $\mathcal{E}(\mathsf{phil}) := 0$ and $\mathcal{E}(\mathsf{john}) := 1$.
- Let $\mathcal{E}(\mathsf{woodchuck})$ be the function $\mathsf{w} \in \mathcal{D}_{\iota\to o}$ with $\mathsf{w}(0) = \mathsf{b}$ and $\mathsf{w}(1) = \mathsf{c}$.
- Let $\mathcal{E}(\mathsf{groundhog})$ be the function $\mathsf{g} \in \mathcal{D}_{\iota\to o}$ with $\mathsf{g}(0) = \mathsf{a}$ and $\mathsf{g}(1) = \mathsf{c}$.

Ex.: Groundhogs and Woodchucks

- Let $\mathcal{M}^{\beta\mathfrak{f}}$ be given as above and suppose $\mathsf{woodchuck}_{\iota\rightarrow o}$, $\mathsf{groundhog}_{\iota\rightarrow o}$, john_ι, and phil_ι are in the signature Σ.
- Let $\mathcal{E}(\mathsf{phil}) := 0$ and $\mathcal{E}(\mathsf{john}) := 1$.
- Let $\mathcal{E}(\mathsf{woodchuck})$ be the function $\mathsf{w} \in \mathcal{D}_{\iota\rightarrow o}$ with $\mathsf{w}(0) = \mathsf{b}$ and $\mathsf{w}(1) = \mathsf{c}$.
- Let $\mathcal{E}(\mathsf{groundhog})$ be the function $\mathsf{g} \in \mathcal{D}_{\iota\rightarrow o}$ with $\mathsf{g}(0) = \mathsf{a}$ and $\mathsf{g}(1) = \mathsf{c}$.
- One can show that the sentence $\forall X_\iota.(\mathsf{woodchuck}\ X) \Leftrightarrow (\mathsf{groundhog}\ X)$ is valid.

Ex.: Groundhogs and Woodchucks

- Let $\mathcal{M}^{\beta\mathfrak{f}}$ be given as above and suppose $\mathsf{woodchuck}_{\iota\to o}$, $\mathsf{groundhog}_{\iota\to o}$, john_ι, and phil_ι are in the signature Σ.
- Let $\mathcal{E}(\mathsf{phil}) := 0$ and $\mathcal{E}(\mathsf{john}) := 1$.
- Let $\mathcal{E}(\mathsf{woodchuck})$ be the function $\mathsf{w} \in \mathcal{D}_{\iota\to o}$ with $\mathsf{w}(0) = \mathsf{b}$ and $\mathsf{w}(1) = \mathsf{c}$.
- Let $\mathcal{E}(\mathsf{groundhog})$ be the function $\mathsf{g} \in \mathcal{D}_{\iota\to o}$ with $\mathsf{g}(0) = \mathsf{a}$ and $\mathsf{g}(1) = \mathsf{c}$.
- One can show that the sentence $\forall X_\iota.(\mathsf{woodchuck}\ X) \Leftrightarrow (\mathsf{groundhog}\ X)$ is valid.
- Also, $\mathcal{E}(\mathsf{woodchuck\ phil}) = \mathsf{b}$ and $\mathcal{E}(\mathsf{groundhog\ phil}) = \mathsf{a}$, so the propositions $(\mathsf{woodchuck\ phil})$ and $(\mathsf{groundhog\ phil})$ are valid.

Ex.: Groundhogs and Woodchucks

- Suppose $\mathsf{believe}_{\iota\to o\to o} \in \Sigma$ and $\mathcal{E}(\mathsf{believe})$ is the (Curried) function $\mathsf{bel} \in \mathcal{D}_{\iota\to o\to o}$ such that $\mathsf{bel}(1)(\mathsf{b}) = \mathsf{b}$ and $\mathsf{bel}(1)(\mathsf{a}) = \mathsf{bel}(1)(\mathsf{c}) = \mathsf{bel}(0)(\mathsf{a}) = \mathsf{bel}(0)(\mathsf{b}) = \mathsf{bel}(0)(\mathsf{c}) = \mathsf{c}$.

Ex.: Groundhogs and Woodchucks

- Suppose $\mathsf{believe}_{\iota\rightarrow o\rightarrow o} \in \Sigma$ and $\mathcal{E}(\mathsf{believe})$ is the (Curried) function $\mathsf{bel} \in \mathcal{D}_{\iota\rightarrow o\rightarrow o}$ such that $\mathsf{bel}(1)(\mathsf{b}) = \mathsf{b}$ and $\mathsf{bel}(1)(\mathsf{a}) = \mathsf{bel}(1)(\mathsf{c}) = \mathsf{bel}(0)(\mathsf{a}) = \mathsf{bel}(0)(\mathsf{b}) = \mathsf{bel}(0)(\mathsf{c}) = \mathsf{c}$.
- Intuitively, John believes propositions with value b, but not those with value a or c.

Ex.: Groundhogs and Woodchucks

- Suppose $\mathsf{believe}_{\iota\to o\to o} \in \Sigma$ and $\mathcal{E}(\mathsf{believe})$ is the (Curried) function $\mathsf{bel} \in \mathcal{D}_{\iota\to o\to o}$ such that $\mathsf{bel}(1)(\mathsf{b}) = \mathsf{b}$ and $\mathsf{bel}(1)(\mathsf{a}) = \mathsf{bel}(1)(\mathsf{c}) = \mathsf{bel}(0)(\mathsf{a}) = \mathsf{bel}(0)(\mathsf{b}) = \mathsf{bel}(0)(\mathsf{c}) = \mathsf{c}$.
- Intuitively, John believes propositions with value b, but not those with value a or c.
- So, $\mathsf{believes\ john(woodchuck\ phil)}$ is valid, while $\mathsf{believes\ john(groundhog\ phil)}$ is not.

Generalizing the Previous Model

As we have seen, Boolean extensionality fails when one has more than two values in $\mathcal{D}_o$.

Generalizing the Previous Model

As we have seen, Boolean extensionality fails when one has more than two values in $\mathcal{D}_o$. We can generalize the construction defining $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$, where $\mathcal{B}$ is any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$.

Generalizing the Previous Model

As we have seen, Boolean extensionality fails when one has more than two values in $\mathcal{D}_o$. We can generalize the construction defining $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$, where $\mathcal{B}$ is any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$. The model will satisfy Boolean extensionality iff $\mathcal{B} = \{\mathtt{T}\}$.

Generalizing the Previous Model

As we have seen, Boolean extensionality fails when one has more than two values in $\mathcal{D}_o$. We can generalize the construction defining $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$, where $\mathcal{B}$ is any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$. The model will satisfy Boolean extensionality iff $\mathcal{B} = \{\mathtt{T}\}$. In this way, we can easily construct models for the case with property $\mathfrak{b}$ and the case without property $\mathfrak{b}$ simultaneously.

Generalizing the Previous Model

As we have seen, Boolean extensionality fails when one has more than two values in $\mathcal{D}_o$. We can generalize the construction defining $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$, where $\mathcal{B}$ is any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$. The model will satisfy Boolean extensionality iff $\mathcal{B} = \{\mathtt{T}\}$. In this way, we can easily construct models for the case with property $\mathfrak{b}$ and the case without property $\mathfrak{b}$ simultaneously. We will use this idea to parameterize the remaining model constructions by $\mathcal{B}$.

Generalizing the Previous Model

As we have seen, Boolean extensionality fails when one has more than two values in $\mathcal{D}_o$. We can generalize the construction defining $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$, where $\mathcal{B}$ is any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$. The model will satisfy Boolean extensionality iff $\mathcal{B} = \{\mathtt{T}\}$. In this way, we can easily construct models for the case with property $\mathfrak{b}$ and the case without property $\mathfrak{b}$ simultaneously. We will use this idea to parameterize the remaining model constructions by $\mathcal{B}$.

These semantic constructions are similar to those in multi-valued logics.

Generalizing the Previous Model

As we have seen, Boolean extensionality fails when one has more than two values in $\mathcal{D}_o$. We can generalize the construction defining $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$, where $\mathcal{B}$ is any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$. The model will satisfy Boolean extensionality iff $\mathcal{B} = \{\mathtt{T}\}$. In this way, we can easily construct models for the case with property $\mathfrak{b}$ and the case without property $\mathfrak{b}$ simultaneously. We will use this idea to parameterize the remaining model constructions by $\mathcal{B}$.

These semantic constructions are similar to those in multi-valued logics. In contrast to these logics where the logical connectives are adapted to talk about multiple truth values, in our setting we are mainly interested in multiple truth values as diverse υ-pre-images of $\mathtt{T}$ and $\mathtt{F}$.

Semantics: Examples of Σ-Models (Contd.)

Ex.: Models without Funct. Extensionality

- Idea: attach distinguishing labels to functions without changing their applicative behavior

Ex.: Models without Funct. Extensionality

- Idea: attach distinguishing labels to functions without changing their applicative behavior
- Let $\mathcal{B}$ be any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$

Ex.: Models without Funct. Extensionality

- Idea: attach distinguishing labels to functions without changing their applicative behavior
- Let $\mathcal{B}$ be any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$
- Let $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$ and $\mathcal{D}_\iota := \{*\}$

Ex.: Models without Funct. Extensionality

- Idea: attach distinguishing labels to functions without changing their applicative behavior
- Let $\mathcal{B}$ be any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$
- Let $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$ and $\mathcal{D}_\iota := \{*\}$
- For each function type $\beta\alpha$, let

$$\mathcal{D}_{\beta\alpha} := \{(\mathsf{i}, \mathsf{f}) \mid \mathsf{i} \in \{0, 1\} \text{ and } \mathsf{f} \colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta\}$$

Ex.: Models without Funct. Extensionality

- Idea: attach distinguishing labels to functions without changing their applicative behavior
- Let $\mathcal{B}$ be any set with $\mathtt{T} \in \mathcal{B}$ and $\mathtt{F} \notin \mathcal{B}$
- Let $\mathcal{D}_o := \{\mathtt{F}\} \cup \mathcal{B}$ and $\mathcal{D}_\iota := \{*\}$
- For each function type $\beta\alpha$, let

$$\mathcal{D}_{\beta\alpha} := \{(\mathsf{i}, \mathsf{f}) \mid \mathsf{i} \in \{0, 1\} \text{ and } \mathsf{f}\colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta\}$$

- We define application by

$$(\mathsf{i}, \mathsf{f})@\mathsf{a} := \mathsf{f}(\mathsf{a})$$

whenever $(\mathsf{i}, \mathsf{f}) \in \mathcal{D}_{\beta\alpha}$ and $\mathsf{a} \in \mathcal{D}_\alpha$

Ex.: Models without η and $\mathfrak{f}$

- Easy to check that $(\mathcal{D}, @)$ is an applicative structure:

Ex.: Models without η and $\mathfrak{f}$

- Easy to check that $(\mathcal{D}, @)$ is an applicative structure:
- Evaluation function defined by induction on terms

Ex.: Models without η and $\mathfrak{f}$

- Easy to check that $(\mathcal{D}, @)$ is an applicative structure:
- Evaluation function defined by induction on terms
  - $\mathcal{E}(\neg) := (0, \mathsf{n})$ where $\mathsf{n}(\mathsf{b}) := \mathtt{F}$ for every $\mathsf{b} \in \mathcal{B}$ and $\mathsf{n}(\mathtt{F}) := \mathtt{T}$

Ex.: Models without η and $\mathfrak{f}$

- Easy to check that $(\mathcal{D}, @)$ is an applicative structure:
- Evaluation function defined by induction on terms
  - $\mathcal{E}(\neg) := (0, \mathsf{n})$ where $\mathsf{n}(\mathsf{b}) := \mathtt{F}$ for every $\mathsf{b} \in \mathcal{B}$ and $\mathsf{n}(\mathtt{F}) := \mathtt{T}$
  - $\mathcal{E}(\vee) := (0, \mathsf{d})$ where
    $\mathsf{d}(\mathsf{b}) := (0, \mathsf{k}^{\mathtt{T}})$ for every $\mathsf{b} \in \mathcal{B}$ and
    $\mathsf{d}(\mathtt{F}) := (0, \mathsf{id})$
    ($\mathsf{k}^{\mathtt{T}}$ is the constant $\mathtt{T}$ function)
    (id is the identity function from $\mathcal{D}_o$ to $\mathcal{D}_o$)

Ex.: Models without η and $\mathfrak{f}$

- Easy to check that $(\mathcal{D}, @)$ is an applicative structure:
- Evaluation function defined by induction on terms
  - $\mathcal{E}(\neg) := (0, \mathsf{n})$ where $\mathsf{n}(\mathsf{b}) := \mathtt{F}$ for every $\mathsf{b} \in \mathcal{B}$ and $\mathsf{n}(\mathtt{F}) := \mathtt{T}$
  - $\mathcal{E}(\vee) := (0, \mathsf{d})$ where
    $\mathsf{d}(\mathsf{b}) := (0, \mathsf{k}^{\mathtt{T}})$ for every $\mathsf{b} \in \mathcal{B}$ and
    $\mathsf{d}(\mathtt{F}) := (0, \mathsf{id})$
    ($\mathsf{k}^{\mathtt{T}}$ is the constant $\mathtt{T}$ function)
    (id is the identity function from $\mathcal{D}_o$ to $\mathcal{D}_o$)
  - $\mathcal{E}(\Pi^\alpha) := (0, \pi^\alpha)$ where for each $(\mathsf{i}, \mathsf{f}) \in \mathcal{D}_{o\alpha}$, $\pi^\alpha((\mathsf{i}, \mathsf{f})) := \mathtt{T}$ if $\mathsf{f}(\mathsf{a}) \in \mathcal{B}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$ and $\pi^\alpha(\mathsf{i}, \mathsf{f}) := \mathtt{F}$ otherwise

Ex.: Models without η and $\mathfrak{f}$

- Easy to check that $(\mathcal{D}, @)$ is an applicative structure:
- Evaluation function defined by induction on terms
  - $\mathcal{E}(\neg) := (0, \mathsf{n})$ where $\mathsf{n}(\mathsf{b}) := \mathtt{F}$ for every $\mathsf{b} \in \mathcal{B}$ and $\mathsf{n}(\mathtt{F}) := \mathtt{T}$
  - $\mathcal{E}(\vee) := (0, \mathsf{d})$ where
    $\mathsf{d}(\mathsf{b}) := (0, \mathsf{k}^{\mathtt{T}})$ for every $\mathsf{b} \in \mathcal{B}$ and
    $\mathsf{d}(\mathtt{F}) := (0, \mathsf{id})$
    ($\mathsf{k}^{\mathtt{T}}$ is the constant $\mathtt{T}$ function)
    (id is the identity function from $\mathcal{D}_{\mathsf{o}}$ to $\mathcal{D}_{\mathsf{o}}$)
  - $\mathcal{E}(\Pi^{\alpha}) := (0, \pi^{\alpha})$ where for each $(\mathsf{i}, \mathsf{f}) \in \mathcal{D}_{\mathsf{o}\alpha}$, $\pi^{\alpha}((\mathsf{i}, \mathsf{f})) := \mathtt{T}$ if $\mathsf{f}(\mathsf{a}) \in \mathcal{B}$ for all $\mathsf{a} \in \mathcal{D}_{\alpha}$ and $\pi^{\alpha}(\mathsf{i}, \mathsf{f}) := \mathtt{F}$ otherwise
  - $\mathsf{q}^{\alpha} := (0, \mathsf{q}^{\alpha}) \in \mathcal{D}_{\mathsf{o}\alpha\alpha}$ where $\mathsf{q}^{\alpha}(\mathsf{a}) := (0, \mathsf{s}^{\mathsf{a}})$ and $\mathsf{s}^{\mathsf{a}}(\mathsf{b}) := \mathtt{T}$ if $\mathsf{a} = \mathsf{b}$ and $\mathsf{s}^{\mathsf{a}}(\mathsf{b}) := \mathtt{F}$ otherwise

Ex.: Models without η and $\mathfrak{f}$

- Easy to check that $(\mathcal{D}, @)$ is an applicative structure:
- Evaluation function defined by induction on terms
  - $\mathcal{E}(\neg) := (0, \mathsf{n})$ where $\mathsf{n}(\mathsf{b}) := \mathtt{F}$ for every $\mathsf{b} \in \mathcal{B}$ and $\mathsf{n}(\mathtt{F}) := \mathtt{T}$
  - $\mathcal{E}(\vee) := (0, \mathsf{d})$ where
    $\mathsf{d}(\mathsf{b}) := (0, \mathsf{k}^{\mathtt{T}})$ for every $\mathsf{b} \in \mathcal{B}$ and
    $\mathsf{d}(\mathtt{F}) := (0, \mathsf{id})$
    ($\mathsf{k}^{\mathtt{T}}$ is the constant $\mathtt{T}$ function)
    (id is the identity function from $\mathcal{D}_{\mathsf{o}}$ to $\mathcal{D}_{\mathsf{o}}$)
  - $\mathcal{E}(\Pi^{\alpha}) := (0, \pi^{\alpha})$ where for each $(\mathsf{i}, \mathsf{f}) \in \mathcal{D}_{\mathsf{o}\alpha}$, $\pi^{\alpha}((\mathsf{i}, \mathsf{f})) := \mathtt{T}$ if $\mathsf{f}(\mathsf{a}) \in \mathcal{B}$ for all $\mathsf{a} \in \mathcal{D}_{\alpha}$ and $\pi^{\alpha}(\mathsf{i}, \mathsf{f}) := \mathtt{F}$ otherwise
  - $\mathsf{q}^{\alpha} := (0, \mathsf{q}^{\alpha}) \in \mathcal{D}_{\mathsf{o}\alpha\alpha}$ where $\mathsf{q}^{\alpha}(\mathsf{a}) := (0, \mathsf{s}^{\mathsf{a}})$ and $\mathsf{s}^{\mathsf{a}}(\mathsf{b}) := \mathtt{T}$ if $\mathsf{a} = \mathsf{b}$ and $\mathsf{s}^{\mathsf{a}}(\mathsf{b}) := \mathtt{F}$ otherwise
  - $\mathcal{E}(\mathsf{w}) \in \mathcal{D}_{\alpha}$ arbitrary for parameters $\mathsf{w} \in \Sigma_{\alpha}$.

Ex.: Models without η and $\mathfrak{f}$

Ex.: Models without η and $\mathfrak{f}$

- For variables, we define $\mathcal{E}_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$

Ex.: Models without η and $\mathfrak{f}$

- For variables, we define $\mathcal{E}_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$
- For application, we define $\mathcal{E}_\varphi(\mathbf{FA}) := \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$

Ex.: Models without η and $\mathfrak{f}$

- For variables, we define $\mathcal{E}_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$
- For application, we define $\mathcal{E}_\varphi(\mathbf{FA}) := \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$
- For λ-abstractions, we define $\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha.\mathbf{B}_\beta) := (0, \mathsf{f})$ where $\mathsf{f}\colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta$ is the function such that $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_\alpha$

Ex.: Models without η and $\mathfrak{f}$

- For variables, we define $\mathcal{E}_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$
- For application, we define $\mathcal{E}_\varphi(\mathbf{FA}) := \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$
- For λ-abstractions, we define $\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha\mathbf{.}\mathbf{B}_\beta) := (0, \mathsf{f})$ where $\mathsf{f}\colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta$ is the function such that $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_\alpha$

- With some work (which we omit), one can show that this $\mathcal{E}$ is an evaluation function

Ex.: Models without η and $\mathfrak{f}$

- For variables, we define $\mathcal{E}_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$
- For application, we define $\mathcal{E}_\varphi(\mathbf{FA}) := \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$
- For λ-abstractions, we define $\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha.\mathbf{B}_\beta) := (0, \mathsf{f})$ where $\mathsf{f}\colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta$ is the function such that $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_\alpha$

- With some work (which we omit), one can show that this $\mathcal{E}$ is an evaluation function
- Taking υ to be the function such that $\upsilon(\mathsf{b}) := \mathtt{T}$ for every $\mathsf{b} \in \mathcal{B}$ and $\upsilon(\mathtt{F}) := \mathtt{F}$, one can easily show that this is a valuation

Ex.: Models without η and $\mathfrak{f}$

- For variables, we define $\mathcal{E}_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$
- For application, we define $\mathcal{E}_\varphi(\mathbf{FA}) := \mathcal{E}_\varphi(\mathbf{F})@\mathcal{E}_\varphi(\mathbf{A})$
- For λ-abstractions, we define $\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha . \mathbf{B}_\beta) := (0, \mathsf{f})$ where $\mathsf{f}\colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta$ is the function such that $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_\alpha$

- With some work (which we omit), one can show that this $\mathcal{E}$ is an evaluation function
- Taking υ to be the function such that $\upsilon(\mathsf{b}) := \mathtt{T}$ for every $\mathsf{b} \in \mathcal{B}$ and $\upsilon(\mathtt{F}) := \mathtt{F}$, one can easily show that this is a valuation
- Hence, $\mathcal{M}^{\mathcal{B}} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is a Σ-model

Ex.: Models without η and $\mathfrak{f}$

- The objects $\mathsf{q}^{\alpha} := (0, \mathsf{q}^{\alpha})$ witness property $\mathfrak{q}$ for $\mathcal{M}^{\mathcal{B}}$

Ex.: Models without η and $\mathfrak{f}$

- The objects $\mathsf{q}^\alpha := (0, \mathsf{q}^\alpha)$ witness property $\mathfrak{q}$ for $\mathcal{M}^\mathcal{B}$
- The objects $(1, \mathsf{q}^\alpha)$ also witness property $\mathfrak{q}$ (so, in the non-functional case such witnesses are not unique)

Ex.: Models without η and $\mathfrak{f}$

- The objects $\mathsf{q}^\alpha := (0, \mathsf{q}^\alpha)$ witness property $\mathfrak{q}$ for $\mathcal{M}^\mathcal{B}$
- The objects $(1, \mathsf{q}^\alpha)$ also witness property $\mathfrak{q}$ (so, in the non-functional case such witnesses are not unique)
- Hence, $\mathcal{M}^\mathcal{B} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is a Σ-model with property $\mathfrak{q}$

Ex.: Models without η and $\mathfrak{f}$

- Property $\mathfrak{f}$ fails for $\mathcal{M}^{\mathcal{B}}$, since the applicative structure $(\mathcal{D}, @)$ is not functional:

Ex.: Models without η and $\mathfrak{f}$

- Property $\mathfrak{f}$ fails for $\mathcal{M}^{\mathcal{B}}$, since the applicative structure $(\mathcal{D}, @)$ is not functional:
  - Consider $\mathrm{u}\colon \mathcal{D}_\iota \longrightarrow \mathcal{D}_\iota$.

Ex.: Models without η and $\mathfrak{f}$

- Property $\mathfrak{f}$ fails for $\mathcal{M}^{\mathcal{B}}$, since the applicative structure $(\mathcal{D}, @)$ is not functional:
  - Consider $\mathsf{u}\colon \mathcal{D}_\iota \longrightarrow \mathcal{D}_\iota$.
  - For both $(0, \mathsf{u}), (1, \mathsf{u}) \in \mathcal{D}_{\iota\iota}$ we have

$$(\mathsf{i}, \mathsf{u})@* = *$$

although $(0, \mathsf{u}) \neq (1, \mathsf{u})$

Ex.: Models without η and $\mathfrak{f}$

- Does η hold?

Ex.: Models without η and $\mathfrak{f}$

- Does η hold?
- No!

Ex.: Models without η and $\mathfrak{f}$

- Does η hold?
- No!
- Compute, for example, $\mathcal{E}(\lambda F_{\beta\alpha}. F)$ and $\mathcal{E}(\lambda F_{\beta\alpha}. \lambda X_{\alpha}. FX)$

Ex.: Models without η and $\mathfrak{f}$

- Does η hold?
- No!
- Compute, for example, $\mathcal{E}(\lambda F_{\beta\alpha}\text{.}F)$ and $\mathcal{E}(\lambda F_{\beta\alpha}\text{.}\lambda X_{\alpha}\text{.}FX)$
  - $\mathcal{E}(\lambda F_{\beta\alpha}\text{.}F) = (0, \mathsf{id})$ where id is the identity function from $\mathcal{D}_{\beta\alpha}$ to $\mathcal{D}_{\beta\alpha}$

Ex.: Models without η and $\mathfrak{f}$

- Does η hold?
- No!
- Compute, for example, $\mathcal{E}(\lambda F_{\beta\alpha}.F)$ and $\mathcal{E}(\lambda F_{\beta\alpha}.\lambda X_{\alpha}.FX)$
  - $\mathcal{E}(\lambda F_{\beta\alpha}.F) = (0, \mathsf{id})$ where id is the identity function from $\mathcal{D}_{\beta\alpha}$ to $\mathcal{D}_{\beta\alpha}$
  - $\mathcal{E}(\lambda F_{\beta\alpha}.\lambda X_{\alpha}.FX) = (0, \mathsf{p})$ where p is the function from $\mathcal{D}_{\beta\alpha}$ to $\mathcal{D}_{\beta\alpha}$ such that $\mathsf{p}((\mathsf{i}, \mathsf{f})) = (0, \mathsf{f})$ for each $\mathsf{f}\colon \mathcal{D}_{\alpha} \longrightarrow \mathcal{D}_{\beta}$

Ex.: Models without η and $\mathfrak{f}$

- Does η hold?
- No!
- Compute, for example, $\mathcal{E}(\lambda F_{\beta\alpha} . F)$ and $\mathcal{E}(\lambda F_{\beta\alpha} . \lambda X_\alpha . FX)$
  - $\mathcal{E}(\lambda F_{\beta\alpha} . F) = (0, \mathsf{id})$ where id is the identity function from $\mathcal{D}_{\beta\alpha}$ to $\mathcal{D}_{\beta\alpha}$
  - $\mathcal{E}(\lambda F_{\beta\alpha} . \lambda X_\alpha . FX) = (0, \mathsf{p})$ where p is the function from $\mathcal{D}_{\beta\alpha}$ to $\mathcal{D}_{\beta\alpha}$ such that $\mathsf{p}((\mathsf{i}, \mathsf{f})) = (0, \mathsf{f})$ for each $\mathsf{f}\colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta$
- Hence $\mathcal{E}(\lambda F_{\beta\alpha} . F) \neq \mathcal{E}(\lambda F_{\beta\alpha} . \lambda X_\alpha . FX)$

Ex.: Models without η and $\mathfrak{f}$

- Does ξ hold?

Ex.: Models without η and $\mathfrak{f}$

- Does ξ hold?
- Yes!

Ex.: Models without η and $\mathfrak{f}$

- Does ξ hold?
- Yes!
- If

$$\mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{M}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{N})$$

  for every $\mathsf{a} \in \mathcal{D}_\alpha$, then

$$\mathcal{E}_\varphi(\lambda \mathsf{X}_\alpha \mathbf{.} \mathbf{M}) = (0, \mathsf{f}) = \mathcal{E}_\varphi(\lambda \mathsf{X} \mathbf{.} \mathbf{N})$$

  where $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{M}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{N})$ for every $\mathsf{a} \in \mathcal{D}_\alpha$.

Ex.: Models without η and $\mathfrak{f}$

- If $\mathcal{B} = \{\mathtt{T}\}$, then the model $\mathcal{M}^{\beta\xi\mathfrak{b}} := \mathcal{M}^{\{\mathtt{T}\}}$ satisfies property $\mathfrak{b}$.

Ex.: Models without η and $\mathfrak{f}$

- If $\mathcal{B} = \{\mathtt{T}\}$, then the model $\mathcal{M}^{\beta\xi\mathfrak{b}} := \mathcal{M}^{\{\mathtt{T}\}}$ satisfies property $\mathfrak{b}$.
- So, we know $\mathcal{M}^{\beta\xi\mathfrak{b}} \in \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma) \setminus \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma)$.

Ex.: Models without η and $\mathfrak{f}$

- If $\mathcal{B} = \{\mathtt{T}\}$, then the model $\mathcal{M}^{\beta\xi\mathfrak{b}} := \mathcal{M}^{\{\mathtt{T}\}}$ satisfies property $\mathfrak{b}$.
- So, we know $\mathcal{M}^{\beta\xi\mathfrak{b}} \in \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma) \setminus \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma)$.
- On the other hand, if $\mathtt{b}$ is any value with $\mathtt{b} \notin \{\mathtt{T}, \mathtt{F}\}$, and $\mathcal{B} = \{\mathtt{T}, \mathtt{b}\}$, then the model $\mathcal{M}^{\beta\xi} := \mathcal{M}^{\{\mathtt{T},\mathtt{b}\}}$ does not satisfy property $\mathfrak{b}$.

Ex.: Models without η and $\mathfrak{f}$

- If $\mathcal{B} = \{\mathtt{T}\}$, then the model $\mathcal{M}^{\beta\xi\mathfrak{b}} := \mathcal{M}^{\{\mathtt{T}\}}$ satisfies property $\mathfrak{b}$.
- So, we know $\mathcal{M}^{\beta\xi\mathfrak{b}} \in \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma) \setminus \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma)$.
- On the other hand, if b is any value with $\mathsf{b} \notin \{\mathtt{T}, \mathtt{F}\}$, and $\mathcal{B} = \{\mathtt{T}, \mathtt{b}\}$, then the model $\mathcal{M}^{\beta\xi} := \mathcal{M}^{\{\mathtt{T},\mathtt{b}\}}$ does not satisfy property $\mathfrak{b}$.
- In this case, we know $\mathcal{M}^{\beta\xi} \in \mathfrak{M}_{\beta\xi}(\Sigma) \setminus (\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma) \cup \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma))$.

Ex.: Models without η and $\mathfrak{f}$

$$
\begin{tikzcd}
 & & \mathfrak{M}_{\beta}(\Sigma) \arrow[dll, "\xi"'] \arrow[d, "\eta"] \arrow[drr, "\mathfrak{b}"] \arrow[ddll, "\mathfrak{f}"] & & \\
\mathfrak{M}_{\beta\xi}(\Sigma) \arrow[d, "\eta"] \arrow[drr, "\mathfrak{b}"] & & \mathfrak{M}_{\beta\eta}(\Sigma) \arrow[dll, "\xi"'] \arrow[drr, "\mathfrak{b}"] & & \mathfrak{M}_{\beta\mathfrak{b}}(\Sigma) \arrow[dll, "\xi"'] \arrow[d, "\eta"] \arrow[ddll, "\mathfrak{f}"] \\
\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma) \arrow[drr, "\mathfrak{b}"] & & \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma) \arrow[d, "\eta"] & & \mathfrak{M}_{\beta\eta\mathfrak{b}}(\Sigma) \arrow[dll, "\xi"'] \\
 & & \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma) \simeq \mathfrak{H}(\Sigma) \arrow[d, "\text{full}"] & & \\
 & & \mathfrak{S}\mathfrak{T}(\Sigma) & &
\end{tikzcd}
$$

Ex.: Models without η and $\mathfrak{f}$

$$
\begin{tikzcd}[column sep=small]
 & & \mathfrak{M}_{\beta}(\Sigma) \arrow[dll, "\xi"'] \arrow[ddll, "\mathfrak{f}"] \arrow[d, "\eta"] \arrow[drr, "\mathfrak{b}"] & & \\
\mathfrak{M}_{\beta\xi}(\Sigma) \arrow[d, "\eta"] \arrow[drr, "\mathfrak{b}"] & & \mathfrak{M}_{\beta\eta}(\Sigma) \arrow[dll, "\xi"'] \arrow[drr, "\mathfrak{b}"] & & \mathfrak{M}_{\beta\mathfrak{b}}(\Sigma) \arrow[dll, "\xi"'] \arrow[ddll, "\mathfrak{f}"] \arrow[d, "\eta"] \\
\mathfrak{M}_{\beta\mathfrak{f}}(\Sigma) \arrow[drr, "\mathfrak{b}"] & & \mathfrak{M}_{\beta\xi\mathfrak{b}}(\Sigma) \arrow[d, "\eta"] & & \mathfrak{M}_{\beta\eta\mathfrak{b}}(\Sigma) \arrow[dll, "\xi"'] \\
 & & \mathfrak{M}_{\beta\mathfrak{f}\mathfrak{b}}(\Sigma) \simeq \mathfrak{H}(\Sigma) \arrow[d, "\text{full}"] & & \\
 & & \mathfrak{S}\mathfrak{T}(\Sigma) & &
\end{tikzcd}
$$

Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before

Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before
- Define an alternative evaluation function $\mathcal{E}'$ by induction:

Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before
- Define an alternative evaluation function $\mathcal{E}'$ by induction:
  - For all $\mathrm{w} \in \Sigma$, let $\mathcal{E}'(\mathrm{w}) := \mathcal{E}(\mathrm{w})$

Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before
- Define an alternative evaluation function $\mathcal{E}'$ by induction:
  - For all $\mathsf{w} \in \Sigma$, let $\mathcal{E}'(\mathsf{w}) := \mathcal{E}(\mathsf{w})$
  - For variables we define $\mathcal{E}'_{\varphi}(\mathsf{X}) := \varphi(\mathsf{X})$

Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before
- Define an alternative evaluation function $\mathcal{E}'$ by induction:
  - For all $\mathrm{w} \in \Sigma$, let $\mathcal{E}'(\mathrm{w}) := \mathcal{E}(\mathrm{w})$
  - For variables we define $\mathcal{E}'_\varphi(\mathrm{X}) := \varphi(\mathrm{X})$
  - We must define $\mathcal{E}'_\varphi(\mathbf{FA}) := \mathcal{E}'_\varphi(\mathbf{F})@\mathcal{E}'_\varphi(\mathbf{A})$

Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before
- Define an alternative evaluation function $\mathcal{E}'$ by induction:
  - For all $\mathsf{w} \in \Sigma$, let $\mathcal{E}'(\mathsf{w}) := \mathcal{E}(\mathsf{w})$
  - For variables we define $\mathcal{E}'_{\varphi}(\mathsf{X}) := \varphi(\mathsf{X})$
  - We must define $\mathcal{E}'_{\varphi}(\mathbf{FA}) := \mathcal{E}'_{\varphi}(\mathbf{F})@\mathcal{E}'_{\varphi}(\mathbf{A})$
  - We choose $\mathcal{E}'_{\varphi}(\lambda \mathsf{X}_{\alpha}\text{.}\mathbf{B}_{\beta}) := (1, \mathsf{f})$ where $\mathsf{f}\colon \mathcal{D}_{\alpha} \longrightarrow \mathcal{D}_{\beta}$ is the function such that $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_{\alpha}$

Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before
- Define an alternative evaluation function $\mathcal{E}'$ by induction:
  - For all $\mathsf{w} \in \Sigma$, let $\mathcal{E}'(\mathsf{w}) := \mathcal{E}(\mathsf{w})$
  - For variables we define $\mathcal{E}'_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$
  - We must define $\mathcal{E}'_\varphi(\mathbf{FA}) := \mathcal{E}'_\varphi(\mathbf{F})@\mathcal{E}'_\varphi(\mathbf{A})$
  - We choose $\mathcal{E}'_\varphi(\lambda \mathsf{X}_\alpha . \mathbf{B}_\beta) := (1, \mathsf{f})$ where $\mathsf{f} : \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta$ is the function such that $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_\alpha$
- $\mathcal{E}$ and $\mathcal{E}'$ agree on all constants, they are different though:

$$\mathcal{E}(\lambda \mathsf{X}_\iota . \mathsf{X}) = (0, \mathsf{id}) \neq (1, \mathsf{id}) = \mathcal{E}'(\lambda \mathsf{X}_\iota . \mathsf{X})$$

where $\mathsf{id} : \mathcal{D}_\iota \longrightarrow \mathcal{D}_\iota$ is the identity function

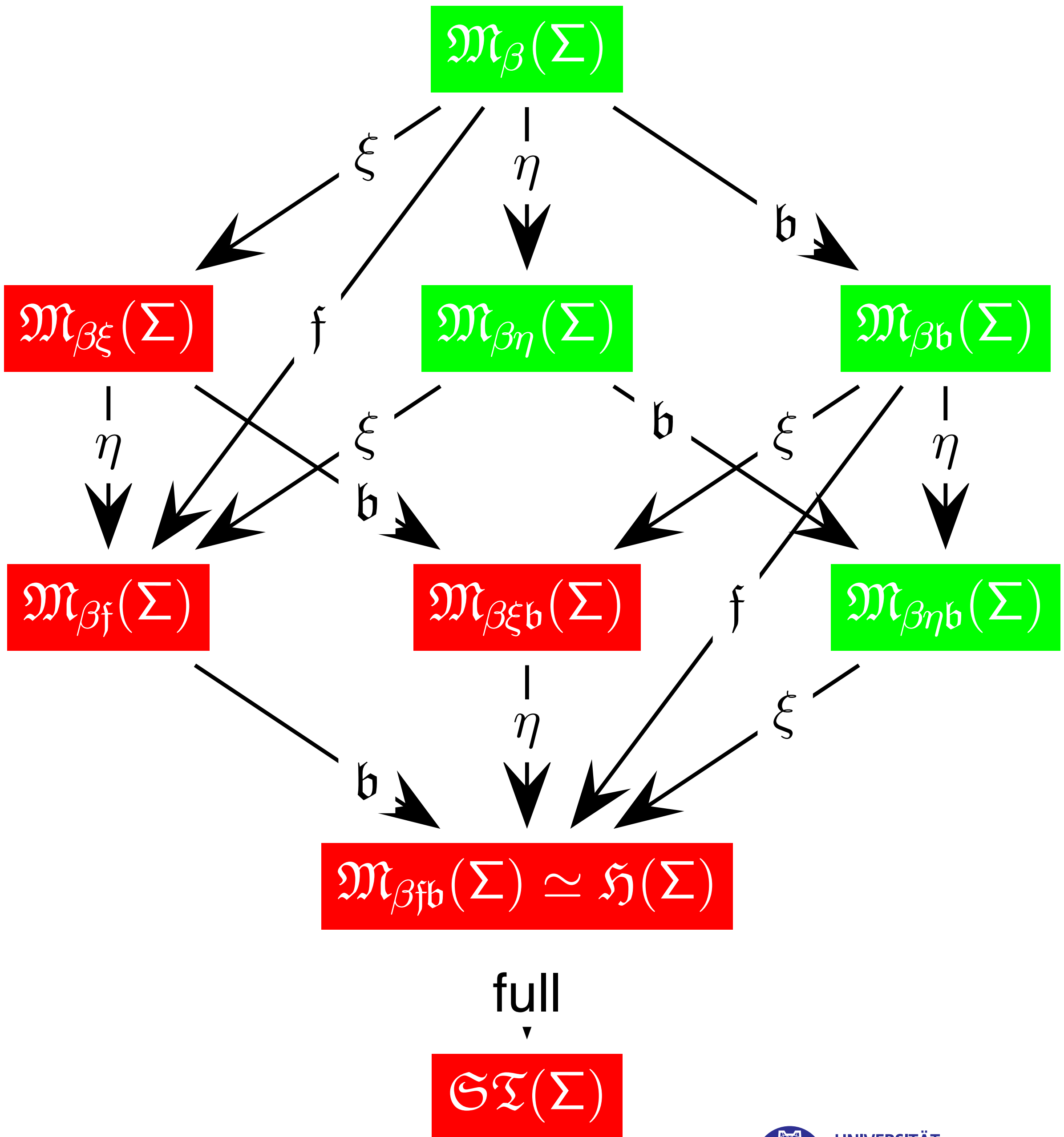
Ex.: Models without η and $\mathfrak{f}$

- Let $\mathcal{M}^{\mathcal{B}}$ be the Σ-model $(\mathcal{D}, @, \mathcal{E}, \upsilon)$ as constructed before
- Define an alternative evaluation function $\mathcal{E}'$ by induction:
  - For all $\mathsf{w} \in \Sigma$, let $\mathcal{E}'(\mathsf{w}) := \mathcal{E}(\mathsf{w})$
  - For variables we define $\mathcal{E}'_\varphi(\mathsf{X}) := \varphi(\mathsf{X})$
  - We must define $\mathcal{E}'_\varphi(\mathbf{FA}) := \mathcal{E}'_\varphi(\mathbf{F})@\mathcal{E}'_\varphi(\mathbf{A})$
  - We choose $\mathcal{E}'_\varphi(\lambda \mathsf{X}_\alpha.\mathbf{B}_\beta) := (1, \mathsf{f})$ where $\mathsf{f}\colon \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta$ is the function such that $\mathsf{f}(\mathsf{a}) = \mathcal{E}_{\varphi,[\mathsf{a}/\mathsf{X}]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_\alpha$
- $\mathcal{E}$ and $\mathcal{E}'$ agree on all constants, they are different though:

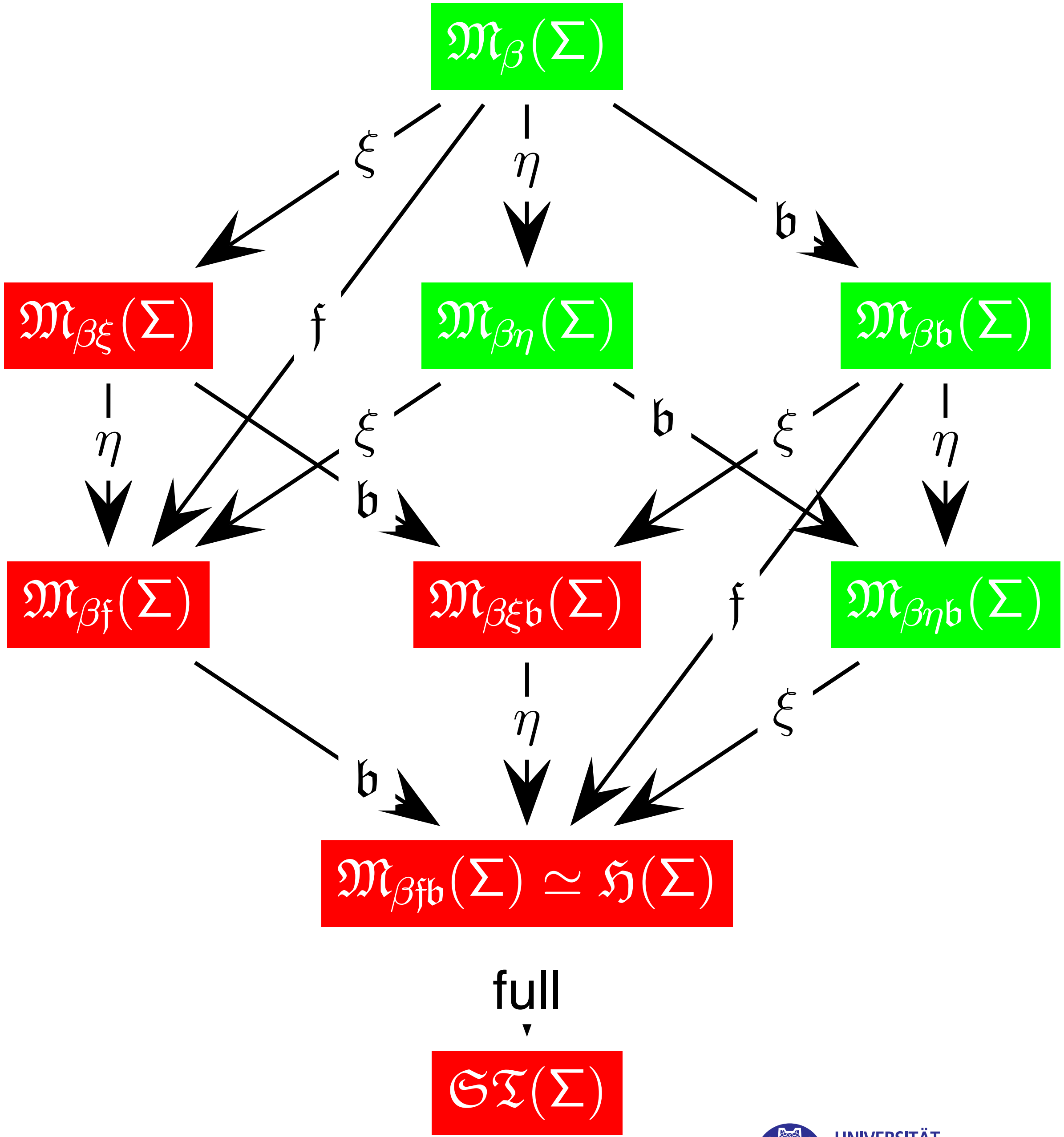
$$\mathcal{E}(\lambda \mathsf{X}_\iota.\mathsf{X}) = (0, \mathsf{id}) \neq (1, \mathsf{id}) = \mathcal{E}'(\lambda \mathsf{X}_\iota.\mathsf{X})$$

  where $\mathsf{id} : \mathcal{D}_\iota \longrightarrow \mathcal{D}_\iota$ is the identity function
- Thus, in non-functional models evaluation functions are not uniquely determined by their values on constants

Ex.: Models without ξ



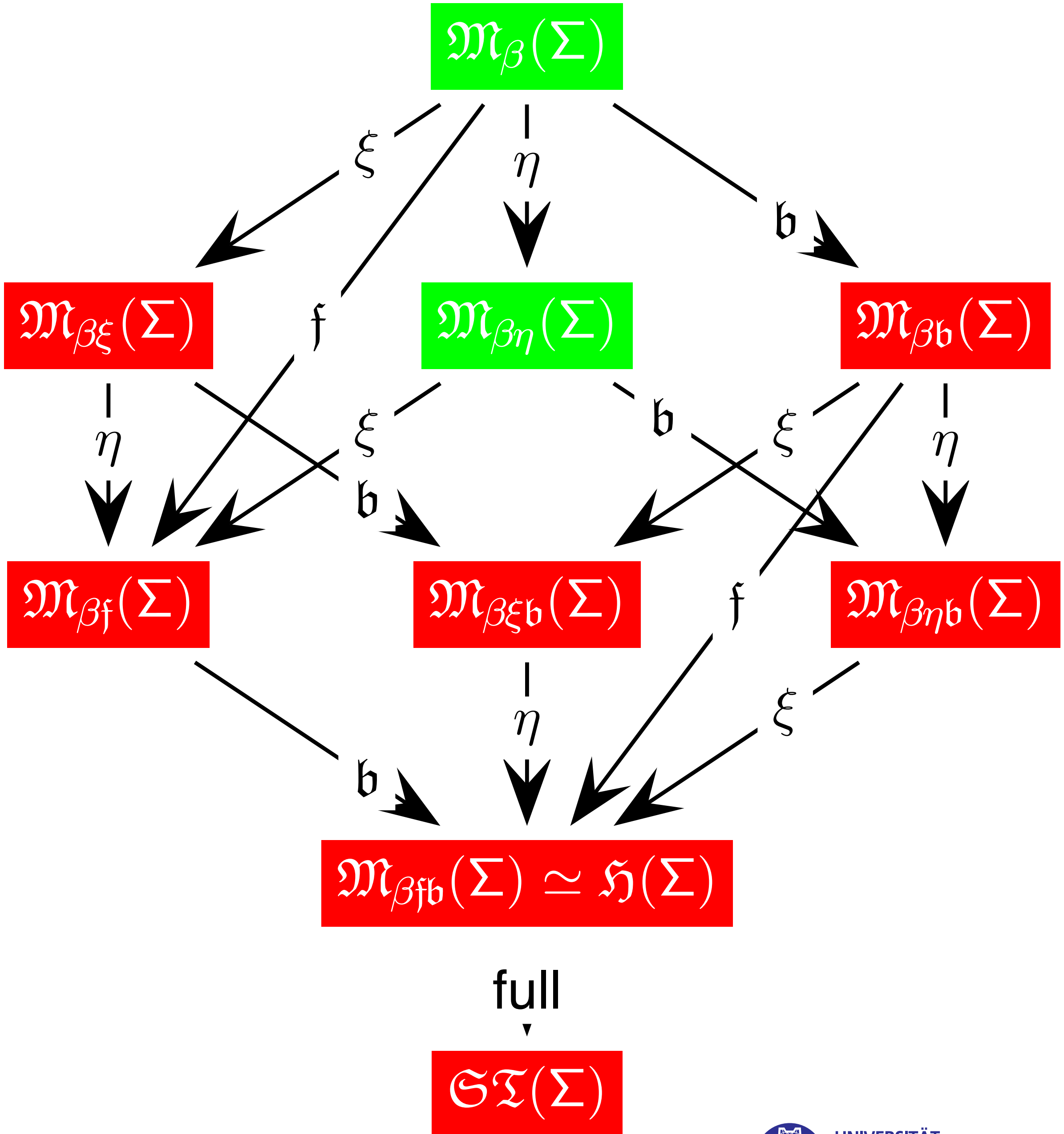
Ex.: Models without ξ



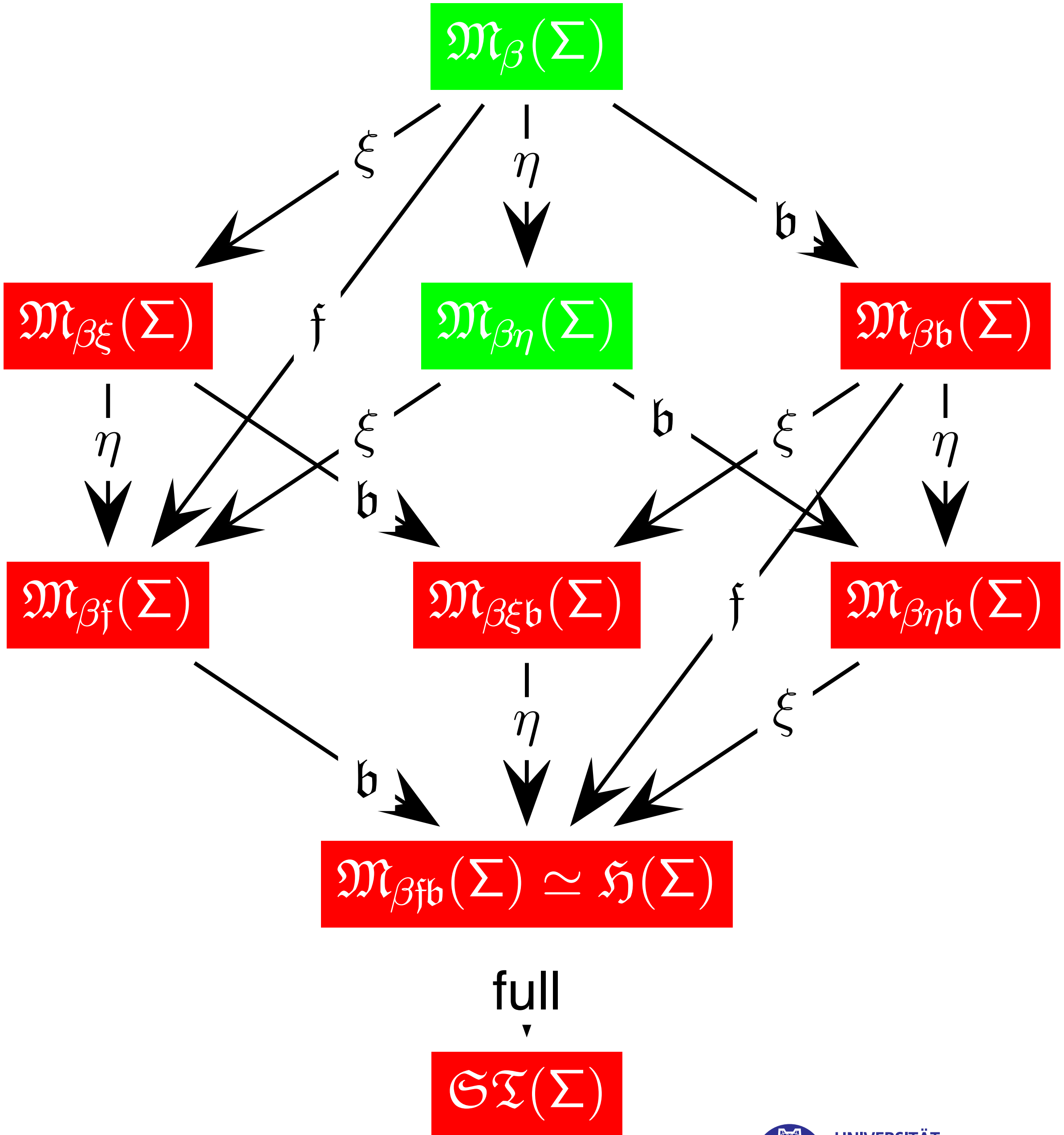
Not here!

See [JSL-04]

Ex.: Models without ξ



Ex.: Models without ξ



Not here!

See [JSL-04]

Calculi: First-Order Natural Deduction and Sequent Calculus

From Natural Deduction

to Sequent Calculus

and Back

From Natural Deduction
to Sequent Calculus
and Back

Remark: We first illustrate the correspondence between natural deduction and sequent calculus in first-order logic. Later we will present natural deduction calculi for HOL. More precisely we will present one sound and complete calculus for each class in our landscape of semantics as presented before.

Reading

- F. Pfenning: Automated Theorem Proving, Course at Carnegie Mellon University. Draft. 1999.
- A.S. Troelstra and H. Schwichtenberg: Basic Proof Theory. Cambridge. 2nd Edition 2000.
- John Byrnes: Proof Search and Normal Forms in Natural Deduction. PhD Thesis. Carnegie Mellon University. 1999.
- . . . many more books on Proof Theory . . .

Natural Deduction: Motivation

- Frege, Russel, Hilbert: Predicate calculus and type theory as formal basis for mathematics

Natural Deduction: Motivation

- Frege, Russel, Hilbert: Predicate calculus and type theory as formal basis for mathematics

- Gentzen: Natural deduction (ND) as intuitive formulation of predicate calculus; introduction and elimination rules for each logical connective

Natural Deduction: Motivation

- Frege, Russel, Hilbert: Predicate calculus and type theory as formal basis for mathematics

- Gentzen: Natural deduction (ND) as intuitive formulation of predicate calculus; introduction and elimination rules for each logical connective

The formalization of logical deduction, especially as it has been developed by Frege, Russel, and Hilbert, is rather far removed from the forms of deduction used in practice in mathematical proofs. . . . In contrast I intended first to set up a formal system which comes as close as possible to actual reasoning. The result was a calculus of natural deduction (NJ for intuitionist, NK for classical predicate logic).
[Gentzen: Investigations into logical deduction]

Sequent Calculus: Motivation

- Gentzen had a pure technical motivation for sequent calculus:

Sequent Calculus: Motivation

- Gentzen had a pure technical motivation for sequent calculus:
  - same theorems as natural deduction

Sequent Calculus: Motivation

- Gentzen had a pure technical motivation for sequent calculus:
  - same theorems as natural deduction
  - prove of the Hauptsatz (all sequent proofs can be found with a simple strategy)

Sequent Calculus: Motivation

- Gentzen had a pure technical motivation for sequent calculus:
  - same theorems as natural deduction
  - prove of the Hauptsatz (all sequent proofs can be found with a simple strategy)
  - corollary: consistency of formal system(s)

Sequent Calculus: Motivation

- Gentzen had a pure technical motivation for sequent calculus:
  - same theorems as natural deduction
  - prove of the Hauptsatz (all sequent proofs can be found with a simple strategy)
  - corollary: consistency of formal system(s)

The Hauptsatz says that every purely logical proof can be reduced to a definite, though not unique, normal form. Perhaps we may express the essential properties of such a normal proof by saying: it is not round-about. . . .
In order to be able to prove the Hauptsatz in a convenient form, I had to provide a logical calculus especially for the purpose. For this the natural calculus proved unsuitable.
[Gentzen: Investigations into logical deduction]

Sequent Calculus: Introduction

- Sequent calculus exposes many details of fine structure of proofs in a very clear manner. Therefore it is well suited to serve as a basic representation formalism for many automation oriented search procedures

Sequent Calculus: Introduction

- Sequent calculus exposes many details of fine structure of proofs in a very clear manner. Therefore it is well suited to serve as a basic representation formalism for many automation oriented search procedures
  - Backward: tableaux, connection methods, matrix methods, some forms of resolution

Sequent Calculus: Introduction

- Sequent calculus exposes many details of fine structure of proofs in a very clear manner. Therefore it is well suited to serve as a basic representation formalism for many automation oriented search procedures
  - Backward: tableaux, connection methods, matrix methods, some forms of resolution
  - Forward: classical resolution, inverse method

Sequent Calculus: Introduction

- Sequent calculus exposes many details of fine structure of proofs in a very clear manner. Therefore it is well suited to serve as a basic representation formalism for many automation oriented search procedures
  - Backward: tableaux, connection methods, matrix methods, some forms of resolution
  - Forward: classical resolution, inverse method
- Don't be afraid of the many variants of sequent calculi.

Sequent Calculus: Introduction

- Sequent calculus exposes many details of fine structure of proofs in a very clear manner. Therefore it is well suited to serve as a basic representation formalism for many automation oriented search procedures
  - Backward: tableaux, connection methods, matrix methods, some forms of resolution
  - Forward: classical resolution, inverse method
- Don't be afraid of the many variants of sequent calculi.
- Choose the one that is most suited for you.

Natural Deduction

Natural deduction rules operate on proof trees.

Example:

Natural Deduction

Natural deduction rules operate on proof trees.

Example:

- Conjunction: $\dfrac{\begin{matrix}\mathsf{D}_1 & & \mathsf{D}_2\\ \mathbf{A} & & \mathbf{B}\end{matrix}}{\mathbf{A} \wedge \mathbf{B}} \wedge\mathsf{I} \qquad \dfrac{\begin{matrix}\mathsf{D}_1\\ \mathbf{A} \wedge \mathbf{B}\end{matrix}}{\mathbf{A}} \wedge\mathsf{E}_l \qquad \dfrac{\begin{matrix}\mathsf{D}_1\\ \mathbf{A} \wedge \mathbf{B}\end{matrix}}{\mathbf{B}} \wedge\mathsf{E}_r$

Natural Deduction

Natural deduction rules operate on proof trees.
Example:

- Conjunction: $\dfrac{\begin{array}{cc}\mathsf{D}_1 & \mathsf{D}_2\\ \mathbf{A} & \mathbf{B}\end{array}}{\mathbf{A} \wedge \mathbf{B}} \wedge\mathsf{I}$ $\quad \dfrac{\begin{array}{c}\mathsf{D}_1\\ \mathbf{A} \wedge \mathbf{B}\end{array}}{\mathbf{A}} \wedge\mathsf{E}_l$ $\quad \dfrac{\begin{array}{c}\mathsf{D}_1\\ \mathbf{A} \wedge \mathbf{B}\end{array}}{\mathbf{B}} \wedge\mathsf{E}_r$

The presentation on the next slides treats the proof tree aspects implicit.
Example:

Natural Deduction

Natural deduction rules operate on proof trees.
Example:

- Conjunction: $\dfrac{\begin{matrix}\mathcal{D}_1 & \mathcal{D}_2\\ \mathbf{A} & \mathbf{B}\end{matrix}}{\mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \dfrac{\begin{matrix}\mathcal{D}_1\\ \mathbf{A} \wedge \mathbf{B}\end{matrix}}{\mathbf{A}} \wedge E_l \qquad \dfrac{\begin{matrix}\mathcal{D}_1\\ \mathbf{A} \wedge \mathbf{B}\end{matrix}}{\mathbf{B}} \wedge E_r$

The presentation on the next slides treats the proof tree aspects implicit.
Example:

- Conjunction: $\dfrac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}} \wedge E_l \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}} \wedge E_r$

Natural Deduction Rules Ia

- Conjunction: $\dfrac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}}\ \wedge\mathsf{I} \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}}\ \wedge\mathsf{E}_l \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}}\ \wedge\mathsf{E}_r$

Natural Deduction Rules Ia

- Conjunction: $\dfrac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}} \wedge\mathsf{I} \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}} \wedge\mathsf{E}_l \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}} \wedge\mathsf{E}_r$
- Disjunction: $\dfrac{\mathbf{A}}{\mathbf{A} \vee \mathbf{B}} \vee\mathsf{I}_l \qquad \dfrac{\mathbf{B}}{\mathbf{A} \vee \mathbf{B}} \vee\mathsf{I}_r \qquad \dfrac{\mathbf{A} \vee \mathbf{B} \quad \begin{matrix}[\mathbf{A}]_1 \\ \vdots \\ \mathbf{C}\end{matrix} \quad \begin{matrix}[\mathbf{B}]_2 \\ \vdots \\ \mathbf{C}\end{matrix}}{\mathbf{C}} \vee\mathsf{E}_r^{1,2}$

Natural Deduction Rules Ia

- Conjunction: $\dfrac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}} \wedge \mathsf{I}$ $\quad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}} \wedge \mathsf{E}_l$ $\quad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}} \wedge \mathsf{E}_r$

- Disjunction: $\dfrac{\mathbf{A}}{\mathbf{A} \vee \mathbf{B}} \vee \mathsf{I}_l$ $\quad \dfrac{\mathbf{B}}{\mathbf{A} \vee \mathbf{B}} \vee \mathsf{I}_r$ $\quad \dfrac{\mathbf{A} \vee \mathbf{B} \quad \begin{array}{c}[\mathbf{A}]_1 \\ \vdots \\ \mathbf{C}\end{array} \quad \begin{array}{c}[\mathbf{B}]_2 \\ \vdots \\ \mathbf{C}\end{array}}{\mathbf{C}} \vee \mathsf{E}_r^{1,2}$

- Implication: $\dfrac{\begin{array}{c}[\mathbf{A}]_1 \\ \vdots \\ \mathbf{B}\end{array}}{\mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow \mathsf{I}^1$ $\quad \dfrac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}} \Rightarrow \mathsf{E}$

Natural Deduction Rules Ia

- Conjunction: $\dfrac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}}\ \wedge\mathsf{I} \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}}\ \wedge\mathsf{E}_{\mathsf{l}} \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}}\ \wedge\mathsf{E}_{\mathsf{r}}$

- Disjunction: $\dfrac{\mathbf{A}}{\mathbf{A} \vee \mathbf{B}}\ \vee\mathsf{I}_{\mathsf{l}} \qquad \dfrac{\mathbf{B}}{\mathbf{A} \vee \mathbf{B}}\ \vee\mathsf{I}_{\mathsf{r}} \qquad \dfrac{\mathbf{A} \vee \mathbf{B} \quad \begin{matrix}[\mathbf{A}]_1 \\ \vdots \\ \mathbf{C}\end{matrix} \quad \begin{matrix}[\mathbf{B}]_2 \\ \vdots \\ \mathbf{C}\end{matrix}}{\mathbf{C}}\ \vee\mathsf{E}_{\mathsf{r}}^{1,2}$

- Implication: $\dfrac{\begin{matrix}[\mathbf{A}]_1 \\ \vdots \\ \mathbf{B}\end{matrix}}{\mathbf{A} \Rightarrow \mathbf{B}}\ \Rightarrow\mathsf{I}^1 \qquad \dfrac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}\ \Rightarrow\mathsf{E}$

- Truth and Falsehood: $\dfrac{}{\top}\ \top\mathsf{I} \qquad \dfrac{\bot}{\mathbf{C}}\ \bot\mathsf{E}$

Natural Deduction Rules IIa

- Negation:

$$\frac{\begin{array}{c}[\mathbf{A}]_1 \\ \vdots \\ \bot\end{array}}{\neg\mathbf{A}}\ \neg\mathsf{I}^1 \qquad \frac{\neg\mathbf{A} \quad \mathbf{A}}{\bot}\ \neg\mathsf{E}$$

Natural Deduction Rules IIa

- Negation:

$$\dfrac{\begin{array}{c}[\mathbf{A}]_1 \\ \vdots \\ \bot\end{array}}{\neg\mathbf{A}}\ \neg\mathrm{I}^1 \qquad \dfrac{\neg\mathbf{A} \quad \mathbf{A}}{\bot}\ \neg\mathrm{E}$$

- Universal Quantif.:

$$\dfrac{\mathbf{A}[\mathsf{x}/\mathsf{P}^*]}{\forall \mathsf{x}.\mathbf{A}}\ \forall\mathrm{I} \qquad \dfrac{\forall \mathsf{x}.\mathbf{A}}{\mathbf{A}[\mathsf{x}/\mathbf{T}]}\ \forall\mathrm{E}$$

(*: parameter P must be new in context)

Natural Deduction Rules IIa

- Negation:

$$\dfrac{\begin{array}{c}[\mathbf{A}]_1\\ \vdots\\ \bot\end{array}}{\neg\mathbf{A}}\ \neg\mathrm{I}^1 \qquad \dfrac{\neg\mathbf{A}\quad \mathbf{A}}{\bot}\ \neg\mathrm{E}$$

- Universal Quantif.:

$$\dfrac{\mathbf{A}[\mathsf{x}/\mathsf{P}^*]}{\forall \mathsf{x}.\mathbf{A}}\ \forall\mathrm{I} \qquad \dfrac{\forall \mathsf{x}.\mathbf{A}}{\mathbf{A}[\mathsf{x}/\mathbf{T}]}\ \forall\mathrm{E}$$

(*: parameter P must be new in context)

- Existential Quantif.:

$$\dfrac{\mathbf{A}[\mathsf{x}/\mathbf{T}]}{\exists \mathsf{x}.\mathbf{A}}\ \exists\mathrm{I} \qquad \dfrac{\exists \mathsf{x}.\mathbf{A} \quad \begin{array}{c}[\mathbf{A}[\mathsf{x}/\mathsf{P}^*]]\\ \vdots\\ \mathbf{C}\end{array}}{\mathbf{C}}\ \exists\mathrm{E}$$

(*: parameter P must be new in context)

Natural Deduction Rules IIIa

- For classical logic choose one of the following

Natural Deduction Rules IIIa

- For classical logic choose one of the following
  - Excluded Middle $$\frac{}{\mathbf{A} \lor \neg \mathbf{A}}\ \text{XM}$$

Natural Deduction Rules IIIa

- For classical logic choose one of the following
  - Excluded Middle $$\frac{}{\mathbf{A} \vee \neg\mathbf{A}}\ \mathrm{XM}$$
  - Double Negation $$\frac{\neg\neg\mathbf{A}}{\mathbf{A}}\ \neg\neg\mathrm{C}$$

Natural Deduction Rules IIIa

- For classical logic choose one of the following
  - Excluded Middle $$\frac{}{\mathbf{A} \lor \neg \mathbf{A}}\ \mathrm{XM}$$
  - Double Negation $$\frac{\neg\neg \mathbf{A}}{\mathbf{A}}\ \neg\neg\mathrm{C}$$
  - Proof by Contradiction $$\frac{\begin{array}{c}[\neg \mathbf{A}] \\ \vdots \\ \bot\end{array}}{\mathbf{A}}\ \bot_{\mathrm{c}}$$

Natural Deduction

- Structural properties

Natural Deduction

- Structural properties
  - Exchange — hypotheses order is irrelevant

Natural Deduction

- Structural properties
  - Exchange — hypotheses order is irrelevant
  - Weakening — hypothesis need not be used

Natural Deduction

- Structural properties
  - Exchange — hypotheses order is irrelevant
  - Weakening — hypothesis need not be used
  - Contraction — hypotheses can be used more than once

Natural Deduction Proofs

$$\dfrac{\dfrac{\dfrac{[\mathbf{A}]_1 \quad [\mathbf{A}]_2}{\mathbf{A} \wedge \mathbf{A}} \wedge\mathsf{I}}{\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A})} \Rightarrow\mathsf{I}^2}{\mathbf{A} \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A}))} \Rightarrow\mathsf{I}^1$$

or

$$\dfrac{\dfrac{\dfrac{[\mathbf{A}]_1 \quad [\mathbf{A}]_1}{\mathbf{A} \wedge \mathbf{A}} \wedge\mathsf{I}}{\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A})} \Rightarrow\mathsf{I}^2}{\mathbf{A} \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A}))} \Rightarrow\mathsf{I}^1$$

$$\dfrac{\dfrac{\dfrac{[\mathbf{A} \wedge \mathbf{B}]_1}{\mathbf{B}} \wedge\mathsf{E}_r \quad \dfrac{\dfrac{[\mathbf{A} \wedge \mathbf{B}]_1}{\mathbf{A}} \wedge\mathsf{E}_l}{\mathbf{C} \vee \mathbf{A}} \vee\mathsf{I}_r}{\mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A})} \wedge\mathsf{I}}{(\mathbf{A} \wedge \mathbf{B}) \Rightarrow (\mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A}))} \Rightarrow \mathsf{I}^1$$

Natural Deduction with Contexts

- FO-Soundness of ND: Let F be a first-order formula such that there is a ND proof of F. Then F is valid. $(\vdash F \quad \Rightarrow \models \quad F)$
(Proof: Standard textbooks)

Natural Deduction with Contexts

- FO-Soundness of ND: Let F be a first-order formula such that there is a ND proof of F. Then F is valid. $(\vdash F \;\Rightarrow\; \models F)$
  (Proof: Standard textbooks)
- FO-Completeness of ND: Let F be a valid first-order formula then there is a ND proof of F $(\models F \;\Rightarrow\; \vdash F)$.
  (Proof: Standard textbooks)

Natural Deduction with Contexts

Idea: Localizing hypotheses; explicit representation of the available assumptions for each formula occurrence in a ND proof:

$$\Gamma \vdash \mathbf{A}$$

Γ is a multiset of the (uncanceled) assumptions on which formula $\mathbf{A}$ depends. Γ is called context.

Natural Deduction with Contexts

Idea: Localizing hypotheses; explicit representation of the available assumptions for each formula occurrence in a ND proof:

$$\Gamma \vdash \mathbf{A}$$

Γ is a multiset of the (uncanceled) assumptions on which formula $\mathbf{A}$ depends. Γ is called context.

Example proof in context notation:

$$\dfrac{\dfrac{\dfrac{\overline{\mathbf{A}_1 \vdash \mathbf{A}} \quad \overline{\mathbf{A}_2 \vdash \mathbf{A}}}{\mathbf{A}_1, \mathbf{A}_2 \vdash \mathbf{A} \wedge \mathbf{A}} \wedge I}{\mathbf{A}_1 \vdash \mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A})} \Rightarrow I_2}{\vdash \mathbf{A} \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A}))} \Rightarrow I_1$$

Natural Deduction with Contexts

Another Idea: Consider sets of assumptions instead of multisets.

$$\Gamma \vdash \mathbf{A}$$

Γ is now a set of (uncanceled) assumptions on which formula $\mathbf{A}$ depends.

Natural Deduction with Contexts

Another Idea: Consider sets of assumptions instead of multisets.

$$\Gamma \vdash \mathbf{A}$$

Γ is now a set of (uncanceled) assumptions on which formula $\mathbf{A}$ depends.

Example proof:

$$\cfrac{\cfrac{\cfrac{\overline{\mathbf{A} \vdash \mathbf{A}} \quad \overline{\mathbf{A} \vdash \mathbf{A}}}{\mathbf{A} \vdash \mathbf{A} \land \mathbf{A}}\ \land\mathrm{I}}{\mathbf{A} \vdash \mathbf{A} \Rightarrow (\mathbf{A} \land \mathbf{A})}\ \Rightarrow\mathrm{I}}{\vdash \mathbf{A} \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{A} \land \mathbf{A}))}\ \Rightarrow\mathrm{I}$$

Natural Deduction with Contexts

Structural properties to ensure

Natural Deduction with Contexts

Structural properties to ensure

- Exchange (hypotheses order is irrelevant)

$$\frac{\Gamma, \mathbf{B}, \mathbf{A} \vdash \mathbf{C}}{\Gamma, \mathbf{A}, \mathbf{B} \vdash \mathbf{C}}$$

Natural Deduction with Contexts

Structural properties to ensure

- Exchange (hypotheses order is irrelevant)

$$\frac{\Gamma, \mathbf{B}, \mathbf{A} \vdash \mathbf{C}}{\Gamma, \mathbf{A}, \mathbf{B} \vdash \mathbf{C}}$$

- Weakening (hypothesis need not be used)

$$\frac{\Gamma \vdash \mathbf{C}}{\Gamma, \mathbf{A} \vdash \mathbf{C}}$$

Natural Deduction with Contexts

Structural properties to ensure

- Exchange (hypotheses order is irrelevant)

$$\frac{\Gamma, \mathbf{B}, \mathbf{A} \vdash \mathbf{C}}{\Gamma, \mathbf{A}, \mathbf{B} \vdash \mathbf{C}}$$

- Weakening (hypothesis need not be used)

$$\frac{\Gamma \vdash \mathbf{C}}{\Gamma, \mathbf{A} \vdash \mathbf{C}}$$

- Contraction (hypotheses can be used more than once)

$$\frac{\Gamma, \mathbf{A}, \mathbf{A} \vdash \mathbf{C}}{\Gamma, \mathbf{A} \vdash \mathbf{C}}$$

Natural Deduction Rules Ib

- Hypotheses: $$\overline{\Gamma, \mathbf{A}, \Delta \vdash \mathbf{A}}$$

Natural Deduction Rules Ib

- Hypotheses: $$\frac{}{\Gamma, \mathbf{A}, \Delta \vdash \mathbf{A}}$$
- Conjunction:

$$\frac{\Gamma \vdash \mathbf{A} \quad \Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \land \mathbf{B}} \land\mathsf{I} \qquad \frac{\Gamma \vdash \mathbf{A} \land \mathbf{B}}{\Gamma \vdash \mathbf{A}} \land\mathsf{E}_l \qquad \frac{\Gamma \vdash \mathbf{A} \land \mathbf{B}}{\Gamma \vdash \mathbf{B}} \land\mathsf{E}_r$$

Natural Deduction Rules Ib

- Hypotheses: $$\frac{}{\Gamma, \mathbf{A}, \Delta \vdash \mathbf{A}}$$
- Conjunction:

$$\frac{\Gamma \vdash \mathbf{A} \quad \Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \land \mathbf{B}} \land \mathrm{I} \qquad \frac{\Gamma \vdash \mathbf{A} \land \mathbf{B}}{\Gamma \vdash \mathbf{A}} \land \mathrm{E}_l \qquad \frac{\Gamma \vdash \mathbf{A} \land \mathbf{B}}{\Gamma \vdash \mathbf{B}} \land \mathrm{E}_r$$

- Disjunction: $$\frac{\Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{A} \lor \mathbf{B}} \lor \mathrm{I}_l \qquad \frac{\Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \lor \mathbf{B}} \lor \mathrm{I}_r$$

$$\frac{\Gamma \vdash \mathbf{A} \lor \mathbf{B} \quad \Gamma, \mathbf{A} \vdash \mathbf{C} \quad \Gamma, \mathbf{B} \vdash \mathbf{C}}{\Gamma \vdash \mathbf{C}} \lor \mathrm{E}_r$$

Natural Deduction Rules Ib

- Hypotheses: $$\overline{\Gamma, \mathbf{A}, \Delta \vdash \mathbf{A}}$$
- Conjunction:

$$\frac{\Gamma \vdash \mathbf{A} \quad \Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}} \wedge\mathrm{I} \qquad \frac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{A}} \wedge\mathrm{E}_l \qquad \frac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{B}} \wedge\mathrm{E}_r$$

- Disjunction:

$$\frac{\Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee\mathrm{I}_l \qquad \frac{\Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee\mathrm{I}_r$$

$$\frac{\Gamma \vdash \mathbf{A} \vee \mathbf{B} \quad \Gamma, \mathbf{A} \vdash \mathbf{C} \quad \Gamma, \mathbf{B} \vdash \mathbf{C}}{\Gamma \vdash \mathbf{C}} \vee\mathrm{E}_r$$

- Implication:

$$\frac{\Gamma, \mathbf{A} \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow\mathrm{I} \qquad \frac{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{B}} \Rightarrow\mathrm{E}$$

Natural Deduction Rules IIb

- Truth and Falsehood: $\dfrac{}{\Gamma \vdash \top}\ \top\mathrm{I} \qquad \dfrac{\Gamma \vdash \bot}{\Gamma \vdash \mathbf{C}}\ \bot\mathrm{E}$

Natural Deduction Rules IIb

- Truth and Falsehood: $$\frac{}{\Gamma \vdash \top}\ \top\mathrm{I} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash \mathbf{C}}\ \bot\mathrm{E}$$
- Negation: $$\frac{\Gamma, \mathbf{A} \vdash \bot}{\Gamma \vdash \neg\mathbf{A}}\ \neg\mathrm{I} \qquad \frac{\Gamma \vdash \neg\mathbf{A} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \bot}\ \neg\mathrm{E}$$

Natural Deduction Rules IIb

- Truth and Falsehood: $\dfrac{}{\Gamma \vdash \top}\ \top\text{I} \qquad \dfrac{\Gamma \vdash \bot}{\Gamma \vdash \mathbf{C}}\ \bot\text{E}$
- Negation: $\dfrac{\Gamma, \mathbf{A} \vdash \bot}{\Gamma \vdash \neg\mathbf{A}}\ \neg\text{I} \qquad \dfrac{\Gamma \vdash \neg\mathbf{A} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \bot}\ \neg\text{E}$
- Universal Quantif.: $\dfrac{\Gamma \vdash \mathbf{A}[\mathsf{x}/\mathsf{P}^*]}{\Gamma \vdash \forall \mathsf{x}.\mathbf{A}}\ \forall\text{I} \qquad \dfrac{\Gamma \vdash \forall \mathsf{x}.\mathbf{A}}{\Gamma \vdash \mathbf{A}[\mathsf{x}/\mathbf{T}]}\ \forall\text{E}$

(*: parameter P must be new in context)

Natural Deduction Rules IIb

- Truth and Falsehood: $\dfrac{}{\Gamma \vdash \top}\ \top I \qquad \dfrac{\Gamma \vdash \bot}{\Gamma \vdash \mathbf{C}}\ \bot E$

- Negation: $\dfrac{\Gamma, \mathbf{A} \vdash \bot}{\Gamma \vdash \neg \mathbf{A}}\ \neg I \qquad \dfrac{\Gamma \vdash \neg \mathbf{A} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \bot}\ \neg E$

- Universal Quantif.: $\dfrac{\Gamma \vdash \mathbf{A}[x/P^*]}{\Gamma \vdash \forall x. \mathbf{A}}\ \forall I \qquad \dfrac{\Gamma \vdash \forall x. \mathbf{A}}{\Gamma \vdash \mathbf{A}[x/\mathbf{T}]}\ \forall E$

  (*: parameter P must be new in context)

- Existential Quantif.:

$$\dfrac{\Gamma \vdash \mathbf{A}[x/\mathbf{T}]}{\Gamma \vdash \exists x. \mathbf{A}}\ \exists I \qquad \dfrac{\Gamma \vdash \exists x. \mathbf{A} \quad \Gamma, \mathbf{A}[x/P^*] \vdash \mathbf{C}}{\Gamma \vdash \mathbf{C}}\ \exists E$$

(*: parameter P must be new in context)

Natural Deduction Rules IIIb

For classical logic add:

- Proof by Contradiction:

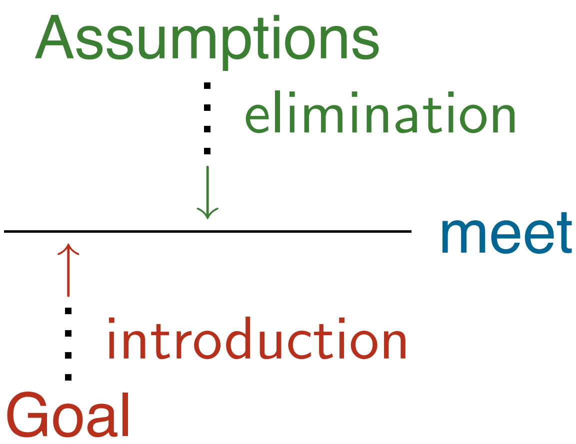
$$\frac{\Gamma, \neg \mathbf{A} \vdash \bot}{\Gamma \vdash \mathbf{A}} \; \bot_c$$

Intercalation

- Idea (Prawitz, Sieg & Scheines, Byrnes & Sieg): Detour free proofs: strictly use introduction rules bottom up (from proposed theorem to hypothesis) and elimination rules top down (from assumptions to proposed theorem). When they meet in the middle we have found a proof in normal form.

Intercalation

- Idea (Prawitz, Sieg & Scheines, Byrnes & Sieg): Detour free proofs: strictly use introduction rules bottom up (from proposed theorem to hypothesis) and elimination rules top down (from assumptions to proposed theorem). When they meet in the middle we have found a proof in normal form.



$$\begin{array}{c}\dfrac{\dfrac{\begin{matrix}\vdots & & \vdots\\ \mathbf{A} & & \mathbf{B}\end{matrix}}{\mathbf{A}\wedge\mathbf{B}}\,\wedge I}{\mathbf{A}}\,\wedge E_l\\ \vdots\end{array}$$

Intercalating Natural Deductions

- New annotations:

Intercalating Natural Deductions

- New annotations:
  - $\mathbf{A} \uparrow$: $\mathbf{A}$ is obtained by an introduction derivation

Intercalating Natural Deductions

- New annotations:
  - $\mathbf{A} \uparrow$: $\mathbf{A}$ is obtained by an introduction derivation
  - $\mathbf{A} \downarrow$: $\mathbf{A}$ is extracted from a hypothesis by an elimination derivation

Intercalating Natural Deductions

- New annotations:
  - $\mathbf{A} \uparrow$: $\mathbf{A}$ is obtained by an introduction derivation
  - $\mathbf{A} \downarrow$: $\mathbf{A}$ is extracted from a hypothesis by an elimination derivation

- Example:

$$\frac{\Gamma, \mathbf{A} \vdash_{\mathsf{ic}} \mathbf{B} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \Rightarrow \mathbf{B} \uparrow} \Rightarrow\mathsf{I} \qquad \frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \Rightarrow \mathbf{B} \downarrow \qquad \Gamma \vdash_{\mathsf{ic}} \mathbf{A} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{B} \uparrow} \Rightarrow\mathsf{E}$$

ND Intercalation Rules I

- Hypotheses: $$\overline{\Gamma, \mathbf{A}, \Delta \vdash_{\mathsf{ic}} \mathbf{A} \downarrow}$$

ND Intercalation Rules I

- Hypotheses: $$\frac{}{\Gamma, \mathbf{A}, \Delta \vdash_{ic} \mathbf{A} \downarrow}$$
- Conjunction:

$$\frac{\Gamma \vdash_{ic} \mathbf{A} \uparrow \quad \Gamma \vdash_{ic} \mathbf{B} \uparrow}{\Gamma \vdash_{ic} \mathbf{A} \wedge \mathbf{B} \uparrow} \wedge I \qquad \frac{\Gamma \vdash_{ic} \mathbf{A} \wedge \mathbf{B} \downarrow}{\Gamma \vdash_{ic} \mathbf{A} \downarrow} \wedge E_l \qquad \frac{\Gamma \vdash_{ic} \mathbf{A} \wedge \mathbf{B} \downarrow}{\Gamma \vdash_{ic} \mathbf{B} \downarrow} \wedge E_r$$

ND Intercalation Rules I

- Hypotheses: $$\overline{\Gamma, \mathbf{A}, \Delta \vdash_{\mathsf{IC}} \mathbf{A} \downarrow}$$
- Conjunction:

$$\frac{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \uparrow \quad \Gamma \vdash_{\mathsf{IC}} \mathbf{B} \uparrow}{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \wedge \mathbf{B} \uparrow} \wedge\mathrm{I} \qquad \frac{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \wedge \mathbf{B} \downarrow}{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \downarrow} \wedge\mathrm{E}_l \qquad \frac{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \wedge \mathbf{B} \downarrow}{\Gamma \vdash_{\mathsf{IC}} \mathbf{B} \downarrow} \wedge\mathrm{E}_r$$

- Disjunction:

$$\frac{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \uparrow}{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \vee \mathbf{B} \uparrow} \vee\mathrm{I}_l \qquad \frac{\Gamma \vdash_{\mathsf{IC}} \mathbf{B} \uparrow}{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \vee \mathbf{B} \uparrow} \vee\mathrm{I}_r$$

$$\frac{\Gamma \vdash_{\mathsf{IC}} \mathbf{A} \vee \mathbf{B} \downarrow \quad \Gamma, \mathbf{A} \vdash_{\mathsf{IC}} \mathbf{C} \uparrow \quad \Gamma, \mathbf{B} \vdash_{\mathsf{IC}} \mathbf{C} \uparrow}{\Gamma \vdash_{\mathsf{IC}} \mathbf{C} \uparrow} \vee\mathrm{E}$$

ND Intercalation Rules I

- Hypotheses: $$\overline{\Gamma, \mathbf{A}, \Delta \vdash_{\mathsf{ic}} \mathbf{A} \downarrow}$$
- Conjunction:

$$\frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \uparrow \quad \Gamma \vdash_{\mathsf{ic}} \mathbf{B} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \wedge \mathbf{B} \uparrow} \wedge \mathsf{I} \qquad \frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \wedge \mathbf{B} \downarrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \downarrow} \wedge \mathsf{E}_{\mathsf{l}} \qquad \frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \wedge \mathbf{B} \downarrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{B} \downarrow} \wedge \mathsf{E}_{\mathsf{r}}$$

- Disjunction:

$$\frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \vee \mathbf{B} \uparrow} \vee \mathsf{I}_{\mathsf{l}} \qquad \frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{B} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \vee \mathbf{B} \uparrow} \vee \mathsf{I}_{\mathsf{r}}$$

$$\frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \vee \mathbf{B} \downarrow \quad \Gamma, \mathbf{A} \vdash_{\mathsf{ic}} \mathbf{C} \uparrow \quad \Gamma, \mathbf{B} \vdash_{\mathsf{ic}} \mathbf{C} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{C} \uparrow} \vee \mathsf{E}$$

- Implication:

$$\frac{\Gamma, \mathbf{A} \vdash_{\mathsf{ic}} \mathbf{B} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \Rightarrow \mathbf{B} \uparrow} \Rightarrow \mathsf{I} \qquad \frac{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \Rightarrow \mathbf{B} \downarrow \quad \Gamma \vdash_{\mathsf{ic}} \mathbf{A} \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{B} \uparrow} \Rightarrow \mathsf{E}$$

ND Intercalation Rules II

- Truth and Falsehood: $$\frac{}{\Gamma \vdash_{ic} \top \uparrow} \top I \qquad \frac{\Gamma \vdash_{ic} \bot \downarrow}{\Gamma \vdash_{ic} \mathbf{C} \uparrow} \bot E$$

ND Intercalation Rules II

- Truth and Falsehood: $\dfrac{}{\Gamma \vdash_{ic} \top \uparrow}\ \top I \qquad \dfrac{\Gamma \vdash_{ic} \bot \downarrow}{\Gamma \vdash_{ic} \mathbf{C} \uparrow}\ \bot E$

- Negation: $\dfrac{\Gamma, \mathbf{A} \vdash_{ic} \bot \uparrow}{\Gamma \vdash_{ic} \neg\mathbf{A} \uparrow}\ \neg I \qquad \dfrac{\Gamma \vdash_{ic} \neg\mathbf{A} \downarrow \qquad \Gamma \vdash_{ic} \mathbf{A} \uparrow}{\Gamma \vdash_{ic} \bot \uparrow}\ \neg E$

ND Intercalation Rules II

- Truth and Falsehood: $$\frac{}{\Gamma \vdash_{ic} \top \uparrow}\ \top I \qquad \frac{\Gamma \vdash_{ic} \bot \downarrow}{\Gamma \vdash_{ic} \mathbf{C} \uparrow}\ \bot E$$
- Negation: $$\frac{\Gamma, \mathbf{A} \vdash_{ic} \bot \uparrow}{\Gamma \vdash_{ic} \neg\mathbf{A} \uparrow}\ \neg I \qquad \frac{\Gamma \vdash_{ic} \neg\mathbf{A} \downarrow \quad \Gamma \vdash_{ic} \mathbf{A} \uparrow}{\Gamma \vdash_{ic} \bot \uparrow}\ \neg E$$
- Universal Quantif.:

$$\frac{\Gamma \vdash_{ic} \mathbf{A}[x/P^*] \uparrow}{\Gamma \vdash_{ic} \forall x \mathbf{.} \mathbf{A} \uparrow}\ \forall I \qquad \frac{\Gamma \vdash_{ic} \forall x \mathbf{.} \mathbf{A} \downarrow}{\Gamma \vdash_{ic} \mathbf{A}[x/\mathbf{T}] \downarrow}\ \forall E$$

(*: parameter P must be new in context)

ND Intercalation Rules II

- Truth and Falsehood: $$\frac{}{\Gamma \vdash_{\mathrm{IC}} \top \uparrow}\ \top\mathrm{I} \qquad \frac{\Gamma \vdash_{\mathrm{IC}} \bot \downarrow}{\Gamma \vdash_{\mathrm{IC}} \mathbf{C} \uparrow}\ \bot\mathrm{E}$$
- Negation: $$\frac{\Gamma, \mathbf{A} \vdash_{\mathrm{IC}} \bot \uparrow}{\Gamma \vdash_{\mathrm{IC}} \neg\mathbf{A} \uparrow}\ \neg\mathrm{I} \qquad \frac{\Gamma \vdash_{\mathrm{IC}} \neg\mathbf{A} \downarrow \quad \Gamma \vdash_{\mathrm{IC}} \mathbf{A} \uparrow}{\Gamma \vdash_{\mathrm{IC}} \bot \uparrow}\ \neg\mathrm{E}$$
- Universal Quantif.: $$\frac{\Gamma \vdash_{\mathrm{IC}} \mathbf{A}[\mathrm{x}/\mathrm{P}^*] \uparrow}{\Gamma \vdash_{\mathrm{IC}} \forall \mathrm{x}.\mathbf{A} \uparrow}\ \forall\mathrm{I} \qquad \frac{\Gamma \vdash_{\mathrm{IC}} \forall \mathrm{x}.\mathbf{A} \downarrow}{\Gamma \vdash_{\mathrm{IC}} \mathbf{A}[\mathrm{x}/\mathbf{T}] \downarrow}\ \forall\mathrm{E}$$
  (*: parameter P must be new in context)
- Existential Quantif.: $$\frac{\Gamma \vdash_{\mathrm{IC}} \mathbf{A}[\mathrm{x}/\mathbf{T}] \uparrow}{\Gamma \vdash_{\mathrm{IC}} \exists \mathrm{x}.\mathbf{A} \uparrow}\ \exists\mathrm{I} \qquad \frac{\Gamma \vdash_{\mathrm{IC}} \exists \mathrm{x}.\mathbf{A} \downarrow \quad \Gamma, \mathbf{A}[\mathrm{x}/\mathrm{P}^*] \vdash_{\mathrm{IC}} \mathbf{C} \uparrow}{\Gamma \vdash \mathbf{C} \uparrow}\ \exists\mathrm{E}$$
  (*: parameter P must be new in context)

ND Intercalation Rules III

For classical logic add:

- Proof by Contradiction:

$$\frac{\Gamma, \neg \mathbf{A} \vdash_{\mathsf{ic}} \bot \uparrow}{\Gamma \vdash_{\mathsf{ic}} \mathbf{A} \uparrow} \bot_c$$

Intercalation and ND

- Normal form proofs

Intercalation and ND

- Normal form proofs

Assumptions

$\vdots$ elimination

$\downarrow$

$\overline{}$ meet

$\uparrow$

$\vdots$ introduction

Goal

guaranteed by

$$\frac{\Gamma \vdash_{ic} \mathbf{A} \downarrow}{\Gamma \vdash_{ic} \mathbf{A} \uparrow}\ \text{meet}$$

Intercalation and ND

- Normal form proofs

Assumptions

⋮ elimination

$\downarrow$

meet

$\uparrow$

⋮ introduction

Goal

guaranteed by $\dfrac{\Gamma \vdash_{ic} \mathbf{A} \downarrow}{\Gamma \vdash_{ic} \mathbf{A} \uparrow}$ meet

. . . proofs without detour . . .

Intercalation and ND

- Normal form proofs

Assumptions

$\vdots$ elimination

$\downarrow$

———————— meet

$\uparrow$

$\vdots$ introduction

Goal

guaranteed by $\dfrac{\Gamma \vdash_{ic} \mathbf{A} \downarrow}{\Gamma \vdash_{ic} \mathbf{A} \uparrow}$ meet

…proofs without detour …

- To model all ND proofs add $\dfrac{\Gamma \vdash_{ic} \mathbf{A} \uparrow}{\Gamma \vdash_{ic} \mathbf{A} \downarrow}$ roundabout

Example Proofs

- In normal form

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{M} \wedge \mathbf{Q} \downarrow}}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \downarrow} \wedge E_r}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \uparrow} \text{meet}}{\dfrac{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \vee \mathbf{S} \uparrow}{\vdash_{\mathsf{ic}} (\mathbf{M} \wedge \mathbf{Q}) \Rightarrow (\mathbf{Q} \vee \mathbf{S}) \uparrow} \Rightarrow I} \vee I_l
$$

Example Proofs

- In normal form

$$\cfrac{\cfrac{\cfrac{\cfrac{\overline{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{M} \wedge \mathbf{Q} \downarrow}}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \downarrow}\ \wedge E_r}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \uparrow}\ \text{meet}}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \vee \mathbf{S} \uparrow}\ \vee I_l}{\vdash_{\mathsf{ic}} (\mathbf{M} \wedge \mathbf{Q}) \Rightarrow (\mathbf{Q} \vee \mathbf{S}) \uparrow}\ \Rightarrow I$$

- With detour

$$\cfrac{\cfrac{\cfrac{\cfrac{\begin{array}{c}\vdots\\ \mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \uparrow\end{array} \qquad \begin{array}{c}\vdots\\ \mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{M} \uparrow\end{array}}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \wedge \mathbf{M} \uparrow}\ \wedge I}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \wedge \mathbf{M} \downarrow}\ \text{roundabout}}{\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \downarrow}\ \wedge E_l}{\begin{array}{c}\mathbf{M} \wedge \mathbf{Q} \vdash_{\mathsf{ic}} \mathbf{Q} \uparrow\\ \vdots\end{array}}\ \text{meet}$$

Soundness and Completeness

Let $\vdash^{\pm}_{\text{ic}}$ denote the intercalation calculus with rule roundabout and $\vdash_{\text{ic}}$ the calculus without this rule.

Soundness and Completeness

Let $\Vdash^{\pm}_{ic}$ denote the intercalation calculus with rule roundabout and $\vdash_{ic}$ the calculus without this rule.

- Theorem 1 (Soundness of $\Gamma \Vdash^{\pm}_{ic}$ relative to $\vdash$): If $\Gamma \Vdash^{\pm}_{ic} \mathbf{A}$ ↑ then $\Gamma \vdash \mathbf{A}$.

Soundness and Completeness

Let $\models^{\pm}_{\text{ic}}$ denote the intercalation calculus with rule roundabout and $\vdash_{\text{ic}}$ the calculus without this rule.

- Theorem 1 (Soundness of $\Gamma \models^{\pm}_{\text{ic}}$ relative to $\vdash$): If $\Gamma \models^{\pm}_{\text{ic}} \mathbf{A} \uparrow$ then $\Gamma \vdash \mathbf{A}$.
- Theorem 2 (Completeness of $\Gamma \models^{\pm}_{\text{ic}}$ relative to $\vdash$): If $\Gamma \vdash \mathbf{A}$ then $\Gamma \models^{\pm}_{\text{ic}} \mathbf{A} \uparrow$.

Soundness and Completeness

Let $\vdash^{\pm}_{ic}$ denote the intercalation calculus with rule roundabout and $\vdash_{ic}$ the calculus without this rule.

- Theorem 1 (Soundness of $\Gamma \vdash^{\pm}_{ic}$ relative to $\vdash$): If $\Gamma \vdash^{\pm}_{ic} \mathbf{A} \uparrow$ then $\Gamma \vdash \mathbf{A}$.
- Theorem 2 (Completeness of $\Gamma \vdash^{\pm}_{ic}$ relative to $\vdash$): If $\Gamma \vdash \mathbf{A}$ then $\Gamma \vdash^{\pm}_{ic} \mathbf{A} \uparrow$.
- Is normal form proof search also complete?:

  If $\Gamma \vdash^{\pm}_{ic} \mathbf{A} \uparrow$ then $\Gamma \vdash_{ic} \mathbf{A} \uparrow$?

  We will investigate this question within the sequent calculus.

Soundness and Completeness

Let $\models^{\pm}_{\mathrm{ic}}$ denote the intercalation calculus with rule roundabout and $\vdash_{\mathrm{ic}}$ the calculus without this rule.

- Theorem 1 (Soundness of $\Gamma \models^{\pm}_{\mathrm{ic}}$ relative to $\vdash$): If $\Gamma \models^{\pm}_{\mathrm{ic}} \mathbf{A} \uparrow$ then $\Gamma \vdash \mathbf{A}$.
- Theorem 2 (Completeness of $\Gamma \models^{\pm}_{\mathrm{ic}}$ relative to $\vdash$): If $\Gamma \vdash \mathbf{A}$ then $\Gamma \models^{\pm}_{\mathrm{ic}} \mathbf{A} \uparrow$.
- Is normal form proof search also complete?: If $\Gamma \models^{\pm}_{\mathrm{ic}} \mathbf{A} \uparrow$ then $\Gamma \vdash_{\mathrm{ic}} \mathbf{A} \uparrow$?
  We will investigate this question within the sequent calculus.

From ND to Sequent Calculus

Normal form ND proofs

Sequent proofs

From ND to Sequent Calculus

Normal form ND proofs

Assumptions

⋮ elim

$\downarrow$

——— meet

$\uparrow$

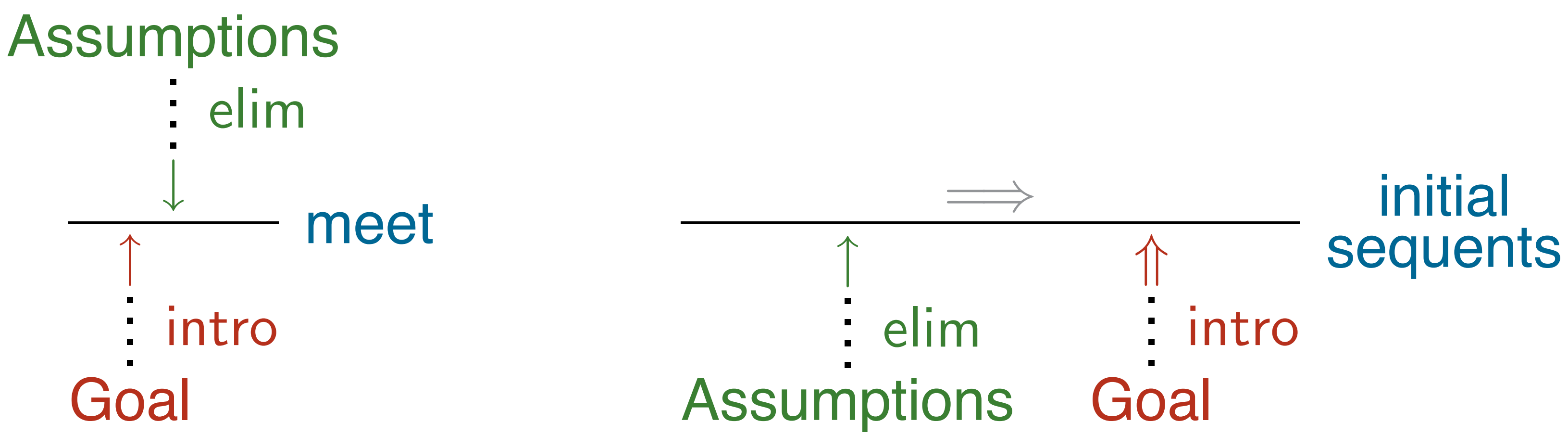
⋮ intro

Goal

Sequent proofs

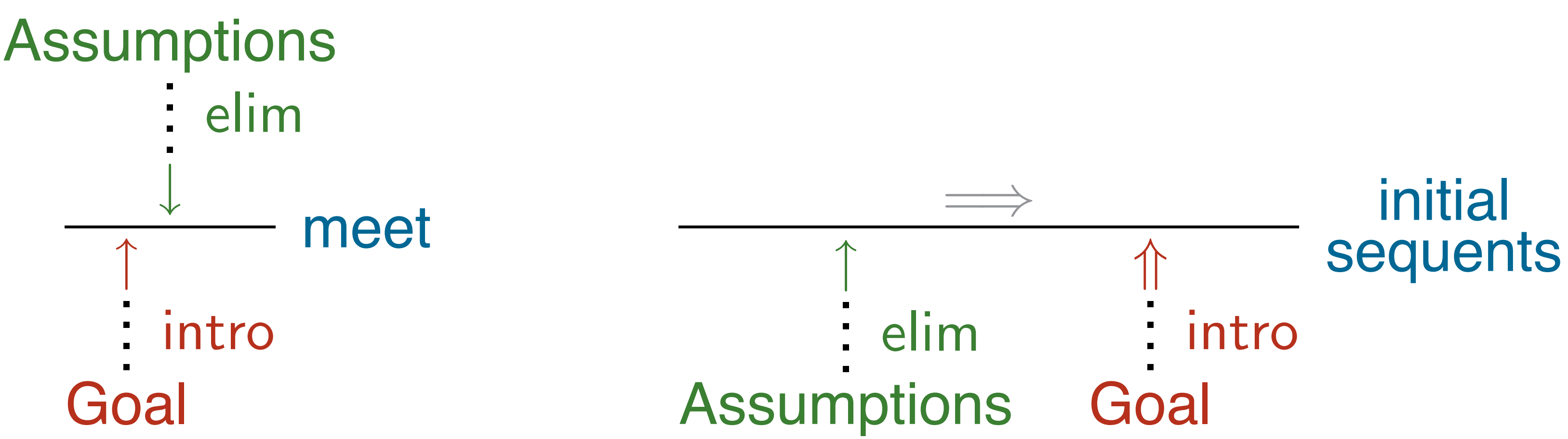
From ND to Sequent Calculus

Normal form ND proofs | Sequent proofs



From ND to Sequent Calculus

Normal form ND proofs | Sequent proofs



Sequents pair <Γ, Δ> of finite lists, multisets, or sets of formulas

Notation: $\Gamma \Longrightarrow \Delta$ Γ conjunctiv and Δ disjunctive

Intuitive: a kind of implication, Δ “follows from” Γ

Sequent Calculus Rules I

- Initial Sequents: $\overline{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{A}}\ \mathsf{init}$ ($\mathbf{A}$ atomic)

Sequent Calculus Rules I

- Initial Sequents: $$\frac{}{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{A}}\ \text{init} \qquad (\mathbf{A} \text{ atomic})$$

- Conjunction:

$$\frac{\Gamma, \mathbf{A}, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \wedge \mathbf{B} \Longrightarrow \Delta}\ \wedge L \qquad \frac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma \Longrightarrow \Delta, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \wedge \mathbf{B}}\ \wedge R$$

Sequent Calculus Rules I

- Initial Sequents: $$\overline{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{A}}\ \text{init} \qquad (\mathbf{A} \text{ atomic})$$
- Conjunction:
$$\frac{\Gamma, \mathbf{A}, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \wedge \mathbf{B} \Longrightarrow \Delta}\ \wedge\mathrm{L} \qquad \frac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma \Longrightarrow \Delta, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \wedge \mathbf{B}}\ \wedge\mathrm{R}$$
- Implication
$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \Rightarrow \mathbf{B} \Longrightarrow \Delta}\ \Rightarrow\mathrm{L} \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \Rightarrow \mathbf{B}}\ \Rightarrow\mathrm{R}$$

Sequent Calculus Rules I

- Initial Sequents: $$\frac{}{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{A}}\text{init} \qquad (\mathbf{A} \text{ atomic})$$

- Conjunction:
$$\frac{\Gamma, \mathbf{A}, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \wedge \mathbf{B} \Longrightarrow \Delta}\wedge\text{L} \qquad \frac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma \Longrightarrow \Delta, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \wedge \mathbf{B}}\wedge\text{R}$$

- Implication
$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \Rightarrow \mathbf{B} \Longrightarrow \Delta}\Rightarrow\text{L} \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \Rightarrow \mathbf{B}}\Rightarrow\text{R}$$

- Truth and Falsehood $$\frac{}{\Gamma, \bot \Longrightarrow \Delta}\bot\text{L} \qquad \frac{}{\Gamma \Longrightarrow \Delta, \top}\top\text{R}$$

Sequent Calculus Rules II

- Negation: $$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A}}{\Gamma, \neg\mathbf{A} \Longrightarrow \Delta}\ \neg\mathrm{L} \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta, \neg\mathbf{A}}\ \neg\mathrm{R}$$

Sequent Calculus Rules II

- Negation:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A}}{\Gamma, \neg\mathbf{A} \Longrightarrow \Delta}\ \neg\mathrm{L} \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta, \neg\mathbf{A}}\ \neg\mathrm{R}$$

- Disjunction:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A}, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \vee \mathbf{B}}\ \vee\mathrm{R} \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta \quad \Gamma, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \vee \mathbf{B} \Longrightarrow \Delta}\ \vee\mathrm{L}$$

Sequent Calculus Rules II

- Negation:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A}}{\Gamma, \neg\mathbf{A} \Longrightarrow \Delta}\ \neg L \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta, \neg\mathbf{A}}\ \neg R$$

- Disjunction:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A}, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \vee \mathbf{B}}\ \vee R \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta \quad \Gamma, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \vee \mathbf{B} \Longrightarrow \Delta}\ \vee L$$

- Universal Quantification:

$$\frac{\Gamma, \forall x_{\blacksquare}\mathbf{A}, \mathbf{A}[x/\mathbf{T}] \Longrightarrow \Delta}{\Gamma, \forall x_{\blacksquare}\mathbf{A} \Longrightarrow \Delta}\ \forall L \qquad \frac{\Gamma \Longrightarrow \Delta, \mathbf{A}[x/P^*]}{\Gamma \Longrightarrow \Delta, \forall x_{\blacksquare}\mathbf{A}}\ \forall R$$

(*: parameter P must be new in context)

Sequent Calculus Rules II

- Negation:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A}}{\Gamma, \neg\mathbf{A} \Longrightarrow \Delta}\ \neg\mathrm{L} \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta, \neg\mathbf{A}}\ \neg\mathrm{R}$$

- Disjunction:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A}, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \vee \mathbf{B}}\ \vee\mathrm{R} \qquad \frac{\Gamma, \mathbf{A} \Longrightarrow \Delta \quad \Gamma, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \vee \mathbf{B} \Longrightarrow \Delta}\ \vee\mathrm{L}$$

- Universal Quantification:

$$\frac{\Gamma, \forall x\text{.}\mathbf{A}, \mathbf{A}[x/\mathbf{T}] \Longrightarrow \Delta}{\Gamma, \forall x\text{.}\mathbf{A} \Longrightarrow \Delta}\ \forall\mathrm{L} \qquad \frac{\Gamma \Longrightarrow \Delta, \mathbf{A}[x/P^*]}{\Gamma \Longrightarrow \Delta, \forall x\text{.}\mathbf{A}}\ \forall\mathrm{R}$$

(*: parameter P must be new in context)

- Existential Quantification:

$$\frac{\Gamma, \mathbf{A}[x/P^*] \Longrightarrow \Delta}{\Gamma, \exists x\text{.}\mathbf{A} \Longrightarrow \Delta}\ \exists\mathrm{L} \qquad \frac{\Gamma \Longrightarrow \Delta, \exists x\text{.}\mathbf{A}, \mathbf{A}[x/\mathbf{T}]}{\Gamma \Longrightarrow \Delta, \exists x\text{.}\mathbf{A}}\ \exists\mathrm{R}$$

(*: parameter P must be new in context)

Example Proof

$$
\dfrac{
\dfrac{\dfrac{}{\mathbf{A},\mathbf{B} \Longrightarrow \mathbf{B}}\ \textsf{init}}{\mathbf{A}\wedge\mathbf{B} \Longrightarrow \mathbf{B}}\ \wedge\textsf{L}
\qquad
\dfrac{\dfrac{\dfrac{}{\mathbf{A},\mathbf{B} \Longrightarrow \mathbf{C},\mathbf{A}}\ \textsf{init}}{\mathbf{A}\wedge\mathbf{B} \Longrightarrow \mathbf{C},\mathbf{A}}\ \wedge\textsf{L}}{\mathbf{A}\wedge\mathbf{B} \Longrightarrow \mathbf{C}\vee\mathbf{A}}\ \vee\textsf{R}
}{
\dfrac{\mathbf{A}\wedge\mathbf{B} \Longrightarrow \mathbf{B}\wedge(\mathbf{C}\vee\mathbf{A})}{\Longrightarrow (\mathbf{A}\wedge\mathbf{B}) \Rightarrow \mathbf{B}\wedge(\mathbf{C}\vee\mathbf{A})}\ \Rightarrow\textsf{R}
}\ \wedge\textsf{R}
$$

Sequent Calculus: Cut-rule

- To map natural deductions (in $\vdash$ and $\Vdash^{\pm}_{ic}$) to sequent calculus derivations we add the so called cut-rule:

Sequent Calculus: Cut-rule

- To map natural deductions (in $\vdash$ and $\Vdash^{\pm}_{ic}$) to sequent calculus derivations we add the so called cut-rule:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta}\ \mathrm{Cut}$$

Sequent Calculus: Cut-rule

- To map natural deductions (in $\vdash$ and $\vdash^{\pm}_{ic}$) to sequent calculus derivations we add the so called cut-rule:

$$\frac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta} \text{ Cut}$$

- The question whether normal form proof search ($\vdash_{ic}$) is complete corresponds to the question whether the cut-rule can be eliminated (is *admissible*) in sequent calculus.

Sequent Calculus

Let $\Longrightarrow^{+}$ denote the sequent calculus with cut-rule and $\Longrightarrow$ the sequent calculus without the cut-rule.

Sequent Calculus

Let $\Longrightarrow^{+}$ denote the sequent calculus with cut-rule and $\Longrightarrow$ the sequent calculus without the cut-rule.

Theorem 3 (Soundness of $\Longrightarrow$ relative to $\Gamma \vdash_{ic}$ and $\Gamma \Vdash^{\pm}_{ic}$)

(a) If $\Gamma \Longrightarrow \mathbf{C}$ then $\Gamma \vdash_{ic} \mathbf{C} \uparrow$.

(b) If $\Gamma \Longrightarrow^{+} \mathbf{C}$ then $\Gamma \Vdash^{\pm}_{ic} \mathbf{C} \uparrow$.

Sequent Calculus

Let $\Longrightarrow^{+}$ denote the sequent calculus with cut-rule and $\Longrightarrow$ the sequent calculus without the cut-rule.

Theorem 3 (Soundness of $\Longrightarrow$ relative to $\Gamma \vdash_{\text{ic}}$ and $\Gamma \Vdash^{\pm}_{\text{ic}}$)

(a) If $\Gamma \Longrightarrow \mathbf{C}$ then $\Gamma \vdash_{\text{ic}} \mathbf{C} \uparrow$.

(b) If $\Gamma \Longrightarrow^{+} \mathbf{C}$ then $\Gamma \Vdash^{\pm}_{\text{ic}} \mathbf{C} \uparrow$.

Theorem 4 (Completeness of $\Longrightarrow$ relative to $\Gamma \vdash_{\text{ic}}$ and $\Gamma \Vdash^{\pm}_{\text{ic}}$)

(a) If $\Gamma \vdash_{\text{ic}} \mathbf{C} \uparrow$ then $\Gamma \Longrightarrow \mathbf{C}$.

(b) If $\Gamma \Vdash^{\pm}_{\text{ic}} \mathbf{C} \uparrow$ then $\Gamma \Longrightarrow^{+} \mathbf{C}$.

Gentzen's Hauptsatz

Theorem 5 (Cut-Elimination): Cut-elimination holds for the sequent calculus. In other words: The cut rule is *admissible* in the sequent calculus.

Gentzen's Hauptsatz

Theorem 5 (Cut-Elimination): Cut-elimination holds for the sequent calculus. In other words: The cut rule is *admissible* in the sequent calculus.

$$\text{If } \Gamma \Longrightarrow^{+} \Delta \text{ then } \Gamma \Longrightarrow \Delta$$

Gentzen's Hauptsatz

Theorem 5 (Cut-Elimination): Cut-elimination holds for the sequent calculus. In other words: The cut rule is *admissible* in the sequent calculus.

$$\text{If } \Gamma \Longrightarrow^{+} \Delta \text{ then } \Gamma \Longrightarrow \Delta$$

Proof non-trivial; main means: nested inductions and case distinctions over rule applications

Gentzen's Hauptsatz

Theorem 5 (Cut-Elimination): Cut-elimination holds for the sequent calculus. In other words: The cut rule is *admissible* in the sequent calculus.

$$\text{If } \Gamma \Longrightarrow^{+} \Delta \text{ then } \Gamma \Longrightarrow \Delta$$

Proof non-trivial; main means: nested inductions and case distinctions over rule applications

This result qualifies the sequent calculus as suitable for automating proof search.

Applications of Cut-Elimination

Theorem (Normalization for ND):

If $\Gamma \vdash \mathbf{C}$ then $\Gamma \vdash_{ic} \mathbf{C}\uparrow$.

Applications of Cut-Elimination

Theorem (Normalization for ND):

If $\Gamma \vdash \mathbf{C}$ then $\Gamma \vdash_{ic} \mathbf{C} \uparrow$.

Proof sketch:

Applications of Cut-Elimination

Theorem (Normalization for ND):

$$\text{If } \Gamma \vdash \mathbf{C} \text{ then } \Gamma \vdash_{ic} \mathbf{C} \uparrow .$$

Proof sketch:

- Assume $\Gamma \vdash \mathbf{C}$.

Applications of Cut-Elimination

Theorem (Normalization for ND):

If $\Gamma \vdash \mathbf{C}$ then $\Gamma \vdash_{ic} \mathbf{C} \uparrow$.

Proof sketch:

- Assume $\Gamma \vdash \mathbf{C}$.
- Then $\Gamma \models^{\pm}_{ic} \mathbf{C} \uparrow$ by completeness of $\models^{\pm}_{ic}$.

Applications of Cut-Elimination

Theorem (Normalization for ND):

If $\Gamma \vdash \mathbf{C}$ then $\Gamma \vdash_{\mathrm{ic}} \mathbf{C} \uparrow$.

Proof sketch:

- Assume $\Gamma \vdash \mathbf{C}$.
- Then $\Gamma \Vdash^{\pm}_{\mathrm{ic}} \mathbf{C} \uparrow$ by completeness of $\Vdash^{\pm}_{\mathrm{ic}}$.
- Then $\Gamma \Longrightarrow^{+} \mathbf{C}$ by completeness of $\Longrightarrow^{+}$.

Applications of Cut-Elimination

Theorem (Normalization for ND):

If $\Gamma \vdash \mathbf{C}$ then $\Gamma \vdash_{\text{ic}} \mathbf{C} \uparrow$.

Proof sketch:

- Assume $\Gamma \vdash \mathbf{C}$.
- Then $\Gamma \Vdash^{\pm}_{\text{ic}} \mathbf{C} \uparrow$ by completeness of $\Vdash^{\pm}_{\text{ic}}$.
- Then $\Gamma \Longrightarrow^{+} \mathbf{C}$ by completeness of $\Longrightarrow^{+}$.
- Then $\Gamma \Longrightarrow \mathbf{C}$ by cut-elimination.

Applications of Cut-Elimination

Theorem (Normalization for ND):

If $\Gamma \vdash \mathbf{C}$ then $\Gamma \vdash_{\mathrm{ic}} \mathbf{C} \uparrow$.

Proof sketch:

- Assume $\Gamma \vdash \mathbf{C}$.
- Then $\Gamma \Vdash^{\pm}_{\mathrm{ic}} \mathbf{C} \uparrow$ by completeness of $\Vdash^{\pm}_{\mathrm{ic}}$.
- Then $\Gamma \Longrightarrow^{+} \mathbf{C}$ by completeness of $\Longrightarrow^{+}$.
- Then $\Gamma \Longrightarrow \mathbf{C}$ by cut-elimination.
- Then $\Gamma \vdash_{\mathrm{ic}} \mathbf{C} \uparrow$ by soundness of $\Longrightarrow$.

What have we done?

Natural Deduction		
$\vdash$ (with detours) $\longrightarrow$		

What have we done?

Natural Deduction	Intercalation	
$\vdash$ (with detours) $\longrightarrow$	$\longrightarrow$ $\vdash^{\pm}_{ic}$ (with roundabout) $\longrightarrow$	

What have we done?

Natural Deduction	Intercalation	Sequent Calculus
$\vdash$ (with detours) $\longrightarrow$	$\longrightarrow$ $\vdash^{\pm}_{ic}$ (with roundabout) $\longrightarrow$	$\longrightarrow$ $\Longrightarrow^{+}$ (with cut) $\downarrow$

What have we done?

Natural Deduction	Intercalation	Sequent Calculus
$\vdash$ (with detours) $\longrightarrow$	$\longrightarrow$ $\vdash^{\pm}_{ic}$ (with roundabout) $\longrightarrow$	$\longrightarrow$ $\Longrightarrow^{+}$ (with cut) $\downarrow$
		$\longleftarrow$ $\longleftarrow$ $\Longrightarrow$ (without cut)

What have we done?

Natural Deduction	Intercalation	Sequent Calculus
$\vdash$ (with detours) $\longrightarrow$	$\longrightarrow$ $\vdash^{+}_{ic}$ (with roundabout) $\longrightarrow$	$\longrightarrow$ $\Longrightarrow^{+}$ (with cut) $\downarrow$
	$\longleftarrow$ $\longleftarrow$ $\vdash_{ic}$ (without roundabout)	$\longleftarrow$ $\longleftarrow$ $\Longrightarrow$ (without cut)

What have we done?

Natural Deduction	Intercalation	Sequent Calculus
$\vdash$ (with detours) $\longrightarrow$	$\longrightarrow$ $\vdash^{+}_{ic}$ (with roundabout) $\longrightarrow$	$\longrightarrow$ $\Longrightarrow^{+}$ (with cut) $\downarrow$
$\longleftarrow$ $\vdash$ (without detours)	$\longleftarrow$ $\longleftarrow$ $\vdash_{ic}$ (without roundabout)	$\longleftarrow$ $\longleftarrow$ $\Longrightarrow$ (without cut)

Applications of Cut-Elimination

Theorem (Consistency of ND): There is no natural deduction derivation $\vdash \bot$.

Applications of Cut-Elimination

Theorem (Consistency of ND): There is no natural deduction derivation $\vdash \bot$.

Proof sketch:

Applications of Cut-Elimination

Theorem (Consistency of ND): There is no natural deduction derivation $\vdash \bot$.

Proof sketch:

- Assume there is a proof of $\vdash \bot$.

Applications of Cut-Elimination

Theorem (Consistency of ND): There is no natural deduction derivation $\vdash \bot$.

Proof sketch:

- Assume there is a proof of $\vdash \bot$.
- Then $\Longrightarrow^{+} \bot$ by completeness of $\Longrightarrow^{+}$ and $\models^{+}_{\mathrm{ic}}$.

Applications of Cut-Elimination

Theorem (Consistency of ND): There is no natural deduction derivation $\vdash \bot$.

Proof sketch:

- Assume there is a proof of $\vdash \bot$.
- Then $\Longrightarrow^{+} \bot$ by completeness of $\Longrightarrow^{+}$ and $\models^{\pm}_{ic}$.
- Then $\Longrightarrow \bot$ by cut-elimination.

Applications of Cut-Elimination

Theorem (Consistency of ND): There is no natural deduction derivation $\vdash \bot$.

Proof sketch:

- Assume there is a proof of $\vdash \bot$.
- Then $\Longrightarrow^{+} \bot$ by completeness of $\Longrightarrow^{+}$ and $\models^{\pm}_{ic}$.
- Then $\Longrightarrow \bot$ by cut-elimination.
- But $\Longrightarrow \bot$ cannot be the conclusion of any sequent rule.

Summary

- We have illustrated the connection of

Summary

- We have illustrated the connection of
  - natural deduction and sequent calculus

Summary

- We have illustrated the connection of
  - natural deduction and sequent calculus
  - normal form natural deductions and cut-free sequent calculus.

Summary

- We have illustrated the connection of
  - natural deduction and sequent calculus
  - normal form natural deductions and cut-free sequent calculus.
- Fact: Sequent calculus often employed as meta-theory for specialized proof search calculi and strategies.

Summary

- We have illustrated the connection of
  - natural deduction and sequent calculus
  - normal form natural deductions and cut-free sequent calculus.
- Fact: Sequent calculus often employed as meta-theory for specialized proof search calculi and strategies.
- Question: Can these calculi and strategies be transformed to natural deduction proof search?

Calculi: Higher-Order Natural Deduction

ND Calculi for HOL

Some conventions for this part:

- signature Σ contains only the logical constants $\neg, \vee, \Pi^{\alpha}$ unless stated otherwise.

ND Calculi for HOL

Some conventions for this part:

- signature Σ contains only the logical constants $\neg, \vee, \Pi^{\alpha}$ unless stated otherwise.
- $\Phi * \mathrm{A} := \Phi \cup \{\mathbf{A}\}$

ND Calculi for HOL

Some conventions for this part:

- signature Σ contains only the logical constants $\neg, \vee, \Pi^{\alpha}$ unless stated otherwise.
- $\Phi * \mathrm{A} := \Phi \cup \{\mathbf{A}\}$
- context representation of ND calculi

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\beta)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\,\mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg\mathbf{A}}\,\mathfrak{NK}(\neg I)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}} \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}} \mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg \mathbf{A}} \mathfrak{NK}(\neg I) \qquad \frac{\Phi \Vdash \neg \mathbf{A} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{C}} \mathfrak{NK}(\neg E)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg \mathbf{A}}\ \mathfrak{NK}(\neg I) \qquad \frac{\Phi \Vdash \neg \mathbf{A} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\neg E)$$

$$\frac{\Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_L)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg\mathbf{A}}\ \mathfrak{NK}(\neg I) \qquad \frac{\Phi \Vdash \neg\mathbf{A} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\neg E)$$

$$\frac{\Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_L) \qquad \frac{\Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_R)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg\mathbf{A}}\ \mathfrak{NK}(\neg I) \qquad \frac{\Phi \Vdash \neg\mathbf{A} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\neg E)$$

$$\frac{\Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_L) \qquad \frac{\Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_R)$$

$$\frac{\Phi \Vdash \mathbf{A} \vee \mathbf{B} \quad \Phi * \mathbf{A} \Vdash \mathbf{C} \quad \Phi * \mathbf{B} \Vdash \mathbf{C}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\vee E)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg \mathbf{A}}\ \mathfrak{NK}(\neg I) \qquad \frac{\Phi \Vdash \neg \mathbf{A} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\neg E)$$

$$\frac{\Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_L) \qquad \frac{\Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_R)$$

$$\frac{\Phi \Vdash \mathbf{A} \vee \mathbf{B} \quad \Phi * \mathbf{A} \Vdash \mathbf{C} \quad \Phi * \mathbf{B} \Vdash \mathbf{C}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\vee E)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathsf{w}_\alpha \quad \mathsf{w}\ \text{new parameter}}{\Phi \Vdash \Pi^\alpha \mathbf{G}}\ \mathfrak{NK}(\Pi I)^{\mathsf{w}}$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg \mathbf{A}}\ \mathfrak{NK}(\neg I) \qquad \frac{\Phi \Vdash \neg \mathbf{A} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\neg E)$$

$$\frac{\Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_L) \qquad \frac{\Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_R)$$

$$\frac{\Phi \Vdash \mathbf{A} \vee \mathbf{B} \quad \Phi * \mathbf{A} \Vdash \mathbf{C} \quad \Phi * \mathbf{B} \Vdash \mathbf{C}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\vee E)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathsf{w}_\alpha \quad \mathsf{w} \text{ new parameter}}{\Phi \Vdash \Pi^\alpha \mathbf{G}}\ \mathfrak{NK}(\Pi I)^{\mathsf{w}}$$

$$\frac{\Phi \Vdash \Pi^\alpha \mathbf{G}}{\Phi \Vdash \mathbf{G}\mathbf{A}}\ \mathfrak{NK}(\Pi E)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$

$$\frac{\mathbf{A} \in \Phi}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Hyp) \qquad \frac{\mathbf{A} =_\beta \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\beta)$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \neg \mathbf{A}}\ \mathfrak{NK}(\neg I) \qquad \frac{\Phi \Vdash \neg \mathbf{A} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\neg E)$$

$$\frac{\Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_L) \qquad \frac{\Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \vee \mathbf{B}}\ \mathfrak{NK}(\vee I_R)$$

$$\frac{\Phi \Vdash \mathbf{A} \vee \mathbf{B} \quad \Phi * \mathbf{A} \Vdash \mathbf{C} \quad \Phi * \mathbf{B} \Vdash \mathbf{C}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\vee E)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathsf{w}_\alpha \quad \text{w new parameter}}{\Phi \Vdash \Pi^\alpha \mathbf{G}}\ \mathfrak{NK}(\Pi I)^{\mathsf{w}}$$

$$\frac{\Phi \Vdash \Pi^\alpha \mathbf{G}}{\Phi \Vdash \mathbf{G}\mathbf{A}}\ \mathfrak{NK}(\Pi E) \qquad \frac{\Phi * \neg \mathbf{A} \Vdash \mathbf{F}_o}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(Contr)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(\land E_L)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(\wedge E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\wedge E_R)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{A}}\,\mathfrak{NK}(\land E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\land E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \land \mathbf{B}}\,\mathfrak{NK}(\land I)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(\land E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\land E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \land \mathbf{B}}\ \mathfrak{NK}(\land I)$$

$$\frac{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\Rightarrow E)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{A}}\,\mathfrak{NK}(\land E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\land E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \land \mathbf{B}}\,\mathfrak{NK}(\land I)$$

$$\frac{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\Rightarrow E) \qquad \frac{\Phi, \mathbf{A} \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B}}\,\mathfrak{NK}(\Rightarrow I)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(\wedge E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\wedge E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}\ \mathfrak{NK}(\wedge I)$$

$$\frac{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\Rightarrow E) \qquad \frac{\Phi, \mathbf{A} \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B}}\ \mathfrak{NK}(\Rightarrow I)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathbf{T}_\alpha}{\Phi \Vdash \Sigma^\alpha \mathbf{G}}\ \mathfrak{NK}(\Sigma I)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{A}}\,\mathfrak{NK}(\wedge E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\wedge E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}\,\mathfrak{NK}(\wedge I)$$

$$\frac{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\Rightarrow E) \qquad \frac{\Phi, \mathbf{A} \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B}}\,\mathfrak{NK}(\Rightarrow I)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathbf{T}_\alpha}{\Phi \Vdash \Sigma^\alpha \mathbf{G}}\,\mathfrak{NK}(\Sigma I) \qquad \frac{\Phi \Vdash \Sigma^\alpha \mathbf{G} \quad \Phi * \mathbf{G}\mathrm{w}_\alpha \Vdash \mathbf{C} \quad \text{w new parameter}}{\Phi \Vdash \mathbf{C}}\,\mathfrak{NK}(\Sigma E)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(\wedge E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\wedge E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}\ \mathfrak{NK}(\wedge I)$$

$$\frac{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\Rightarrow E) \qquad \frac{\Phi, \mathbf{A} \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B}}\ \mathfrak{NK}(\Rightarrow I)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathbf{T}_\alpha}{\Phi \Vdash \Sigma^\alpha \mathbf{G}}\ \mathfrak{NK}(\Sigma I) \qquad \frac{\Phi \Vdash \Sigma^\alpha \mathbf{G} \quad \Phi * \mathbf{G}\mathsf{w}_\alpha \Vdash \mathbf{C} \quad \text{w new parameter}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\Sigma E)$$

$$\frac{\Phi \Vdash \mathbf{T} =^\alpha \mathbf{W} \quad \Phi \Vdash \mathbf{A}[\mathbf{T}]}{\Phi \Vdash \mathbf{A}[\mathbf{W}]}\ \mathfrak{NK}(= Subst)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{A}}\ \mathfrak{NK}(\wedge E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\wedge E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \wedge \mathbf{B}}\ \mathfrak{NK}(\wedge I)$$

$$\frac{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\Rightarrow E) \qquad \frac{\Phi, \mathbf{A} \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B}}\ \mathfrak{NK}(\Rightarrow I)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathbf{T}_\alpha}{\Phi \Vdash \Sigma^\alpha \mathbf{G}}\ \mathfrak{NK}(\Sigma I) \qquad \frac{\Phi \Vdash \Sigma^\alpha \mathbf{G} \quad \Phi * \mathbf{G}\mathrm{w}_\alpha \Vdash \mathbf{C} \quad \text{w new parameter}}{\Phi \Vdash \mathbf{C}}\ \mathfrak{NK}(\Sigma E)$$

$$\frac{\Phi \Vdash \mathbf{T} =^\alpha \mathbf{W} \quad \Phi \Vdash \mathbf{A}[\mathbf{T}]}{\Phi \Vdash \mathbf{A}[\mathbf{W}]}\ \mathfrak{NK}(=Subst) \qquad \frac{}{\Phi \Vdash \mathbf{A} = \mathbf{A}}\ \mathfrak{NK}(=Refl)$$

ND Calculi for HOL

Inference rules for $\mathfrak{NK}_\beta$ (for richer signatures)

$$\frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{A}}\,\mathfrak{NK}(\land E_L) \qquad \frac{\Phi \Vdash \mathbf{A} \land \mathbf{B}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\land E_R) \qquad \frac{\Phi \Vdash \mathbf{A} \quad \Phi \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \land \mathbf{B}}\,\mathfrak{NK}(\land I)$$

$$\frac{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\Rightarrow E) \qquad \frac{\Phi, \mathbf{A} \Vdash \mathbf{B}}{\Phi \Vdash \mathbf{A} \Rightarrow \mathbf{B}}\,\mathfrak{NK}(\Rightarrow I)$$

$$\frac{\Phi \Vdash \mathbf{G}\mathbf{T}_\alpha}{\Phi \Vdash \Sigma^\alpha \mathbf{G}}\,\mathfrak{NK}(\Sigma I) \qquad \frac{\Phi \Vdash \Sigma^\alpha \mathbf{G} \quad \Phi * \mathbf{G}\mathsf{w}_\alpha \Vdash \mathbf{C} \quad \mathsf{w}\text{ new parameter}}{\Phi \Vdash \mathbf{C}}\,\mathfrak{NK}(\Sigma E)$$

$$\frac{\Phi \Vdash \mathbf{T} =^\alpha \mathbf{W} \quad \Phi \Vdash \mathbf{A}[\mathbf{T}]}{\Phi \Vdash \mathbf{A}[\mathbf{W}]}\,\mathfrak{NK}(=Subst) \qquad \frac{}{\Phi \Vdash \mathbf{A} = \mathbf{A}}\,\mathfrak{NK}(=Refl)$$

Alternative: Define logical constants $\land, \Rightarrow, \Sigma$, etc. in terms of $\neg, \lor, \Pi$ as usual and strictly use Leibniz equality instead of primitive equality; then the above rules are not needed.

ND Calculi for HOL

Inference rules for extensionality (rules for $\xi, \eta, \mathfrak{f}, \mathfrak{b}$)

$$\frac{\mathbf{A} \stackrel{\beta\eta}{=} \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}} \mathfrak{NK}(\eta)$$

ND Calculi for HOL

Inference rules for extensionality (rules for $\xi, \eta, \mathfrak{f}, \mathfrak{b}$)

$$\frac{\mathbf{A} \stackrel{\beta\eta}{=} \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}} \mathfrak{NK}(\eta) \qquad \frac{\Phi \Vdash \forall \mathsf{x}_\alpha\text{.}\mathbf{M} \doteq^{\beta} \mathbf{N}}{\Phi \Vdash (\lambda \mathsf{x}_\alpha\text{.}\mathbf{M}) \doteq^{\beta\alpha} (\lambda \mathsf{x}_\alpha\text{.}\mathbf{N})} \mathfrak{NK}(\xi)$$

ND Calculi for HOL

Inference rules for extensionality (rules for $\xi, \eta, \mathfrak{f}, \mathfrak{b}$)

$$\frac{\mathbf{A} \overset{\beta\eta}{=} \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}} \mathfrak{NK}(\eta) \qquad \frac{\Phi \Vdash \forall \mathsf{x}_\alpha \blacksquare \mathbf{M} \doteq^{\beta} \mathbf{N}}{\Phi \Vdash (\lambda \mathsf{x}_\alpha \blacksquare \mathbf{M}) \doteq^{\beta\alpha} (\lambda \mathsf{x}_\alpha \blacksquare \mathbf{N})} \mathfrak{NK}(\xi)$$

$$\frac{\Phi \Vdash \forall \mathsf{x}_\alpha \blacksquare \mathbf{G}\mathsf{x} \doteq^{\beta} \mathbf{H}\mathsf{x}}{\Phi \Vdash \mathbf{G} \doteq^{\beta\alpha} \mathbf{H}} \mathfrak{NK}(\mathfrak{f})$$

ND Calculi for HOL

Inference rules for extensionality (rules for $\xi, \eta, \mathfrak{f}, \mathfrak{b}$)

$$\frac{\mathbf{A} \stackrel{\beta\eta}{=} \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\ \mathfrak{NK}(\eta) \qquad \frac{\Phi \Vdash \forall \mathsf{x}_\alpha \text{.} \mathbf{M} \doteq^\beta \mathbf{N}}{\Phi \Vdash (\lambda \mathsf{x}_\alpha \text{.} \mathbf{M}) \doteq^{\beta\alpha} (\lambda \mathsf{x}_\alpha \text{.} \mathbf{N})}\ \mathfrak{NK}(\xi)$$

$$\frac{\Phi \Vdash \forall \mathsf{x}_\alpha \text{.} \mathbf{G}\mathsf{x} \doteq^\beta \mathbf{H}\mathsf{x}}{\Phi \Vdash \mathbf{G} \doteq^{\beta\alpha} \mathbf{H}}\ \mathfrak{NK}(\mathfrak{f})$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{B} \quad \Phi * \mathbf{B} \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \doteq^{o} \mathbf{B}}\ \mathfrak{NK}(\mathfrak{b})$$

ND Calculi for HOL

Inference rules for extensionality (rules for $\xi, \eta, \mathfrak{f}, \mathfrak{b}$)

$$\frac{\mathbf{A} \stackrel{\beta\eta}{=} \mathbf{B} \quad \Phi \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{B}}\,\mathfrak{NK}(\eta) \qquad \frac{\Phi \Vdash \forall \mathsf{x}_\alpha\mathbf{.M} \doteq^\beta \mathbf{N}}{\Phi \Vdash (\lambda \mathsf{x}_\alpha\mathbf{.M}) \doteq^{\beta\alpha} (\lambda \mathsf{x}_\alpha\mathbf{.N})}\,\mathfrak{NK}(\xi)$$

$$\frac{\Phi \Vdash \forall \mathsf{x}_\alpha\mathbf{.G}\mathsf{x} \doteq^\beta \mathbf{H}\mathsf{x}}{\Phi \Vdash \mathbf{G} \doteq^{\beta\alpha} \mathbf{H}}\,\mathfrak{NK}(\mathfrak{f})$$

$$\frac{\Phi * \mathbf{A} \Vdash \mathbf{B} \quad \Phi * \mathbf{B} \Vdash \mathbf{A}}{\Phi \Vdash \mathbf{A} \doteq^{o} \mathbf{B}}\,\mathfrak{NK}(\mathfrak{b})$$

In case of a primitive notion of equality we define respective extensionality rules also for $=$.

ND Calculi for HOL

- The Calculi $\mathfrak{NK}_*$

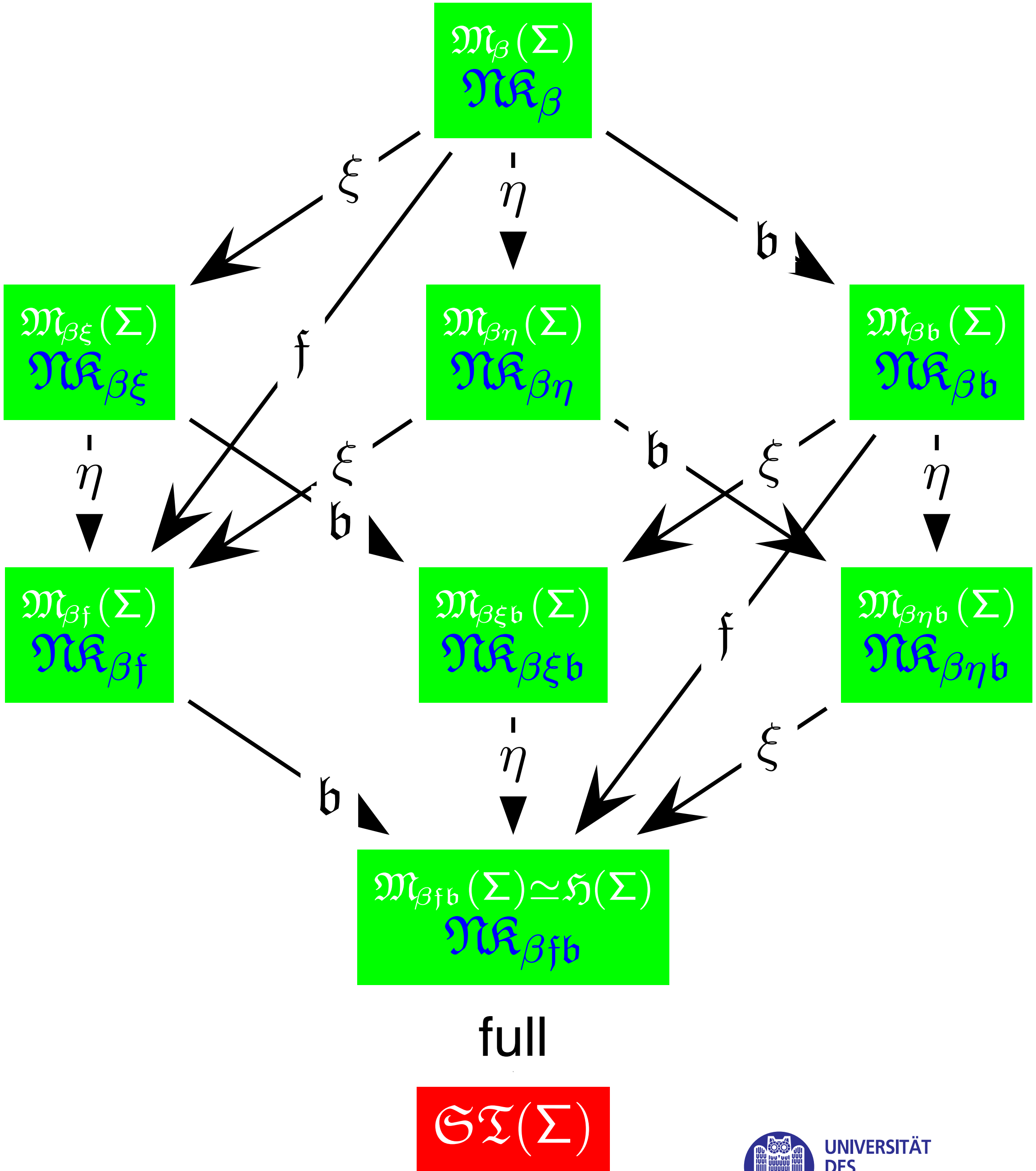
ND Calculi for HOL

- The Calculi $\mathfrak{NK}_*$
  - The calculus $\mathfrak{NK}_\beta$ consists of the inference rules for $\mathfrak{NK}_\beta$ for the provability judgment $\Vdash$ between sets of sentences Φ and sentences $\mathbf{A}$. (We write $\Vdash \mathbf{A}$ for $\emptyset \Vdash \mathbf{A}$.) The rule $\mathfrak{NK}(\beta)$ incorporates β-equality into $\Vdash$. The others characterize ‘the semantics of the connectives and quantifiers.

ND Calculi for HOL

- The Calculi $\mathfrak{NK}_*$
  - The calculus $\mathfrak{NK}_\beta$ consists of the inference rules for $\mathfrak{NK}_\beta$ for the provability judgment $\Vdash$ between sets of sentences Φ and sentences $\mathbf{A}$. (We write $\Vdash \mathbf{A}$ for $\emptyset \Vdash \mathbf{A}$.) The rule $\mathfrak{NK}(\beta)$ incorporates β-equality into $\Vdash$. The others characterize 'the semantics of the connectives and quantifiers.
  - For $* \in \{\beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$ we obtain the calculus $\mathfrak{NK}_*$ by adding the respective extensionality rules when specified in $*$.

ND Calculi for HOL



ND Calculi for HOL

- Note that $\mathfrak{NK}_\beta$ and $\mathfrak{NK}_{\beta\mathfrak{fb}}$ correspond to the extremes of the model classes in our landscape of model classes. For example, $\mathfrak{NK}_{\beta\mathfrak{fb}}$ will be proven sound and complete for Henkin models, and $\mathfrak{NK}_\beta$ will be proven sound and complete for $\mathfrak{M}_\beta(\Sigma)$.

ND Calculi for HOL

- Note that $\mathfrak{NK}_\beta$ and $\mathfrak{NK}_{\beta\mathfrak{fb}}$ correspond to the extremes of the model classes in our landscape of model classes. For example, $\mathfrak{NK}_{\beta\mathfrak{fb}}$ will be proven sound and complete for Henkin models, and $\mathfrak{NK}_\beta$ will be proven sound and complete for $\mathfrak{M}_\beta(\Sigma)$.
- Standard models do not admit (recursively axiomatizable) calculi that are sound and complete.

ND Calculi for HOL

- Note that $\mathfrak{NK}_\beta$ and $\mathfrak{NK}_{\beta\mathfrak{fb}}$ correspond to the extremes of the model classes in our landscape of model classes. For example, $\mathfrak{NK}_{\beta\mathfrak{fb}}$ will be proven sound and complete for Henkin models, and $\mathfrak{NK}_\beta$ will be proven sound and complete for $\mathfrak{M}_\beta(\Sigma)$.
- Standard models do not admit (recursively axiomatizable) calculi that are sound and complete.
- In the following we will develop the abstract consistency proof method for HOL (wrt all the different semantic classes $\mathfrak{M}_*(\Sigma)$ in our landscape) and we will analyse soundness and completeness of each $\mathfrak{NK}_*$ with respect to each corresponding model class $\mathfrak{M}_*(\Sigma)$ with the help of the abstract consistency method.

ND Calculi for HOL

- (Soundness for $\mathfrak{NK}_*$)
  $\mathfrak{NK}_*$ is sound for $\mathfrak{M}_*(\Sigma)$ for $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$.
  That is, if $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{C}$ is derivable, then $\mathcal{M} \models \mathbf{C}$ for all models $\mathcal{M} = (\mathcal{D}, @, \mathcal{E}, \upsilon)$ in $\mathfrak{M}_*(\Sigma)$ such that $\mathcal{M} \models \Phi$.

  Proof: ... exercise ...

ND Calculi for HOL

- (Soundness for $\mathfrak{NK}_*$)
  $\mathfrak{NK}_*$ is sound for $\mathfrak{M}_*(\Sigma)$ for $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$.
  That is, if $\mathbf{\Phi} \Vdash_{\mathfrak{NK}_*} \mathbf{C}$ is derivable, then $\mathcal{M} \models \mathbf{C}$ for all models $\mathcal{M} = (\mathcal{D}, @, \mathcal{E}, \upsilon)$ in $\mathfrak{M}_*(\Sigma)$ such that $\mathcal{M} \models \mathbf{\Phi}$.

  Proof: . . . exercise . . .

- (Completeness for $\mathfrak{NK}_*$)
  Let $\mathbf{\Phi}$ be a sufficiently Σ-pure set of sentences, $\mathbf{A}$ be a sentence, and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. If $\mathbf{A}$ is valid in all models $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ that satisfy $\mathbf{\Phi}$, then $\mathbf{\Phi} \Vdash_{\mathfrak{NK}_*} \mathbf{A}$.

  Proof: . . . will follow . . .

ND Calculi for HOL

- (Soundness for $\mathfrak{NK}_*$)
  $\mathfrak{NK}_*$ is sound for $\mathfrak{M}_*(\Sigma)$ for $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$.
  That is, if $\mathbf{\Phi} \Vdash_{\mathfrak{NK}_*} \mathbf{C}$ is derivable, then $\mathcal{M} \models \mathbf{C}$ for all models $\mathcal{M} = (\mathcal{D}, @, \mathcal{E}, \upsilon)$ in $\mathfrak{M}_*(\Sigma)$ such that $\mathcal{M} \models \mathbf{\Phi}$.

  Proof: . . . exercise . . .

- (Completeness for $\mathfrak{NK}_*$)
  Let $\mathbf{\Phi}$ be a sufficiently Σ-pure set of sentences, $\mathbf{A}$ be a sentence, and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. If $\mathbf{A}$ is valid in all models $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ that satisfy $\mathbf{\Phi}$, then $\mathbf{\Phi} \Vdash_{\mathfrak{NK}_*} \mathbf{A}$.

  Proof: . . . will follow . . .

ND Calculi for HOL

Derivation of $\neg(\mathsf{p} \vee \neg\mathsf{p}) \Vdash (\mathsf{p} \vee \neg\mathsf{p})$

$$
\begin{prooftree}
\infer0[$\mathfrak{NK}(Hyp)$]{\neg(\mathsf{p} \vee \neg\mathsf{p}), \mathsf{p} \Vdash \neg(\mathsf{p} \vee \neg\mathsf{p})}
\infer0[$\mathfrak{NK}(Hyp)$]{\neg(\mathsf{p} \vee \neg\mathsf{p}), \mathsf{p} \Vdash \mathsf{p}}
\infer1[$\mathfrak{NK}(\vee I_L)$]{\neg(\mathsf{p} \vee \neg\mathsf{p}), \mathsf{p} \Vdash (\mathsf{p} \vee \neg\mathsf{p})}
\infer2[$\mathfrak{NK}(\neg E)$]{\neg(\mathsf{p} \vee \neg\mathsf{p}), \mathsf{p} \Vdash \mathbf{F}_o}
\infer1[$\mathfrak{NK}(\neg I)$]{\neg(\mathsf{p} \vee \neg\mathsf{p}) \Vdash \neg\mathsf{p}}
\infer1[$\mathfrak{NK}(\vee I_R)$]{\neg(\mathsf{p} \vee \neg\mathsf{p}) \Vdash (\mathsf{p} \vee \neg\mathsf{p})}
\end{prooftree}
$$

Completeness (of $\mathfrak{NK}_*$)

Completeness (of $\mathfrak{NK}_*$)

- How to prove completeness?

Completeness (of $\mathfrak{NK}_*$)

- How to prove completeness?
- Completeness can be proven rather easily for propositional logic calculi.

Completeness (of $\mathfrak{NK}_*$)

- How to prove completeness?
- Completeness can be proven rather easily for propositional logic calculi.
- For first-order and especially higher-order logic completeness proofs become increasingly difficult and technical.

Completeness (of $\mathfrak{N}\mathfrak{K}_*$)

- How to prove completeness?
- Completeness can be proven rather easily for propositional logic calculi.
- For first-order and especially higher-order logic completeness proofs become increasingly difficult and technical.
- Here we will introduce a strong proof tool that uniformly supports completeness proofs (and many other things): abstract consistency.

Completeness (of $\mathfrak{NK}_*$)

- How to prove completeness?
- Completeness can be proven rather easily for propositional logic calculi.
- For first-order and especially higher-order logic completeness proofs become increasingly difficult and technical.
- Here we will introduce a strong proof tool that uniformly supports completeness proofs (and many other things): abstract consistency.
- This proof tool is based on a strong theorem which connects syntax and semantics: model existence theorem.

Abstract Consistency

Abstract Consistency: History

- Technique was developed for first-order logic by Jaakko Hintikka and Raymond Smullyan [Hintikka55,Smullyan63,Smullyan68]. It is well explained in Fitting's textbook [Fitting96].

Abstract Consistency: History

- Technique was developed for first-order logic by Jaakko Hintikka and Raymond Smullyan [Hintikka55,Smullyan63,Smullyan68]. It is well explained in Fitting's textbook [Fitting96].
- The technidue has been (partly) extended to higher-order logic by Peter Andrews' in [Andrews71]; Peter Andrews only achieves a generalization for his rather weak semantical v-complexes (corresponding to our $\mathfrak{M}_\beta(\Sigma)$) and not, for instance, for Henkin Semantics. This extension is well explained in Peter Andrews's textbook [Andrews02].

Abstract Consistency: History

- Technique was developed for first-order logic by Jaakko Hintikka and Raymond Smullyan [Hintikka55,Smullyan63,Smullyan68]. It is well explained in Fitting's textbook [Fitting96].
- The technidue has been (partly) extended to higher-order logic by Peter Andrews' in [Andrews71]; Peter Andrews only achieves a generalization for his rather weak semantical v-complexes (corresponding to our $\mathfrak{M}_\beta(\Sigma)$) and not, for instance, for Henkin Semantics. This extension is well explained in Peter Andrews's textbook [Andrews02].
- The technique has been extended to our landscape of HOL model classes in [Benzmueller-PhD-99,JSL04].

Abstract Consistency: Idea

- A model existence theorem for a logical system (i.e., a logical language L together with a consequence relation $\models$) is a theorem of the form:

Abstract Consistency: Idea

- A model existence theorem for a logical system (i.e., a logical language L together with a consequence relation $\models$) is a theorem of the form:

  If a set of sentences Φ of L is a member of an (saturated) abstract consistency class Γ, then there exists a model for Φ.

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus C (i.e., the derivability rel. $\vdash_{\mathsf{C}}$) by

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus $\mathcal{C}$ (i.e., the derivability rel. $\vdash_{\mathcal{C}}$) by

  proving that the class Γ of sets of sentences Φ that are $\mathcal{C}$-consistent (i.e., cannot be refuted in $\mathcal{C}$) is an (saturated) abstract consistency class.

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus $\mathcal{C}$ (i.e., the derivability rel. $\vdash_{\mathcal{C}}$) by

  proving that the class Γ of sets of sentences Φ that are $\mathcal{C}$-consistent (i.e., cannot be refuted in $\mathcal{C}$) is an (saturated) abstract consistency class.

- Why does this work?

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus $\mathcal{C}$ (i.e., the derivability rel. $\vdash_{\mathcal{C}}$) by

  proving that the class Γ of sets of sentences Φ that are $\mathcal{C}$-consistent (i.e., cannot be refuted in $\mathcal{C}$) is an (saturated) abstract consistency class.

- Why does this work?
  - The model existence theorem tells us that $\mathcal{C}$-consistent sets of sentences are satisfiable.

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus $\mathcal{C}$ (i.e., the derivability rel. $\vdash_{\mathcal{C}}$) by

  proving that the class Γ of sets of sentences Φ that are $\mathcal{C}$-consistent (i.e., cannot be refuted in $\mathcal{C}$) is an (saturated) abstract consistency class.

- Why does this work?
  - The model existence theorem tells us that $\mathcal{C}$-consistent sets of sentences are satisfiable.
  - Now we assume that a sentence A is valid, so $\neg A$ does not have a model and is therefore $\mathcal{C}$-inconsistent.

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus $\mathcal{C}$ (i.e., the derivability rel. $\vdash_{\mathcal{C}}$) by

  proving that the class Γ of sets of sentences Φ that are $\mathcal{C}$-consistent (i.e., cannot be refuted in $\mathcal{C}$) is an (saturated) abstract consistency class.

- Why does this work?
  - The model existence theorem tells us that $\mathcal{C}$-consistent sets of sentences are satisfiable.
  - Now we assume that a sentence A is valid, so $\neg A$ does not have a model and is therefore $\mathcal{C}$-inconsistent.
  - Hence, $\neg A$ is refutable in $\mathcal{C}$.

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus $\mathcal{C}$ (i.e., the derivability rel. $\vdash_{\mathcal{C}}$) by

  proving that the class Γ of sets of sentences Φ that are $\mathcal{C}$-consistent (i.e., cannot be refuted in $\mathcal{C}$) is an (saturated) abstract consistency class.

- Why does this work?
  - The model existence theorem tells us that $\mathcal{C}$-consistent sets of sentences are satisfiable.
  - Now we assume that a sentence $\mathbf{A}$ is valid, so $\neg\mathbf{A}$ does not have a model and is therefore $\mathcal{C}$-inconsistent.
  - Hence, $\neg\mathbf{A}$ is refutable in $\mathcal{C}$.
  - This shows refutation completeness of $\mathcal{C}$.

Abstract Consistency: Idea

- Employing the model existence theorem we can prove completeness of a calculus C (i.e., the derivability rel. $\vdash_C$) by

  proving that the class Γ of sets of sentences Φ that are C-consistent (i.e., cannot be refuted in C) is an (saturated) abstract consistency class.

- Why does this work?
  - The model existence theorem tells us that C-consistent sets of sentences are satisfiable.
  - Now we assume that a sentence A is valid, so $\neg A$ does not have a model and is therefore C-inconsistent.
  - Hence, $\neg A$ is refutable in C.
  - This shows refutation completeness of C.
  - For many calculi C, this also shows A is provable, thus establishing completeness of C.

Def.: Closed under Subsets / Compact

Let C be a class of sets then C is called closed under subset if for all sets S and T it holds that

Def.: Closed under Subsets / Compact

Let C be a class of sets then C is called closed under subset if for all sets S and T it holds that

from $S \subseteq T$ and $T \in C$ it follows that $S \in C$.

Def.: Closed under Subsets / Compact

Let C be a class of sets then C is called closed under subset if for all sets S and T it holds that

> from $S \subseteq T$ and $T \in C$ it follows that $S \in C$.

Let C be a class of sets. C is called compact or of finite character if and only if for every set S holds:

Def.: Closed under Subsets / Compact

Let C be a class of sets then C is called closed under subset if for all sets S and T it holds that

> from $S \subseteq T$ and $T \in C$ it follows that $S \in C$.

Let C be a class of sets. C is called compact or of finite character if and only if for every set S holds:

> $S \in C$ if and only if every finite subset of S is a member of C.

Ex.: Closed under Subsets / Compact

- not closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg A\}\}$

Ex.: Closed under Subsets / Compact

- not closed under subsets: $\{\{\neg(A \lor B), \neg A, C\}, \{\neg A\}\}$
- closed under subsets: $\{\{\neg(A \lor B), \neg A, C\}, \{\neg(A \lor B), \neg A\}, \{\neg(A \lor B), C\}, \{\neg A, C\}, \{\neg(A \lor B)\}, \{\neg A\}, \{C\}, \{\}\}$

Ex.: Closed under Subsets / Compact

- not closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg A\}\}$
- closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), C\}, \{\neg A, C\}, \{\neg(A \vee B)\}, \{\neg A\}, \{C\}, \{\}\}$
- We define two classes of sets

Ex.: Closed under Subsets / Compact

- not closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg A\}\}$
- closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), C\}, \{\neg A, C\}, \{\neg(A \vee B)\}, \{\neg A\}, \{C\}, \{\}\}$
- We define two classes of sets
  - $C := \{\varphi \mid \varphi$ is finite subset of $\mathbb{N}\}$

Ex.: Closed under Subsets / Compact

- not closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg A\}\}$
- closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), C\}, \{\neg A, C\}, \{\neg(A \vee B)\}, \{\neg A\}, \{C\}, \{\}\}$
- We define two classes of sets
  - $C := \{\varphi \mid \varphi$ is finite subset of $\mathbb{N}\}$
  - $D := 2^{\mathbb{N}}$

Ex.: Closed under Subsets / Compact

- not closed under subsets: $\{\{\neg(A \lor B), \neg A, C\}, \{\neg A\}\}$
- closed under subsets: $\{\{\neg(A \lor B), \neg A, C\}, \{\neg(A \lor B), \neg A\}, \{\neg(A \lor B), C\}, \{\neg A, C\}, \{\neg(A \lor B)\}, \{\neg A\}, \{C\}, \{\}\}$
- We define two classes of sets
  - $C := \{\varphi \mid \varphi$ is finite subset of $\mathbb{N}\}$
  - $D := 2^{\mathbb{N}}$
  - C is closed under subsets but not compact.

Ex.: Closed under Subsets / Compact

- not closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg A\}\}$
- closed under subsets: $\{\{\neg(A \vee B), \neg A, C\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), C\}, \{\neg A, C\}, \{\neg(A \vee B)\}, \{\neg A\}, \{C\}, \{\}\}$
- We define two classes of sets
  - $C := \{\varphi \mid \varphi$ is finite subset of $\mathbb{N}\}$
  - $D := 2^{\mathbb{N}}$
  - C is closed under subsets but not compact.
  - D is closed under subsets and compact.

Lemma: Closed under Subsets / Compact

Lemma:

If C is compact then C is closed under subsets.

Lemma: Closed under Subsets / Compact

Lemma:

If C is compact then C is closed under subsets.

Proof:

Lemma: Closed under Subsets / Compact

Lemma:

If C is compact then C is closed under subsets.

Proof:

Let $T \in C$ and $S \subseteq T$.

Lemma: Closed under Subsets / Compact

Lemma:

If C is compact then C is closed under subsets.

Proof:

Let $T \in C$ and $S \subseteq T$.
We have to show that $S \in C$.

Lemma: Closed under Subsets / Compact

Lemma:

If C is compact then C is closed under subsets.

Proof:

Let $T \in C$ and $S \subseteq T$.
We have to show that $S \in C$.
Every finite subset A of S is also a finite subset of T.

Lemma: Closed under Subsets / Compact

Lemma:

If C is compact then C is closed under subsets.

Proof:

Let $T \in C$ and $S \subseteq T$.
We have to show that $S \in C$.
Every finite subset A of S is also a finite subset of T.
Since C is compact and $T \in C$ we get that all $A \in C$.

Lemma: Closed under Subsets / Compact

Lemma:

If C is compact then C is closed under subsets.

Proof:

Let $T \in C$ and $S \subseteq T$.
We have to show that $S \in C$.
Every finite subset A of S is also a finite subset of T.
Since C is compact and $T \in C$ we get that all $A \in C$.
Thus, $S \in C$ by compactness.

Def.: Sufficiently Σ-Pure

We introduce a technical side-condition that ensures that we always have enough witness constants.

Def.: Sufficiently Σ-Pure

We introduce a technical side-condition that ensures that we always have enough witness constants.

> Let Σ be a signature and Φ be a set of Σ-sentences. Φ is called sufficiently Σ-pure if for each type α there is a set $\mathcal{P}_\alpha \subseteq \Sigma_\alpha$ of parameters with equal cardinality to $\mathit{wff}_\alpha(\Sigma)$, such that the elements of $\mathcal{P}_\alpha$ do not occur in the sentences of Φ.

Def.: Sufficiently Σ-Pure

We introduce a technical side-condition that ensures that we always have enough witness constants.

> Let Σ be a signature and Φ be a set of Σ-sentences. Φ is called sufficiently Σ-pure if for each type α there is a set $\mathcal{P}_\alpha \subseteq \Sigma_\alpha$ of parameters with equal cardinality to $\mathit{wff}_\alpha(\Sigma)$, such that the elements of $\mathcal{P}_\alpha$ do not occur in the sentences of Φ.

This can be obtained in practice by enriching the signature with spurious parameters.

Abstract Consistency: Conventions

Remember the conventions for this part of the lecture:

- signature Σ contains only the logical constants $\neg, \vee, \Pi^\alpha$ unless stated otherwise

Abstract Consistency: Conventions

Remember the conventions for this part of the lecture:

- signature Σ contains only the logical constants $\neg, \vee, \Pi^\alpha$ unless stated otherwise
- as a matter of convenience we will write $\varphi * \mathrm{A}$ for $\varphi \cup \{\mathrm{A}\}$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in cwff_o(\Sigma)$, $\mathbf{F} \in cwff_{\alpha\rightarrow o}(\Sigma)$ are arbitrary):

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha\to o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha\to o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha \to o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} =_\beta \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha \to o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} =_\beta \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\vee$ If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha \to o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} =_\beta \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\vee$ If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\wedge$ If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha\to o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} =_\beta \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\vee$ If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\wedge$ If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\forall$ If $\Pi^\alpha \mathbf{F} \in \Phi$, then $\Phi * \mathbf{F}\mathbf{W} \in \Gamma_\Sigma$ for each $\mathbf{W} \in \mathit{cwff}_\alpha(\Sigma)$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha\to o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} {=_\beta} \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\vee$ If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\wedge$ If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\forall$ If $\Pi^\alpha \mathbf{F} \in \Phi$, then $\Phi * \mathbf{F}\mathbf{W} \in \Gamma_\Sigma$ for each $\mathbf{W} \in \mathit{cwff}_\alpha(\Sigma)$.

$\nabla_\exists$ If $\neg\Pi^\alpha \mathbf{F} \in \Phi$, then $\Phi * \neg(\mathbf{F}w) \in \Gamma_\Sigma$ for any parameter $w_\alpha \in \Sigma_\alpha$ which does not occur in any sentence of Φ.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{F} \in \mathit{cwff}_{\alpha\rightarrow o}(\Sigma)$ are arbitrary):

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} =_\beta \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\vee$ If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\wedge$ If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\forall$ If $\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * \mathbf{F}\mathbf{W} \in \Gamma_\Sigma$ for each $\mathbf{W} \in \mathit{cwff}_\alpha(\Sigma)$.

$\nabla_\exists$ If $\neg\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * \neg(\mathbf{F}\mathsf{w}) \in \Gamma_\Sigma$ for any parameter $\mathsf{w}_\alpha \in \Sigma_\alpha$ which does not occur in any sentence of Φ.

(These properties are going back to Hintikka, Smullyan, and Andrews)

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in cwff_o(\Sigma)$, $\mathbf{G}, \mathbf{H}, (\lambda X_\alpha . \mathbf{M}), (\lambda X_\alpha . \mathbf{N}) \in cwff_{\alpha \to \beta}(\Sigma)$ are arbitrary):

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{G}, \mathbf{H}, (\lambda X_\alpha \mathbf{.} \mathbf{M}), (\lambda X_\alpha \mathbf{.} \mathbf{N}) \in \mathit{cwff}_{\alpha \to \beta}(\Sigma)$ are arbitrary):

$\nabla_\mathfrak{b}$ If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi * \mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} * \mathbf{B} \in \Gamma_\Sigma$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in cwff_o(\Sigma)$, $\mathbf{G}, \mathbf{H}, (\lambda X_\alpha. \mathbf{M}), (\lambda X_\alpha. \mathbf{N}) \in cwff_{\alpha \to \beta}(\Sigma)$ are arbitrary):

$\nabla_\mathfrak{b}$ If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi * \mathbf{A} * \neg \mathbf{B} \in \Gamma_\Sigma$ or $\Phi * \neg \mathbf{A} * \mathbf{B} \in \Gamma_\Sigma$.

∇_η If $\mathbf{A} \overset{\beta\eta}{=} \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{G}, \mathbf{H}, (\lambda X_\alpha \mathbf{.} \mathbf{M}), (\lambda X_\alpha \mathbf{.} \mathbf{N}) \in \mathit{cwff}_{\alpha\to\beta}(\Sigma)$ are arbitrary):

$\nabla_\mathfrak{b}$ If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi * \mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} * \mathbf{B} \in \Gamma_\Sigma$.

∇_η If $\mathbf{A} \overset{\beta\eta}{=} \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

∇_ξ If $\neg(\lambda X_\alpha \mathbf{.} \mathbf{M} \doteq^{\alpha\to\beta} \lambda X_\alpha \mathbf{.} \mathbf{N}) \in \Phi$, then $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N}) \in \Gamma_\Sigma$ for any parameter $w_\alpha \in \Sigma_\alpha$ which does not occur in any sentence of Φ.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$, $\mathbf{G}, \mathbf{H}, (\lambda X_\alpha . \mathbf{M}), (\lambda X_\alpha . \mathbf{N}) \in \mathit{cwff}_{\alpha\to\beta}(\Sigma)$ are arbitrary):

$\nabla_\mathfrak{b}$ If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi * \mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} * \mathbf{B} \in \Gamma_\Sigma$.

∇_η If $\mathbf{A} \stackrel{\beta\eta}{=} \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

∇_ξ If $\neg(\lambda X_\alpha . \mathbf{M} \doteq^{\alpha\to\beta} \lambda X_\alpha . \mathbf{N}) \in \Phi$, then $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N}) \in \Gamma_\Sigma$ for any parameter $w_\alpha \in \Sigma_\alpha$ which does not occur in any sentence of Φ.

$\nabla_\mathfrak{f}$ If $\neg(\mathbf{G} \doteq^{\alpha\to\beta} \mathbf{H}) \in \Phi$, then $\Phi * \neg(\mathbf{G}w \doteq^\beta \mathbf{H}w) \in \Gamma_\Sigma$ for any parameter $w_\alpha \in \Sigma_\alpha$ which does not occur in any sentence of Φ.

Def.: Abstract Consistency Properties

Let Γ_Σ be a class of sets of Σ-sentences. We define (where $\Phi \in \Gamma_\Sigma$, $\alpha, \beta \in \mathcal{T}$, $\mathbf{A}, \mathbf{B} \in cwff_o(\Sigma)$, $\mathbf{G}, \mathbf{H}, (\lambda X_\alpha \mathbf{.M}), (\lambda X_\alpha \mathbf{.N}) \in cwff_{\alpha \to \beta}(\Sigma)$ are arbitrary):

$\nabla_\mathfrak{b}$ If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi * \mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} * \mathbf{B} \in \Gamma_\Sigma$.

∇_η If $\mathbf{A} \stackrel{\beta\eta}{=} \mathbf{B}$ and $\mathbf{A} \in \Phi$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

∇_ξ If $\neg(\lambda X_\alpha \mathbf{.M} \doteq^{\alpha \to \beta} \lambda X_\alpha \mathbf{.N}) \in \Phi$, then $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N}) \in \Gamma_\Sigma$ for any parameter $w_\alpha \in \Sigma_\alpha$ which does not occur in any sentence of Φ.

$\nabla_\mathfrak{f}$ If $\neg(\mathbf{G} \doteq^{\alpha \to \beta} \mathbf{H}) \in \Phi$, then $\Phi * \neg(\mathbf{G}w \doteq^\beta \mathbf{H}w) \in \Gamma_\Sigma$ for any parameter $w_\alpha \in \Sigma_\alpha$ which does not occur in any sentence of Φ.

(These properties are new in [Benzmueller-PhD-99,JSL04])

Def.: Abstract Consistency Classes

Let Σ be a signature and Γ_Σ be a class of sets of Σ-sentences that is closed under subsets.

Def.: Abstract Consistency Classes

Let Σ be a signature and Γ_Σ be a class of sets of Σ-sentences that is closed under subsets.

> If $\nabla_c, \nabla_\neg, \nabla_\beta, \nabla_\vee, \nabla_\wedge, \nabla_\forall$ and $\nabla_\exists$ are valid for Γ_Σ, then Γ_Σ is called an abstract consistency class for Σ-models.

Def.: Abstract Consistency Classes

Let Σ be a signature and Γ_Σ be a class of sets of Σ-sentences that is closed under subsets.

> If $\nabla_c, \nabla_\neg, \nabla_\beta, \nabla_\vee, \nabla_\wedge, \nabla_\forall$ and $\nabla_\exists$ are valid for Γ_Σ, then Γ_Σ is called an abstract consistency class for Σ-models.
>
> We will denote the collection of abstract consistency classes by $\mathfrak{Acc}_\beta$.

Def.: Abstract Consistency Classes

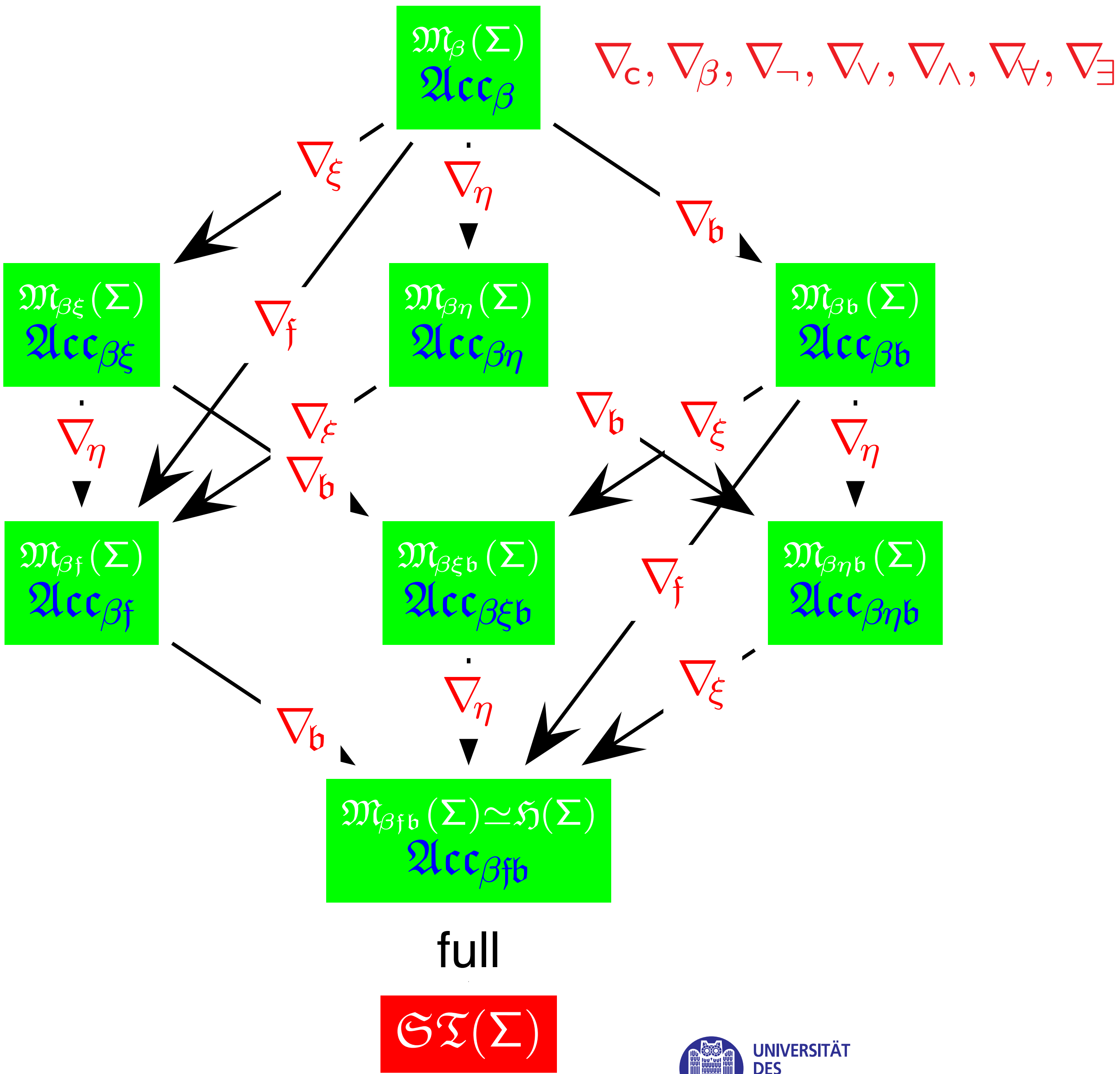
Let Σ be a signature and Γ_Σ be a class of sets of Σ-sentences that is closed under subsets.

If $\nabla_c, \nabla_\neg, \nabla_\beta, \nabla_\vee, \nabla_\wedge, \nabla_\forall$ and $\nabla_\exists$ are valid for Γ_Σ, then Γ_Σ is called an abstract consistency class for Σ-models.

We will denote the collection of abstract consistency classes by $\mathfrak{Acc}_\beta$.

Similarly, we introduce the following collections of specialized abstract consistency classes (with primitive equality): $\mathfrak{Acc}_{\beta\eta}, \mathfrak{Acc}_{\beta\xi}, \mathfrak{Acc}_{\beta\mathfrak{f}}, \mathfrak{Acc}_{\beta\mathfrak{b}}, \mathfrak{Acc}_{\beta\eta\mathfrak{b}}, \mathfrak{Acc}_{\beta\xi\mathfrak{b}}, \mathfrak{Acc}_{\beta\mathfrak{f}\mathfrak{b}}$, where we indicate by indices which additional properties from $\{\nabla_\eta, \nabla_\xi, \nabla_\mathfrak{f}, \nabla_\mathfrak{b}\}$ are required.

Abstract Consistency Classes



UNIVERSITÄT DES SAARLANDES

Ex.: Abstract Consistency Class

- not an abstract consistency class:
  $\{\{\neg(A \vee B), \neg A\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\}\}$

Ex.: Abstract Consistency Class

- not an abstract consistency class:
  $\{\{\neg(A \lor B), \neg A\}, \{\neg(A \lor B)\}, \{\neg A\}, \{\}\}$
- still not:
  $\{\{\neg(A \lor B), \neg A, \neg B\}, \{\neg(A \lor B), \neg A\}, \{\neg(A \lor B)\}, \{\neg A\}, \{\}\}$

Ex.: Abstract Consistency Class

- not an abstract consistency class:
  $\{\{\neg(A \vee B), \neg A\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\}\}$
- still not:
  $\{\{\neg(A \vee B), \neg A, \neg B\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\}\}$
- how about this one:
  $\Gamma := \{\{\neg(A \vee B), \neg A, \neg B\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), \neg B\}, \{\neg A, \neg B\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\neg B\}, \{\}\}$

Ex.: Abstract Consistency Class

- not an abstract consistency class:
  $\{\{\neg(A \vee B), \neg A\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\}\}$
- still not:
  $\{\{\neg(A \vee B), \neg A, \neg B\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\}\}$
- how about this one:
  $\Gamma := \{\{\neg(A \vee B), \neg A, \neg B\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), \neg B\}, \{\neg A, \neg B\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\neg B\}, \{\}\}$
- and how about this:
  $\Gamma_0 := \Gamma$
  $\Phi \in \Gamma_i \wedge A \in \Phi \wedge B =_{\beta\eta} A \wedge B \neq A \wedge (\Phi * B) \notin \Gamma_i \longrightarrow$
  $\Gamma_{i+1} := \text{close-under-subsets}(\Gamma_i * (\Phi * B))$
  $\Gamma^* := \Gamma_\infty$

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg$, $\vee$ and Π^{α}.

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg, \vee$ and Π^{α}. A means to generalize the framework over the concrete choice of logical primitives is provided by the uniform notation approach as, for instance, given in [Fitting96].

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg, \vee$ and Π^{α}. A means to generalize the framework over the concrete choice of logical primitives is provided by the uniform notation approach as, for instance, given in [Fitting96]. This can be done in straightforward manner: $\nabla_{\wedge}$ becomes an α-property, $\nabla_{\vee}$ becomes a β-property, $\nabla_{\forall}$ becomes a γ-property, and $\nabla_{\exists}$ becomes a δ-property.

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg, \vee$ and Π^{α}. A means to generalize the framework over the concrete choice of logical primitives is provided by the uniform notation approach as, for instance, given in [Fitting96]. This can be done in straightforward manner: $\nabla_{\wedge}$ becomes an α-property, $\nabla_{\vee}$ becomes a β-property, $\nabla_{\forall}$ becomes a γ-property, and $\nabla_{\exists}$ becomes a δ-property. Thus they will have the following form:

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg, \vee$ and Π^α. A means to generalize the framework over the concrete choice of logical primitives is provided by the uniform notation approach as, for instance, given in [Fitting96]. This can be done in straightforward manner: $\nabla_\wedge$ becomes an α-property, $\nabla_\vee$ becomes a β-property, $\nabla_\forall$ becomes a γ-property, and $\nabla_\exists$ becomes a δ-property. Thus they will have the following form:

α-case If $\alpha \in \Phi$, then $\Phi * \alpha_1 * \alpha_2 \in \Gamma_\Sigma$.

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg$, $\vee$ and Π^α. A means to generalize the framework over the concrete choice of logical primitives is provided by the uniform notation approach as, for instance, given in [Fitting96]. This can be done in straightforward manner: $\nabla_\wedge$ becomes an α-property, $\nabla_\vee$ becomes a β-property, $\nabla_\forall$ becomes a γ-property, and $\nabla_\exists$ becomes a δ-property. Thus they will have the following form:

α-case If $\alpha \in \Phi$, then $\Phi * \alpha_1 * \alpha_2 \in \Gamma_\Sigma$.

β-case If $\beta \in \Phi$, then $\Phi * \beta_1 \in \Gamma_\Sigma$ or $\Phi * \beta_2 \in \Gamma_\Sigma$.

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg, \vee$ and Π^α. A means to generalize the framework over the concrete choice of logical primitives is provided by the uniform notation approach as, for instance, given in [Fitting96]. This can be done in straightforward manner: $\nabla_\wedge$ becomes an α-property, $\nabla_\vee$ becomes a β-property, $\nabla_\forall$ becomes a γ-property, and $\nabla_\exists$ becomes a δ-property. Thus they will have the following form:

α-case If $\alpha \in \Phi$, then $\Phi * \alpha_1 * \alpha_2 \in \Gamma_\Sigma$.

β-case If $\beta \in \Phi$, then $\Phi * \beta_1 \in \Gamma_\Sigma$ or $\Phi * \beta_2 \in \Gamma_\Sigma$.

γ-case If $\gamma \in \Phi$, then $\Phi * \gamma\mathbf{W} \in \Gamma_\Sigma$ for each $\mathbf{W} \in \mathit{cwff}_\alpha(\Sigma)$.

Rem.: Possible Generalization

The work presented here is based on the choice of the primitive logical connectives $\neg$, $\vee$ and Π^{α}. A means to generalize the framework over the concrete choice of logical primitives is provided by the uniform notation approach as, for instance, given in [Fitting96]. This can be done in straightforward manner: $\nabla_{\wedge}$ becomes an α-property, $\nabla_{\vee}$ becomes a β-property, $\nabla_{\forall}$ becomes a γ-property, and $\nabla_{\exists}$ becomes a δ-property. Thus they will have the following form:

α-case If $\alpha \in \Phi$, then $\Phi * \alpha_1 * \alpha_2 \in \Gamma_{\Sigma}$.

β-case If $\beta \in \Phi$, then $\Phi * \beta_1 \in \Gamma_{\Sigma}$ or $\Phi * \beta_2 \in \Gamma_{\Sigma}$.

γ-case If $\gamma \in \Phi$, then $\Phi * \gamma\mathbf{W} \in \Gamma_{\Sigma}$ for each $\mathbf{W} \in \mathit{cwff}_{\alpha}(\Sigma)$.

δ-case If $\delta \in \Phi$, then $\Phi * \delta\mathrm{w} \in \Gamma_{\Sigma}$ for any parameter $\mathrm{w}_{\alpha} \in \Sigma$ which does not occur in any sentence of Φ.

Def.: Saturated

Consider the following property (where $\Phi \in \Gamma_\Sigma$, $\mathbf{A} \in cwff_o(\Sigma)$):

Def.: Saturated

Consider the following property (where $\Phi \in \Gamma_\Sigma$, $\mathbf{A} \in cwff_o(\Sigma)$):

∇_{sat} Either $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma_\Sigma$.

Def.: Saturated

Consider the following property (where $\Phi \in \Gamma_\Sigma$, $\mathbf{A} \in cwff_o(\Sigma)$):

∇_{sat} Either $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma_\Sigma$.

We call an abstract consistency class Γ_Σ atomically saturated if ∇_{sat} holds for all atomic A.

Def.: Saturated

Consider the following property (where $\Phi \in \Gamma_\Sigma$, $\mathbf{A} \in cwff_o(\Sigma)$):

∇_{sat} Either $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma_\Sigma$.

We call an abstract consistency class Γ_Σ atomically saturated if ∇_{sat} holds for all atomic A.

We call an abstract consistency class Γ_Σ saturated if ∇_{sat} holds for all A.

Ex.: Saturated

- consider Γ (and Γ^*) from before:
  $\{\{\neg(A \vee B), \neg A, \neg B\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), \neg B\}, \{\neg A, \neg B\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\neg B\}, \{\}\}$

Ex.: Saturated

- consider Γ (and Γ^*) from before: $\{\{\neg(A \vee B), \neg A, \neg B\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), \neg B\}, \{\neg A, \neg B\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\neg B\}, \{\}\}$
- Γ (and Γ^*) is atomically saturated in case our signature contains no further constants besides A_o and B_o and the logical connectives.

Ex.: Saturated

- consider Γ (and Γ^*) from before:
  $\{\{\neg(A \lor B), \neg A, \neg B\}, \{\neg(A \lor B), \neg A\}, \{\neg(A \lor B), \neg B\}, \{\neg A, \neg B\}, \{\neg(A \lor B)\}, \{\neg A\}, \{\neg B\}, \{\}\}$
- Γ (and Γ^*) is atomically saturated in case our signature contains no further constants besides A_o and B_o and the logical connectives.
- if there is another symbol C_o in the signature, then Γ (and Γ^*) is not atomically saturated anymore

Ex.: Saturated

- consider Γ (and Γ^*) from before:
  $\{\{\neg(A \vee B), \neg A, \neg B\}, \{\neg(A \vee B), \neg A\}, \{\neg(A \vee B), \neg B\}, \{\neg A, \neg B\}, \{\neg(A \vee B)\}, \{\neg A\}, \{\neg B\}, \{\}\}$
- Γ (and Γ^*) is atomically saturated in case our signature contains no further constants besides A_o and B_o and the logical connectives.
- if there is another symbol C_o in the signature, then Γ (and Γ^*) is not atomically saturated anymore
- Γ (and Γ^*) is not saturated: for instance, it does not provide information on the formulas $(\neg A \vee B) \vee A$ and $\Pi^o(\lambda X_o . X)$

Thm.: Model Existence Theorem

Let Γ_Σ be a saturated abstract consistency class and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.
For all $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$ we have:

Thm.: Model Existence Theorem

Let Γ_Σ be a saturated abstract consistency class and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.
For all $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$ we have:

> If Γ_Σ is an $\mathfrak{Acc}_*$, then there exists a model $\mathcal{M} \in \mathfrak{M}_*$ that satisfies Φ.

Thm.: Model Existence Theorem

Let Γ_Σ be a saturated abstract consistency class and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.
For all $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$ we have:

> If Γ_Σ is an $\mathfrak{Acc}_*$, then there exists a model $\mathcal{M} \in \mathfrak{M}_*$ that satisfies Φ.

Furthermore, each domain of $\mathcal{M}$ has cardinality at most $\aleph_s$.

Thm.: Model Existence Theorem

Let Γ_Σ be a saturated abstract consistency class and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.
For all $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$ we have:

> If Γ_Σ is an $\mathfrak{Acc}_*$, then there exists a model $\mathcal{M} \in \mathfrak{M}_*$ that satisfies Φ.

Furthermore, each domain of $\mathcal{M}$ has cardinality at most $\aleph_s$.

Proof: . . . we are not yet ready for this . . .

Thm.: Model Existence for Henkin Models

Let Γ_Σ be a saturated abstract consistency class in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.

Thm.: Model Existence for Henkin Models

Let Γ_Σ be a saturated abstract consistency class in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.

Then there is a Henkin Model that satisfies Φ.

Thm.: Model Existence for Henkin Models

Let Γ_Σ be a saturated abstract consistency class in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.

Then there is a Henkin Model that satisfies Φ.

Furthermore, each domain of the model has cardinality at most $\aleph_s$.

Thm.: Model Existence for Henkin Models

Let Γ_Σ be a saturated abstract consistency class in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ and let $\Phi \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences.

Then there is a Henkin Model that satisfies Φ.

Furthermore, each domain of the model has cardinality at most $\aleph_s$.

Proof: . . . we are not yet ready for this . . .

Completeness of $\mathfrak{NK}_*$ via Abstract Consistency

Def.: $\mathfrak{NK}_*$-Consistent/Inconsistent

A set of sentences Φ is $\mathfrak{NK}_*$-inconsistent if $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{F}_o$, and $\mathfrak{NK}_*$-consistent otherwise.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi$ is $\mathfrak{NK}_*$-consistent$\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma_\Sigma^* := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ_Σ^* is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$.
Hence Φ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \textit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$.
Hence Φ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$.
Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq cwff_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$. Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$. Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A}{=_\beta}\mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{B}$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$. Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{B}$. Since $\mathbf{A} \in \Phi$, we know $\Phi \Vdash \mathbf{B}$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\beta)$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$.
Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{B}$. Since $\mathbf{A} \in \Phi$, we know $\Phi \Vdash \mathbf{B}$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\beta)$. Using $\mathfrak{NK}(\neg E)$, we know $\Phi \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$.
Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{B}$. Since $\mathbf{A} \in \Phi$, we know $\Phi \Vdash \mathbf{B}$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\beta)$. Using $\mathfrak{NK}(\neg E)$, we know $\Phi \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma_\Sigma^* := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ_Σ^* is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$. Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{B}$. Since $\mathbf{A} \in \Phi$, we know $\Phi \Vdash \mathbf{B}$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\beta)$. Using $\mathfrak{NK}(\neg E)$, we know $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\neg$ Suppose $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A}$ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$. Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{B}$. Since $\mathbf{A} \in \Phi$, we know $\Phi \Vdash \mathbf{B}$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\beta)$. Using $\mathfrak{NK}(\neg E)$, we know $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\neg$ Suppose $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A}$ is $\mathfrak{NK}_*$-inconsistent. From $\Phi * \mathbf{A} \Vdash \mathbf{F}_o$ and $\mathfrak{NK}(\neg I)$, we have $\Phi \Vdash \neg\mathbf{A}$.

Lemma: Saturated $\mathfrak{Acc}_*$

The class $\Gamma^*_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \Phi \text{ is } \mathfrak{NK}_*\text{-consistent}\}$ is a saturated $\mathfrak{Acc}_*$.

Proof: Obviously Γ^*_Σ is closed under subsets, since any subset of an $\mathfrak{NK}_*$-consistent set is $\mathfrak{NK}_*$-consistent. We now check the remaining conditions. We prove all the properties by proving their contrapositive.

∇_c Suppose $\mathbf{A}, \neg\mathbf{A} \in \Phi$. We have $\Phi \Vdash \mathbf{F}_o$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$. Hence Φ is $\mathfrak{NK}_*$-inconsistent.

∇_β Let $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B}$ be $\mathfrak{NK}_*$-inconsistent. That is, $\Phi * \mathbf{B} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{B}$. Since $\mathbf{A} \in \Phi$, we know $\Phi \Vdash \mathbf{B}$ by $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\beta)$. Using $\mathfrak{NK}(\neg E)$, we know $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\neg$ Suppose $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A}$ is $\mathfrak{NK}_*$-inconsistent. From $\Phi * \mathbf{A} \Vdash \mathbf{F}_o$ and $\mathfrak{NK}(\neg I)$, we have $\Phi \Vdash \neg\mathbf{A}$. Since $\neg\neg\mathbf{A} \in \Phi$, we can apply $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\neg E)$ to obtain $\Phi \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_\circ$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$.

Lemma: Saturated $\mathfrak{Acc}_*$

- $\nabla_\vee$ Suppose $(\mathbf{A} \lor \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\lor E)$, we have $\Phi \Vdash \mathbf{F}_o$.

- $\nabla_\land$ Suppose $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\lor I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \lor \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\lor I_L)$, we have $\Phi \Vdash \mathbf{A} \lor \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_\circ$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_\circ$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

$\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

- $\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.
- $\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\mathcal{C}ontr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(\mathcal{C}ontr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.
- $\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\,\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\neg I)$, $\Phi \Vdash \neg(\mathbf{G}\,\mathbf{A})$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

$\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\,\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\neg I)$, $\Phi \Vdash \neg(\mathbf{G}\,\mathbf{A})$. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\Pi E)$, $\Phi \Vdash \mathbf{G}\,\mathbf{A}$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

$\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\neg I)$, $\Phi \Vdash \neg(\mathbf{G}\mathbf{A})$. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\Pi E)$, $\Phi \Vdash \mathbf{G}\mathbf{A}$. Finally, $\mathfrak{NK}(\neg E)$ implies $\Phi \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \lor \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\lor E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\lor I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \lor \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\lor I_L)$, we have $\Phi \Vdash \mathbf{A} \lor \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

$\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\neg I)$, $\Phi \Vdash \neg(\mathbf{G}\mathbf{A})$. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\Pi E)$, $\Phi \Vdash \mathbf{G}\mathbf{A}$. Finally, $\mathfrak{NK}(\neg E)$ implies $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\exists$ Suppose $\neg(\Pi^\alpha \mathbf{G}) \in \Phi$, w_α is a parameter which does not occur in Φ, and $\Phi * \neg(\mathbf{G}w)$ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

$\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\neg I)$, $\Phi \Vdash \neg(\mathbf{G}\mathbf{A})$. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\mathit{\Pi E})$, $\Phi \Vdash \mathbf{G}\mathbf{A}$. Finally, $\mathfrak{NK}(\neg E)$ implies $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\exists$ Suppose $\neg(\Pi^\alpha \mathbf{G}) \in \Phi$, w_α is a parameter which does not occur in Φ, and $\Phi * \neg(\mathbf{G}\mathsf{w})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$, $\Phi \Vdash \mathbf{G}\mathsf{w}$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

$\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\neg I)$, $\Phi \Vdash \neg(\mathbf{G}\mathbf{A})$. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\Pi E)$, $\Phi \Vdash \mathbf{G}\mathbf{A}$. Finally, $\mathfrak{NK}(\neg E)$ implies $\Phi \Vdash \mathbf{F}_o$.

$\nabla_\exists$ Suppose $\neg(\Pi^\alpha \mathbf{G}) \in \Phi$, w_α is a parameter which does not occur in Φ, and $\Phi * \neg(\mathbf{G}\mathsf{w})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$, $\Phi \Vdash \mathbf{G}\mathsf{w}$. By $\mathfrak{NK}(\Pi I)^{\mathsf{w}}$, $\Phi \Vdash (\Pi^\alpha \mathbf{G})$.

Lemma: Saturated $\mathfrak{Acc}_*$

- $\nabla_\vee$ Suppose $(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and both $\Phi * \mathbf{A}$ and $\Phi * \mathbf{B}$ are $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\vee E)$, we have $\Phi \Vdash \mathbf{F}_o$.
- $\nabla_\wedge$ Suppose $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B}$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_R)$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, we have $\Phi, \neg\mathbf{A} \Vdash \mathbf{F}_o$. By $\mathfrak{NK}(Contr)$ and $\mathfrak{NK}(\vee I_L)$, we have $\Phi \Vdash \mathbf{A} \vee \mathbf{B}$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.
- $\nabla_\forall$ Suppose $(\Pi^\alpha \mathbf{G}) \in \Phi$ and $\Phi * (\mathbf{G}\,\mathbf{A})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(\neg I)$, $\Phi \Vdash \neg(\mathbf{G}\,\mathbf{A})$. By $\mathfrak{NK}(Hyp)$ and $\mathfrak{NK}(\Pi E)$, $\Phi \Vdash \mathbf{G}\,\mathbf{A}$. Finally, $\mathfrak{NK}(\neg E)$ implies $\Phi \Vdash \mathbf{F}_o$.
- $\nabla_\exists$ Suppose $\neg(\Pi^\alpha \mathbf{G}) \in \Phi$, w_α is a parameter which does not occur in Φ, and $\Phi * \neg(\mathbf{G}\mathsf{w})$ is $\mathfrak{NK}_*$-inconsistent. By $\mathfrak{NK}(Contr)$, $\Phi \Vdash \mathbf{G}\mathsf{w}$. By $\mathfrak{NK}(\Pi I)^{\mathsf{w}}$, $\Phi \Vdash (\Pi^\alpha \mathbf{G})$. Using $\mathfrak{NK}(\neg E)$ with $\neg(\Pi^\alpha \mathbf{G}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.

Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.

Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

∇_ξ Suppose $*$ includes ξ, $\neg(\lambda X.\mathbf{M} \doteq^{\alpha\to\beta} \lambda X.\mathbf{N}) \in \Phi$, and $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$ is $\mathfrak{NK}_*$-inconsistent for some parameter w_α which does not occur in any sentence of Φ.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

∇_ξ Suppose $*$ includes ξ, $\neg(\lambda \mathsf{X}_\bullet \mathbf{M} \doteq^{\alpha\to\beta} \lambda \mathsf{X}_\bullet \mathbf{N}) \in \Phi$, and $\Phi * \neg([\mathsf{w}/\mathsf{X}]\mathbf{M} \doteq^\beta [\mathsf{w}/\mathsf{X}]\mathbf{N})$ is $\mathfrak{NK}_*$-inconsistent for some parameter w_α which does not occur in any sentence of Φ. By $\mathfrak{NK}(Contr)$, we have $\Phi \Vdash ([\mathsf{w}/\mathsf{X}]\mathbf{M} \doteq^\beta [\mathsf{w}/\mathsf{X}]\mathbf{N})$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

∇_ξ Suppose $*$ includes ξ, $\neg(\lambda X_\bullet \mathbf{M} \doteq^{\alpha\to\beta} \lambda X_\bullet \mathbf{N}) \in \Phi$, and $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$ is $\mathfrak{NK}_*$-inconsistent for some parameter w_α which does not occur in any sentence of Φ. By $\mathfrak{NK}(Contr)$, we have $\Phi \Vdash ([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$. By $\mathfrak{NK}(\beta)$, we have $\Phi \Vdash ((\lambda X_\bullet \mathbf{M} \doteq^\beta \mathbf{N})w)$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

∇_ξ Suppose $*$ includes ξ, $\neg(\lambda X.\mathbf{M} \doteq^{\alpha\to\beta} \lambda X.\mathbf{N}) \in \Phi$, and $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$ is $\mathfrak{NK}_*$-inconsistent for some parameter w_α which does not occur in any sentence of Φ. By $\mathfrak{NK}(Contr)$, we have $\Phi \Vdash ([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$. By $\mathfrak{NK}(\beta)$, we have $\Phi \Vdash ((\lambda X.\mathbf{M} \doteq^\beta \mathbf{N})w)$. By $\mathfrak{NK}(\Pi I)$, $\Phi \Vdash (\forall X.\mathbf{M} \doteq^\beta \mathbf{N})$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

∇_ξ Suppose $*$ includes ξ, $\neg(\lambda X.\mathbf{M} \doteq^{\alpha\to\beta} \lambda X.\mathbf{N}) \in \Phi$, and $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$ is $\mathfrak{NK}_*$-inconsistent for some parameter w_α which does not occur in any sentence of Φ. By $\mathfrak{NK}(Contr)$, we have $\Phi \Vdash ([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$. By $\mathfrak{NK}(\beta)$, we have $\Phi \Vdash ((\lambda X.\mathbf{M} \doteq^\beta \mathbf{N})w)$. By $\mathfrak{NK}(\Pi I)$, $\Phi \Vdash (\forall X.\mathbf{M} \doteq^\beta \mathbf{N})$. By $\mathfrak{NK}(\xi)$, $\Phi \Vdash (\lambda X.\mathbf{M} \doteq^{\alpha\to\beta} \lambda X.\mathbf{N})$.

Lemma: Saturated $\mathfrak{Acc}_*$

∇_{sat} Let $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathfrak{NK}_*$-inconsistent. We show that Φ is $\mathfrak{NK}_*$-inconsistent. Using $\mathfrak{NK}(\neg I)$, we know $\Phi \Vdash \neg\mathbf{A}$ and $\Phi \Vdash \neg\neg\mathbf{A}$. By $\mathfrak{NK}(\neg E)$, we have $\Phi \Vdash \mathbf{F}_o$.

Thus we have shown that Γ_Σ^β is saturated and in $\mathfrak{Acc}_\beta$.
Now let us check the conditions for the additional properties η, ξ, $\mathfrak{f}$, and $\mathfrak{b}$.

∇_η If $*$ includes η, then the proof proceeds as in ∇_β above, but with the rule $\mathfrak{NK}(\eta)$.

∇_ξ Suppose $*$ includes ξ, $\neg(\lambda X.\mathbf{M} \doteq^{\alpha\to\beta} \lambda X.\mathbf{N}) \in \Phi$, and $\Phi * \neg([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$ is $\mathfrak{NK}_*$-inconsistent for some parameter w_α which does not occur in any sentence of Φ. By $\mathfrak{NK}(Contr)$, we have $\Phi \Vdash ([w/X]\mathbf{M} \doteq^\beta [w/X]\mathbf{N})$. By $\mathfrak{NK}(\beta)$, we have $\Phi \Vdash ((\lambda X.\mathbf{M} \doteq^\beta \mathbf{N})w)$. By $\mathfrak{NK}(\Pi I)$, $\Phi \Vdash (\forall X.\mathbf{M} \doteq^\beta \mathbf{N})$. By $\mathfrak{NK}(\xi)$, $\Phi \Vdash (\lambda X.\mathbf{M} \doteq^{\alpha\to\beta} \lambda X.\mathbf{N})$. By $\mathfrak{NK}(\neg E)$, Φ is $\mathfrak{NK}_*$-inconsistent.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\mathfrak{f}$ This case is analogous to the previous one, generalizing $\lambda X_\bullet \mathbf{M} \doteq \lambda X_\bullet \mathbf{N}$ to arbitrary $\mathbf{G} \doteq \mathbf{H}$ and using the extensionality rule $\mathfrak{NK}(\mathfrak{f})$ instead of $\mathfrak{NK}(\xi)$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_{\mathfrak{f}}$ This case is analogous to the previous one, generalizing $\lambda X_{\bullet}\mathbf{M} \doteq \lambda X_{\bullet}\mathbf{N}$ to arbitrary $\mathbf{G} \doteq \mathbf{H}$ and using the extensionality rule $\mathfrak{NK}(\mathfrak{f})$ instead of $\mathfrak{NK}(\xi)$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_{\mathfrak{f}}$ This case is analogous to the previous one, generalizing $\lambda X_\bullet \mathbf{M} \doteq \lambda X_\bullet \mathbf{N}$ to arbitrary $\mathbf{G} \doteq \mathbf{H}$ and using the extensionality rule $\mathfrak{NK}(\mathfrak{f})$ instead of $\mathfrak{NK}(\xi)$.

$\nabla_{\mathfrak{b}}$ Suppose $*$ includes $\mathfrak{b}$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\mathfrak{f}$ This case is analogous to the previous one, generalizing $\lambda X_\bullet \mathbf{M} \doteq \lambda X_\bullet \mathbf{N}$ to arbitrary $\mathbf{G} \doteq \mathbf{H}$ and using the extensionality rule $\mathfrak{NK}(\mathfrak{f})$ instead of $\mathfrak{NK}(\xi)$.

$\nabla_\mathfrak{b}$ Suppose $*$ includes $\mathfrak{b}$. Assume that $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$ but both $\Phi * \neg\mathbf{A} * \mathbf{B} \notin \Gamma_\Sigma^*$ and $\Phi * \mathbf{A} * \neg\mathbf{B} \notin \Gamma_\Sigma^*$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_{\mathfrak{f}}$ This case is analogous to the previous one, generalizing $\lambda X_{\blacksquare}\mathbf{M} \doteq \lambda X_{\blacksquare}\mathbf{N}$ to arbitrary $\mathbf{G} \doteq \mathbf{H}$ and using the extensionality rule $\mathfrak{NK}(\mathfrak{f})$ instead of $\mathfrak{NK}(\xi)$.

$\nabla_{\mathfrak{b}}$ Suppose $*$ includes $\mathfrak{b}$. Assume that $\neg(\mathbf{A} \doteq^{o} \mathbf{B}) \in \Phi$ but both $\Phi * \neg\mathbf{A} * \mathbf{B} \notin \Gamma^{*}_{\Sigma}$ and $\Phi * \mathbf{A} * \neg\mathbf{B} \notin \Gamma^{*}_{\Sigma}$. So both are $\mathfrak{NK}_*$-inconsistent and we have $\Phi * \mathbf{A} \Vdash \mathbf{B}$ and $\Phi * \mathbf{B} \Vdash \mathbf{A}$ by $\mathfrak{NK}(Contr)$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_\mathfrak{f}$ This case is analogous to the previous one, generalizing $\lambda X.\mathbf{M} \doteq \lambda X.\mathbf{N}$ to arbitrary $\mathbf{G} \doteq \mathbf{H}$ and using the extensionality rule $\mathfrak{NK}(\mathfrak{f})$ instead of $\mathfrak{NK}(\xi)$.

$\nabla_\mathfrak{b}$ Suppose $*$ includes $\mathfrak{b}$. Assume that $\neg(\mathbf{A} \doteq^\circ \mathbf{B}) \in \Phi$ but both $\Phi * \neg\mathbf{A} * \mathbf{B} \notin \Gamma_\Sigma^*$ and $\Phi * \mathbf{A} * \neg\mathbf{B} \notin \Gamma_\Sigma^*$. So both are $\mathfrak{NK}_*$-inconsistent and we have $\Phi * \mathbf{A} \Vdash \mathbf{B}$ and $\Phi * \mathbf{B} \Vdash \mathbf{A}$ by $\mathfrak{NK}(Contr)$. By $\mathfrak{NK}(\mathfrak{b})$, we have $\Phi \Vdash (\mathbf{A} \doteq^\circ \mathbf{B})$.

Lemma: Saturated $\mathfrak{Acc}_*$

$\nabla_{\mathfrak{f}}$ This case is analogous to the previous one, generalizing $\lambda X\text{.}\mathbf{M} \doteq \lambda X\text{.}\mathbf{N}$ to arbitrary $\mathbf{G} \doteq \mathbf{H}$ and using the extensionality rule $\mathfrak{NK}(\mathfrak{f})$ instead of $\mathfrak{NK}(\xi)$.

$\nabla_{\mathfrak{b}}$ Suppose $*$ includes $\mathfrak{b}$. Assume that $\neg(\mathbf{A} \doteq^{o} \mathbf{B}) \in \Phi$ but both $\Phi * \neg\mathbf{A} * \mathbf{B} \notin \Gamma_{\Sigma}^{*}$ and $\Phi * \mathbf{A} * \neg\mathbf{B} \notin \Gamma_{\Sigma}^{*}$. So both are $\mathfrak{NK}_*$-inconsistent and we have $\Phi * \mathbf{A} \Vdash \mathbf{B}$ and $\Phi * \mathbf{B} \Vdash \mathbf{A}$ by $\mathfrak{NK}(Contr)$. By $\mathfrak{NK}(\mathfrak{b})$, we have $\Phi \Vdash (\mathbf{A} \doteq^{o} \mathbf{B})$. Since $\neg(\mathbf{A} \doteq^{o} \mathbf{B}) \in \Phi$, Φ is $\mathfrak{NK}_*$-inconsistent.

Thm.: Henkin's Theorem for $\mathfrak{NK}_*$

Let $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Every sufficiently Σ-pure $\mathfrak{NK}_*$-consistent set of sentences has an $\mathfrak{M}_*(\Sigma)$-model.

Proof:

Thm.: Henkin's Theorem for $\mathfrak{NK}_*$

Let $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Every sufficiently Σ-pure $\mathfrak{NK}_*$-consistent set of sentences has an $\mathfrak{M}_*(\Sigma)$-model.

Proof: Let Φ be a sufficiently Σ-pure $\mathfrak{NK}_*$-consistent set of sentences.

Thm.: Henkin's Theorem for $\mathfrak{NK}_*$

Let $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Every sufficiently Σ-pure $\mathfrak{NK}_*$-consistent set of sentences has an $\mathfrak{M}_*(\Sigma)$-model.

Proof: Let Φ be a sufficiently Σ-pure $\mathfrak{NK}_*$-consistent set of sentences. By the previous lemma we know that the class of sets of $\mathfrak{NK}_*$-consistent sentences constitute a saturated $\mathfrak{Acc}_*$,

Thm.: Henkin's Theorem for $\mathfrak{NK}_*$

Let $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Every sufficiently Σ-pure $\mathfrak{NK}_*$-consistent set of sentences has an $\mathfrak{M}_*(\Sigma)$-model.

Proof: Let Φ be a sufficiently Σ-pure $\mathfrak{NK}_*$-consistent set of sentences. By the previous lemma we know that the class of sets of $\mathfrak{NK}_*$-consistent sentences constitute a saturated $\mathfrak{Acc}_*$, thus the Model Existence Theorem guarantees an $\mathfrak{M}_*(\Sigma)$ model for Φ.

Thm.: Completeness Theorem for $\mathfrak{NK}_*$

Let Φ be a sufficiently Σ-pure set of sentences, $\mathbf{A}$ be a sentence, and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. If $\mathbf{A}$ is valid in all models $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ that satisfy Φ, then $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{A}$.

Proof:

Thm.: Completeness Theorem for $\mathfrak{NK}_*$

Let Φ be a sufficiently Σ-pure set of sentences, $\mathbf{A}$ be a sentence, and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. If $\mathbf{A}$ is valid in all models $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ that satisfy Φ, then $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{A}$.

Proof: Let $\mathbf{A}$ be given such that $\mathbf{A}$ is valid in all $\mathfrak{M}_*(\Sigma)$ models that satisfy Φ.

Thm.: Completeness Theorem for $\mathfrak{NK}_*$

Let Φ be a sufficiently Σ-pure set of sentences, $\mathbf{A}$ be a sentence, and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. If $\mathbf{A}$ is valid in all models $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ that satisfy Φ, then $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{A}$.

Proof: Let $\mathbf{A}$ be given such that $\mathbf{A}$ is valid in all $\mathfrak{M}_*(\Sigma)$ models that satisfy Φ. So, $\Phi * \neg\mathbf{A}$ is unsatisfiable in $\mathfrak{M}_*(\Sigma)$. Since only finitely many constants occur in $\neg\mathbf{A}$, $\Phi * \neg\mathbf{A}$ is sufficiently Σ-pure.

Thm.: Completeness Theorem for $\mathfrak{NK}_*$

Let Φ be a sufficiently Σ-pure set of sentences, $\mathbf{A}$ be a sentence, and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. If $\mathbf{A}$ is valid in all models $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ that satisfy Φ, then $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{A}$.

Proof: Let $\mathbf{A}$ be given such that $\mathbf{A}$ is valid in all $\mathfrak{M}_*(\Sigma)$ models that satisfy Φ. So, $\Phi * \neg\mathbf{A}$ is unsatisfiable in $\mathfrak{M}_*(\Sigma)$. Since only finitely many constants occur in $\neg\mathbf{A}$, $\Phi * \neg\mathbf{A}$ is sufficiently Σ-pure. So, $\Phi * \neg\mathbf{A}$ must be $\mathfrak{NK}_*$-inconsistent by Henkin's theorem above.

Thm.: Completeness Theorem for $\mathfrak{NK}_*$

Let Φ be a sufficiently Σ-pure set of sentences, $\mathbf{A}$ be a sentence, and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. If $\mathbf{A}$ is valid in all models $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ that satisfy Φ, then $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{A}$.

Proof: Let $\mathbf{A}$ be given such that $\mathbf{A}$ is valid in all $\mathfrak{M}_*(\Sigma)$ models that satisfy Φ. So, $\Phi * \neg\mathbf{A}$ is unsatisfiable in $\mathfrak{M}_*(\Sigma)$. Since only finitely many constants occur in $\neg\mathbf{A}$, $\Phi * \neg\mathbf{A}$ is sufficiently Σ-pure. So, $\Phi * \neg\mathbf{A}$ must be $\mathfrak{NK}_*$-inconsistent by Henkin's theorem above. Thus, $\Phi \Vdash_{\mathfrak{NK}_*} \mathbf{A}$ by $\mathfrak{NK}(Contr)$.

Compactness

We can use the completeness theorems obtained so far to prove a compactness theorem for our semantics:

Let Φ be a sufficiently Σ-pure set of sentences and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Φ has an $\mathfrak{M}_*(\Sigma)$-model iff every finite subset of Φ has an $\mathfrak{M}_*(\Sigma)$-model.

Proof:

Compactness

We can use the completeness theorems obtained so far to prove a compactness theorem for our semantics:

> Let Φ be a sufficiently Σ-pure set of sentences and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Φ has an $\mathfrak{M}_*(\Sigma)$-model iff every finite subset of Φ has an $\mathfrak{M}_*(\Sigma)$-model.

Proof: If Φ has no $\mathfrak{M}_*(\Sigma)$-model, then by the previous Henkin Theorem Φ is $\mathfrak{NK}_*$-inconsistent.

Compactness

We can use the completeness theorems obtained so far to prove a compactness theorem for our semantics:

> Let Φ be a sufficiently Σ-pure set of sentences and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Φ has an $\mathfrak{M}_*(\Sigma)$-model iff every finite subset of Φ has an $\mathfrak{M}_*(\Sigma)$-model.

Proof: If Φ has no $\mathfrak{M}_*(\Sigma)$-model, then by the previous Henkin Theorem Φ is $\mathfrak{NK}_*$-inconsistent. Since every $\mathfrak{NK}_*$-proof is finite, this means some finite subset Ψ of Φ is $\mathfrak{NK}_*$-inconsistent.

Compactness

We can use the completeness theorems obtained so far to prove a compactness theorem for our semantics:

> Let Φ be a sufficiently Σ-pure set of sentences and $* \in \{\beta, \beta\eta, \beta\xi, \beta\mathfrak{f}, \beta\mathfrak{b}, \beta\eta\mathfrak{b}, \beta\xi\mathfrak{b}, \beta\mathfrak{f}\mathfrak{b}\}$. Φ has an $\mathfrak{M}_*(\Sigma)$-model iff every finite subset of Φ has an $\mathfrak{M}_*(\Sigma)$-model.

Proof: If Φ has no $\mathfrak{M}_*(\Sigma)$-model, then by the previous Henkin Theorem Φ is $\mathfrak{N}\mathfrak{K}_*$-inconsistent. Since every $\mathfrak{N}\mathfrak{K}_*$-proof is finite, this means some finite subset Ψ of Φ is $\mathfrak{N}\mathfrak{K}_*$-inconsistent. Hence, Ψ has no $\mathfrak{M}_*(\Sigma)$-model.

Approaches to Higher-Order Resolution

Preliminaries and Notation

- only logical constants: $\neg_{o \to o}$, $\vee_{o \to o \to o}$, and $\Pi_{(\alpha \to o) \to o}$

Preliminaries and Notation

- only logical constants: $\neg_{o\to o}$, $\vee_{o\to o\to o}$, and $\Pi_{(\alpha\to o)\to o}$
- other logical operators can be defined (e.g., $\mathbf{A} \wedge \mathbf{B} := \neg(\neg\mathbf{A} \vee \neg\mathbf{B})$, $\forall X_\alpha.\mathbf{P}\ X := \Pi_{((\alpha\to o)\to o)}(\lambda X_\alpha.\mathbf{P}\ X)$, and $\exists X_\alpha.\mathbf{P}\ X := \neg\forall X_\alpha.\neg(\mathbf{P}\ X))$)

Preliminaries and Notation

- only logical constants: $\neg_{o\to o}$, $\vee_{o\to o\to o}$, and $\Pi_{(\alpha\to o)\to o}$
- other logical operators can be defined (e.g., $\mathbf{A} \wedge \mathbf{B} := \neg(\neg\mathbf{A} \vee \neg\mathbf{B})$, $\forall X_\alpha.\mathbf{P}\ X := \Pi_{((\alpha\to o)\to o)}(\lambda X_\alpha.\mathbf{P}\ X)$, and $\exists X_\alpha.\mathbf{P}\ X := \neg\forall X_\alpha.\neg(\mathbf{P}\ X))$)
- variables are printed as upper-case (e.g., X_α), constants as lower-case letters (e.g., c_α), and arbitrary terms appear as bold capital letters (e.g., $\mathbf{T}_\alpha$)

Preliminaries and Notation

- only logical constants: $\neg_{o\to o}$, $\vee_{o\to o\to o}$, and $\Pi_{(\alpha\to o)\to o}$
- other logical operators can be defined (e.g., $\mathbf{A} \wedge \mathbf{B} := \neg(\neg\mathbf{A} \vee \neg\mathbf{B})$, $\forall X_\alpha \mathbf{.} \mathbf{P}\ X := \Pi_{((\alpha\to o)\to o)}(\lambda X_\alpha \mathbf{.} \mathbf{P}\ X)$, and $\exists X_\alpha \mathbf{.} \mathbf{P}\ X := \neg\forall X_\alpha \mathbf{.} \neg(\mathbf{P}\ X))$)
- variables are printed as upper-case (e.g., X_α), constants as lower-case letters (e.g., c_α), and arbitrary terms appear as bold capital letters (e.g., $\mathbf{T}_\alpha$)
- we abbreviate function applications by $h_{\alpha_1\to\cdots\to\alpha_n\to\beta}\ \overline{\mathbf{U}^n_{\alpha_n}}$, which stands for $(\cdots(h_{\alpha_1\to\cdots\to\alpha_n\to\beta}\ \mathbf{U}^1_{\alpha_1})\cdots\mathbf{U}^n_{\alpha_n})$.

Preliminaries and Notation

- only logical constants: $\neg_{o\to o}$, $\vee_{o\to o\to o}$, and $\Pi_{(\alpha\to o)\to o}$
- other logical operators can be defined (e.g., $\mathbf{A} \wedge \mathbf{B} := \neg(\neg\mathbf{A} \vee \neg\mathbf{B})$, $\forall X_\alpha\text{.}\mathbf{P}\ X := \Pi_{((\alpha\to o)\to o)}(\lambda X_\alpha\text{.}\mathbf{P}\ X)$, and $\exists X_\alpha\text{.}\mathbf{P}\ X := \neg\forall X_\alpha\text{.}\neg(\mathbf{P}\ X))$)
- variables are printed as upper-case (e.g., X_α), constants as lower-case letters (e.g., c_α), and arbitrary terms appear as bold capital letters (e.g., $\mathbf{T}_\alpha$)
- we abbreviate function applications by $h_{\alpha_1\to\dots\to\alpha_n\to\beta}\ \overline{\mathbf{U}^n_{\alpha_n}}$, which stands for $(\cdots(h_{\alpha_1\to\dots\to\alpha_n\to\beta}\ \mathbf{U}^1_{\alpha_1})\cdots\mathbf{U}^n_{\alpha_n})$.
- α-, β-, η-, $\beta\eta$-conversion and the definition of β-normal, $\beta\eta$-normal, long $\beta\eta$-normal, and head-normal form defined as usual (see [Barendregt84])

Preliminaries and Notation

- only logical constants: $\neg_{o\to o}$, $\vee_{o\to o\to o}$, and $\Pi_{(\alpha\to o)\to o}$
- other logical operators can be defined (e.g., $\mathbf{A} \wedge \mathbf{B} := \neg(\neg\mathbf{A} \vee \neg\mathbf{B})$, $\forall X_\alpha . \mathbf{P}\ X := \Pi_{((\alpha\to o)\to o)}(\lambda X_\alpha . \mathbf{P}\ X)$, and $\exists X_\alpha . \mathbf{P}\ X := \neg\forall X_\alpha . \neg(\mathbf{P}\ X))$)
- variables are printed as upper-case (e.g., X_α), constants as lower-case letters (e.g., c_α), and arbitrary terms appear as bold capital letters (e.g., $\mathbf{T}_\alpha$)
- we abbreviate function applications by $h_{\alpha_1\to\dots\to\alpha_n\to\beta}\ \overline{\mathbf{U}^n_{\alpha_n}}$, which stands for $(\cdots(h_{\alpha_1\to\dots\to\alpha_n\to\beta}\ \mathbf{U}^1_{\alpha_1})\cdots\mathbf{U}^n_{\alpha_n})$.
- α-, β-, η-, $\beta\eta$-conversion and the definition of β-normal, $\beta\eta$-normal, long $\beta\eta$-normal, and head-normal form defined as usual (see [Barendregt84])
- substitutions defined as usual

Preliminaries and Notation

- substitutions are represented as $[\mathbf{T}_1/\mathsf{X}_1, \ldots, \mathbf{T}_\mathsf{n}/\mathsf{X}_\mathsf{n}]$ where the X_i specify the variables to be replaced by the terms $\mathbf{T}_\mathsf{i}$. The application of a substitution σ to a term (resp. literal or clause) $\mathbf{C}$ is printed $\mathbf{C}_\sigma$

Preliminaries and Notation

- substitutions are represented as $[\mathbf{T}_1/\mathsf{X}_1, \ldots, \mathbf{T}_n/\mathsf{X}_n]$ where the X_i specify the variables to be replaced by the terms $\mathbf{T}_i$. The application of a substitution σ to a term (resp. literal or clause) $\mathbf{C}$ is printed $\mathbf{C}_\sigma$
- a resolution calculus R provides a set of rules $\{r_n |\ 0 < n \leq i\}$ defined on clauses

Preliminaries and Notation

- substitutions are represented as $[\mathbf{T}_1/X_1, \ldots, \mathbf{T}_n/X_n]$ where the X_i specify the variables to be replaced by the terms $\mathbf{T}_i$. The application of a substitution σ to a term (resp. literal or clause) $\mathbf{C}$ is printed $\mathbf{C}_\sigma$
- a resolution calculus R provides a set of rules $\{r_n | \ 0 < n \leq i\}$ defined on clauses
- we write $\Phi \vdash^{r_n} \mathcal{C}$ ($\mathcal{C}' \vdash^{r_n} \mathcal{C}$) iff clause $\mathcal{C}$ is the result of a one step application of rule $r_n \in R$ to premise clauses $\mathcal{C}'_i \in \Phi$ (to $\mathcal{C}'$ respectively)

Preliminaries and Notation

- substitutions are represented as $[\mathbf{T}_1/\mathsf{X}_1, \ldots, \mathbf{T}_n/\mathsf{X}_n]$ where the X_i specify the variables to be replaced by the terms $\mathbf{T}_i$. The application of a substitution σ to a term (resp. literal or clause) $\mathbf{C}$ is printed $\mathbf{C}_\sigma$
- a resolution calculus R provides a set of rules $\{r_n \mid 0 < n \leq i\}$ defined on clauses
- we write $\Phi \vdash^{r_n} \mathcal{C}$ ($\mathcal{C}' \vdash^{r_n} \mathcal{C}$) iff clause $\mathcal{C}$ is the result of a one step application of rule $r_n \in \mathsf{R}$ to premise clauses $\mathcal{C}'_i \in \Phi$ (to $\mathcal{C}'$ respectively)
- multiple step derivations in calculus R are abbreviated by $\Phi_1 \vdash_{\mathsf{R}} \Phi_k$ (or $\mathcal{C}_1 \vdash_{\mathsf{R}} \mathcal{C}_k$)

Def.: General Bindings

Let $\alpha := (\overline{\beta^{\mathsf{l}}} \to \gamma)$ and let h be a constant or variable of type $(\overline{\delta_{\mathsf{m}}} \to \gamma)$ in Σ,

Def.: General Bindings

Let $\alpha := (\overline{\beta^{l}} \rightarrow \gamma)$ and let h be a constant or variable of type $(\overline{\delta_m} \rightarrow \gamma)$ in Σ, then

$$\mathbf{G} := \lambda \overline{X^{l}_{\beta^{l}}}.h\ \overline{\mathbf{V}^{m}}$$

($m \geq 0$) is called a partial binding of type α and head h (see also [SnGa89,Snyder91]),

Def.: General Bindings

Let $\alpha := (\overline{\beta^l} \rightarrow \gamma)$ and let h be a constant or variable of type $(\overline{\delta_m} \rightarrow \gamma)$ in Σ, then

$$\mathbf{G} := \lambda \overline{X^l_{\beta^l}}.h\ \overline{\mathbf{V}^m}$$

($m \geq 0$) is called a partial binding of type α and head h (see also [SnGa89,Snyder91]), if $\mathbf{V}^i = H^i\ \overline{X^l_{\beta_l}}$ and the H^i are new variables of types $\overline{\beta^l} \rightarrow \delta^i$.

Def.: General Bindings

Let $\alpha := (\overline{\beta^l} \rightarrow \gamma)$ and let h be a constant or variable of type $(\overline{\delta_m} \rightarrow \gamma)$ in Σ, then

$$\mathbf{G} := \lambda \overline{\mathsf{X}^l_{\beta^l}}.\mathsf{h}\ \overline{\mathbf{V}^m}$$

$(m \geq 0)$ is called a partial binding of type α and head h (see also [SnGa89,Snyder91]), if $\mathbf{V}^i = \mathsf{H}^i\ \overline{\mathsf{X}^l_{\beta_l}}$ and the H^i are new variables of types $\overline{\beta^l} \rightarrow \delta^i$.

Partial bindings, where the head is a bound variable $\mathsf{X}^j_{\beta_j}$ are called projection bindings (we write them as $\mathcal{G}^j_\alpha$) and imitation bindings (written $\mathcal{G}^h_\alpha$) otherwise.

Def.: General Bindings

Let $\alpha := (\overline{\beta^l} \to \gamma)$ and let h be a constant or variable of type $(\overline{\delta_m} \to \gamma)$ in Σ, then

$$\mathbf{G} := \lambda \overline{X^l_{\beta^l}}.\mathsf{h}\ \overline{\mathbf{V}^m}$$

$(m \geq 0)$ is called a partial binding of type α and head h (see also [SnGa89,Snyder91]), if $\mathbf{V}^i = H^i\ \overline{X^l_{\beta_l}}$ and the H^i are new variables of types $\overline{\beta^l} \to \delta^i$.

Partial bindings, where the head is a bound variable $X^j_{\beta_j}$ are called projection bindings (we write them as $\mathcal{G}^j_\alpha$) and imitation bindings (written $\mathcal{G}^h_\alpha$) otherwise.

Since we need both imitation and projection bindings for higher-order unification, we collect them in the set of general bindings for h and α ($\mathcal{AB}^h_\alpha := \{\mathcal{G}^h_\alpha\} \cup \{\mathcal{G}^j_\alpha \mid j \leq l\}$).

Def.: General Bindings

Let $\alpha := (\overline{\beta^l} \to \gamma)$ and let h be a constant or variable of type $(\overline{\delta_m} \to \gamma)$ in Σ, then

$$\mathbf{G} := \lambda \overline{X^l_{\beta^l}}.\mathsf{h}\ \overline{\mathbf{V}^m}$$

($m \geq 0$) is called a partial binding of type α and head h (see also [SnGa89,Snyder91]), if $\mathbf{V}^i = H^i\ \overline{X^l_{\beta_l}}$ and the H^i are new variables of types $\overline{\beta^l} \to \delta^i$.

Partial bindings, where the head is a bound variable $X^j_{\beta_j}$ are called projection bindings (we write them as $\mathcal{G}^j_\alpha$) and imitation bindings (written $\mathcal{G}^h_\alpha$) otherwise.

Since we need both imitation and projection bindings for higher-order unification, we collect them in the set of general bindings for h and α ($\mathcal{AB}^h_\alpha := \{\mathcal{G}^h_\alpha\} \cup \{\mathcal{G}^j_\alpha \mid j \leq l\}$).

Def.: Literals

- literals, e.g., $[\mathbf{A}]^\mu$, consist of a literal atom $\mathbf{A}$ and a polarity $\mu \in \{\mathsf{T}, \mathsf{F}\}$

Def.: Literals

- literals, e.g., $[\mathbf{A}]^\mu$, consist of a literal atom $\mathbf{A}$ and a polarity $\mu \in \{\mathsf{T}, \mathsf{F}\}$
- we distinguish between proper literals and pre-literals: the (normalised) atom of a pre-literal has a logical constant at head position, whereas this must not be the case for proper literals

Def.: Literals

- literals, e.g., $[\mathbf{A}]^{\mu}$, consist of a literal atom $\mathbf{A}$ and a polarity $\mu \in \{\mathsf{T}, \mathsf{F}\}$
- we distinguish between proper literals and pre-literals: the (normalised) atom of a pre-literal has a logical constant at head position, whereas this must not be the case for proper literals
- for instance, $[\mathbf{A} \vee \mathbf{B}]^{\mathsf{T}}$ is a pre-literal and $[\mathsf{p}_{\mathsf{o}\to\mathsf{o}}\ (\mathbf{A} \vee \mathbf{B})]^{\mathsf{T}}$ is a proper literal

Def.: Literals

- literals, e.g., $[\mathbf{A}]^{\mu}$, consist of a literal atom $\mathbf{A}$ and a polarity $\mu \in \{\mathsf{T}, \mathsf{F}\}$
- we distinguish between proper literals and pre-literals: the (normalised) atom of a pre-literal has a logical constant at head position, whereas this must not be the case for proper literals
- for instance, $[\mathbf{A} \lor \mathbf{B}]^{\mathsf{T}}$ is a pre-literal and $[p_{o \to o}\ (\mathbf{A} \lor \mathbf{B})]^{\mathsf{T}}$ is a proper literal
- a literal is called flexible if its atom contains a variable at head position

Def.: Unification Constraints

- a unification problem between two terms $\mathbf{T}^1$ and $\mathbf{T}^2$ (between n terms $\mathbf{T}^1, \ldots, \mathbf{T}^{\mathsf{n}}$) generated during the refutation process is called an unification constraint

Def.: Unification Constraints

- a unification problem between two terms $\mathbf{T}^1$ and $\mathbf{T}^2$ (between n terms $\mathbf{T}^1, \ldots, \mathbf{T}^\mathsf{n}$) generated during the refutation process is called an unification constraint
- it is represented as $[\mathbf{T}^1 \neq^? \mathbf{T}^2]$ (resp. $[\neq^? (\mathbf{T}^1, \ldots, \mathbf{T}^\mathsf{n})]$)

Def.: Unification Constraints

- a unification problem between two terms $\mathbf{T}^1$ and $\mathbf{T}^2$ (between n terms $\mathbf{T}^1, \ldots, \mathbf{T}^\mathsf{n}$) generated during the refutation process is called an unification constraint
- it is represented as $[\mathbf{T}^1 \neq^? \mathbf{T}^2]$ (resp. $[\neq^? (\mathbf{T}^1, \ldots, \mathbf{T}^\mathsf{n})]$)
- a unification constraint is called a flex-flex pair if both unification terms have flexible heads, i.e. variables at head position

Def.: Unification Constraints

- a unification problem between two terms $\mathbf{T}^1$ and $\mathbf{T}^2$ (between n terms $\mathbf{T}^1, \ldots, \mathbf{T}^{\mathsf{n}}$) generated during the refutation process is called an unification constraint
- it is represented as $[\mathbf{T}^1 \neq^? \mathbf{T}^2]$ (resp. $[\neq^? (\mathbf{T}^1, \ldots, \mathbf{T}^{\mathsf{n}})]$)
- a unification constraint is called a flex-flex pair if both unification terms have flexible heads, i.e. variables at head position
- a unification constraint is called a flex-rigid pair if one unification term has a flexible head, i.e. variable at head position

Def.: Clauses

- clauses consist of disjunctions of literals or unification constraints

Def.: Clauses

- clauses consist of disjunctions of literals or unification constraints
- the unification constraints specify conditions under which the other literals are valid

Def.: Clauses

- clauses consist of disjunctions of literals or unification constraints
- the unification constraints specify conditions under which the other literals are valid
- for instance, the clause $[\mathsf{p}_{\alpha\to\beta\to o}\ \mathbf{T}^1_\alpha\ \mathbf{T}^2_\beta]^{\mathsf{T}} \vee [\mathbf{T}^1_\alpha \neq^? \mathbf{S}^1_\alpha] \vee [\mathbf{T}^2_\beta \neq^? \mathbf{S}^2_\beta]$ can be read as: if $\mathbf{T}^1$ is unifiable with $\mathbf{S}^1$ and $\mathbf{T}^2$ with $\mathbf{S}^2$ then $(\mathsf{p}\ \mathbf{T}^1\ \mathbf{T}^2)$ holds

Def.: Clauses

- clauses consist of disjunctions of literals or unification constraints
- the unification constraints specify conditions under which the other literals are valid
- for instance, the clause $[\mathsf{p}_{\alpha\to\beta\to o}\ \mathbf{T}^1_\alpha\ \mathbf{T}^2_\beta]^{\mathsf{T}} \vee [\mathbf{T}^1_\alpha \neq^? \mathbf{S}^1_\alpha] \vee [\mathbf{T}^2_\beta \neq^? \mathbf{S}^2_\beta]$ can be read as: if $\mathbf{T}^1$ is unifiable with $\mathbf{S}^1$ and $\mathbf{T}^2$ with $\mathbf{S}^2$ then $(\mathsf{p}\ \mathbf{T}^1\ \mathbf{T}^2)$ holds
- we implicitly treat the disjunction operator $\vee$ in clauses as commutative and associative

Def.: Clauses

- clauses consist of disjunctions of literals or unification constraints
- the unification constraints specify conditions under which the other literals are valid
- for instance, the clause $[\mathsf{p}_{\alpha\to\beta\to o}\ \mathbf{T}^1_\alpha\ \mathbf{T}^2_\beta]^{\mathsf{T}} \vee [\mathbf{T}^1_\alpha \neq^? \mathbf{S}^1_\alpha] \vee [\mathbf{T}^2_\beta \neq^? \mathbf{S}^2_\beta]$ can be read as: if $\mathbf{T}^1$ is unifiable with $\mathbf{S}^1$ and $\mathbf{T}^2$ with $\mathbf{S}^2$ then $(\mathsf{p}\ \mathbf{T}^1\ \mathbf{T}^2)$ holds
- we implicitly treat the disjunction operator $\vee$ in clauses as commutative and associative
- additionally we presuppose commutativity of $\neq^?$ and implicitly identify any two α-equal constraints or literals.

Def.: Clauses

- clauses consist of disjunctions of literals or unification constraints
- the unification constraints specify conditions under which the other literals are valid
- for instance, the clause $[\mathsf{p}_{\alpha\rightarrow\beta\rightarrow o}\ \mathbf{T}^1_\alpha\ \mathbf{T}^2_\beta]^{\mathsf{T}} \vee [\mathbf{T}^1_\alpha \neq^? \mathbf{S}^1_\alpha] \vee [\mathbf{T}^2_\beta \neq^? \mathbf{S}^2_\beta]$ can be read as: if $\mathbf{T}^1$ is unifiable with $\mathbf{S}^1$ and $\mathbf{T}^2$ with $\mathbf{S}^2$ then $(\mathsf{p}\ \mathbf{T}^1\ \mathbf{T}^2)$ holds
- we implicitly treat the disjunction operator $\vee$ in clauses as commutative and associative
- additionally we presuppose commutativity of $\neq^?$ and implicitly identify any two α-equal constraints or literals.
- furthermore we assume that any two clauses have disjoint sets of free variables, i.e. for each freshly generated clause we choose new free variables

Def.: Clauses (contd.)

- if a clause contains at least one pre-literal we call it a pre-clause, otherwise a proper clause

Def.: Clauses (contd.)

- if a clause contains at least one pre-literal we call it a pre-clause, otherwise a proper clause
- a clause is called empty, denoted by $\square$, if it consists only of (possibly none) flex-flex pairs.

Rem.: Skolemisation

- an important aspect of clause normalisation is Skolemisation

Rem.: Skolemisation

- an important aspect of clause normalisation is Skolemisation
- we employ Miller's sound adaptation of traditional first-order Skolemisation [Miller:pihol83], which associates with each Skolem function the minimum number of arguments the Skolem function has to be applied to

Rem.: Skolemisation

- an important aspect of clause normalisation is Skolemisation
- we employ Miller's sound adaptation of traditional first-order Skolemisation [Miller:pihol83], which associates with each Skolem function the minimum number of arguments the Skolem function has to be applied to
- higher-order Skolemisation becomes sound, if any Skolem function f^n only occurs in a Skolem term, i.e., a formula $\mathbf{S} = f^n\overline{\mathbf{A}^n}$, where none of the $\mathbf{A}^i$ contains a bound variable

Rem.: Skolemisation

- an important aspect of clause normalisation is Skolemisation
- we employ Miller's sound adaptation of traditional first-order Skolemisation [Miller:pihol83], which associates with each Skolem function the minimum number of arguments the Skolem function has to be applied to
- higher-order Skolemisation becomes sound, if any Skolem function f^n only occurs in a Skolem term, i.e., a formula $\mathbf{S} = f^n\overline{\mathbf{A}^n}$, where none of the $\mathbf{A}^i$ contains a bound variable
- thus, the Skolem terms only serve as descriptions of the existential witnesses and never appear as functions proper

Rem.: Skolemisation

- an important aspect of clause normalisation is Skolemisation
- we employ Miller's sound adaptation of traditional first-order Skolemisation [Miller:pihol83], which associates with each Skolem function the minimum number of arguments the Skolem function has to be applied to
- higher-order Skolemisation becomes sound, if any Skolem function f^n only occurs in a Skolem term, i.e., a formula $\mathbf{S} = f^n \overline{\mathbf{A}^n}$, where none of the $\mathbf{A}^i$ contains a bound variable
- thus, the Skolem terms only serve as descriptions of the existential witnesses and never appear as functions proper
- without this additional restriction the calculi do not really become unsound, but one can prove an instance of the axiom of choice ([Andrews73]), which we want to treat as an optional axiom for the resolution calculi presented here

Approaches to Higher-Order Resolution: $\mathcal{R}$

Andrews' Higher-Order Resolution $\mathcal{R}$

We present and discuss Andrews' higher-order resolution calculus [Andrews71] in our uniform notation; we call this calculus $\mathcal{R}$

λ-Conversion

- Andrews' provides two rules for α-conversion and β-reduction

Andrews' Higher-Order Resolution $\mathcal{R}$

We present and discuss Andrews' higher-order resolution calculus [Andrews71] in our uniform notation; we call this calculus $\mathcal{R}$

λ-Conversion

- Andrews' provides two rules for α-conversion and β-reduction
- he does not provide a rule for η-conversion: consequently η-equality of two terms (e.g., $f_{\iota\rightarrow\iota} \doteq \lambda X_\iota . f\, X$) cannot be proven in this approach without employing the functional extensionality axiom of appropriate type

Andrews' Higher-Order Resolution $\mathcal{R}$

We present and discuss Andrews' higher-order resolution calculus [Andrews71] in our uniform notation; we call this calculus $\mathcal{R}$

λ-Conversion

- Andrews' provides two rules for α-conversion and β-reduction
- he does not provide a rule for η-conversion: consequently η-equality of two terms (e.g., $f_{\iota\rightarrow\iota} \doteq \lambda X_\iota . f\ X$) cannot be proven in this approach without employing the functional extensionality axiom of appropriate type
- we omit explicit rules for α- and β-convertibility and instead treat them implicitly, i.e. we assume that the presented rules operate on input and generate output in β-normal form and we automatically identify terms which differ only with respect to the names of bound variables

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation

- $\mathcal{R}$ introduces only four rules belonging to clause normalisation: negation elimination, conjunction elimination, existential elimination, and universal elimination

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation

- $\mathcal{R}$ introduces only four rules belonging to clause normalisation: negation elimination, conjunction elimination, existential elimination, and universal elimination
- as our presentation of clauses in contrast to [Andrews71] explicitly mentions the polarities of clauses and brackets the literal atoms we need additional structural rules, e.g., the rule $\vee^T$

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation

- $\mathcal{R}$ introduces only four rules belonging to clause normalisation: negation elimination, conjunction elimination, existential elimination, and universal elimination
- as our presentation of clauses in contrast to [Andrews71] explicitly mentions the polarities of clauses and brackets the literal atoms we need additional structural rules, e.g., the rule $\vee^{\mathsf{T}}$
- negation elimination: $$\frac{\mathbf{C} \vee [\neg \mathbf{A}]^{\mathsf{T}}}{\mathbf{C} \vee [\mathbf{A}]^{\mathsf{F}}}\ \neg^{\mathsf{T}} \qquad \frac{\mathbf{C} \vee [\neg \mathbf{A}]^{\mathsf{F}}}{\mathbf{C} \vee [\mathbf{A}]^{\mathsf{T}}}\ \neg^{\mathsf{F}}$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation

- $\mathcal{R}$ introduces only four rules belonging to clause normalisation: negation elimination, conjunction elimination, existential elimination, and universal elimination
- as our presentation of clauses in contrast to [Andrews71] explicitly mentions the polarities of clauses and brackets the literal atoms we need additional structural rules, e.g., the rule $\vee^T$
- negation elimination: $\dfrac{\mathbf{C} \vee [\neg \mathbf{A}]^T}{\mathbf{C} \vee [\mathbf{A}]^F}\ \neg^T \qquad \dfrac{\mathbf{C} \vee [\neg \mathbf{A}]^F}{\mathbf{C} \vee [\mathbf{A}]^T}\ \neg^F$
- conjunction/disjunction elimination:

$$\frac{\mathbf{C} \vee [\mathbf{A} \vee \mathbf{B}]^T}{\mathbf{C} \vee [\mathbf{A}]^T \vee [\mathbf{B}]^T}\ \vee^T \qquad \frac{\mathbf{C} \vee [\mathbf{A} \vee \mathbf{B}]^F}{\mathbf{C} \vee [\mathbf{A}]^F}\ \vee^F_l \qquad \frac{\mathbf{C} \vee [\mathbf{A} \vee \mathbf{B}]^F}{\mathbf{C} \vee [\mathbf{B}]^F}\ \vee^F_r$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation (contd.)

- existential/universal elimination:

$$\frac{\mathbf{C} \vee [\Pi^{\alpha} \mathbf{A}]^{\mathsf{T}}}{\mathbf{C} \vee [\mathbf{A}\ \mathsf{X}_{\alpha}]^{\mathsf{T}}}\ \Pi^{\mathsf{T}} \qquad \frac{\mathbf{C} \vee [\Pi^{\alpha} \mathbf{A}]^{\mathsf{F}}}{\mathbf{C} \vee [\mathbf{A}\ \mathsf{s}_{\alpha}]^{\mathsf{F}}}\ \Pi^{\mathsf{F}}$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation (contd.)

- existential/universal elimination:

$$\frac{\mathbf{C} \vee [\Pi^\alpha \mathbf{A}]^{\mathsf{T}}}{\mathbf{C} \vee [\mathbf{A}\ \mathsf{X}_\alpha]^{\mathsf{T}}}\ \Pi^{\mathsf{T}} \qquad \frac{\mathbf{C} \vee [\Pi^\alpha \mathbf{A}]^{\mathsf{F}}}{\mathbf{C} \vee [\mathbf{A}\ \mathsf{s}_\alpha]^{\mathsf{F}}}\ \Pi^{\mathsf{F}}$$

X_α is a new free variable and s_α is a new Skolem term

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation (contd.)

- existential/universal elimination:

$$\frac{\mathbf{C} \lor [\Pi^\alpha \mathbf{A}]^{\mathsf{T}}}{\mathbf{C} \lor [\mathbf{A}\ \mathsf{X}_\alpha]^{\mathsf{T}}}\ \Pi^{\mathsf{T}} \qquad \frac{\mathbf{C} \lor [\Pi^\alpha \mathbf{A}]^{\mathsf{F}}}{\mathbf{C} \lor [\mathbf{A}\ \mathsf{s}_\alpha]^{\mathsf{F}}}\ \Pi^{\mathsf{F}}$$

  X_α is a new free variable and s_α is a new Skolem term

- additionally Andrews presents rules addressing commutativity and associativity of the $\lor$-operator connecting the clauses literals; we have already mentioned the implicit treatment of these aspects here

Andrews' Higher-Order Resolution $\mathcal{R}$

Clause Normalisation (contd.)

- existential/universal elimination:

$$\frac{\mathbf{C} \vee [\Pi^{\alpha}\mathbf{A}]^{\mathsf{T}}}{\mathbf{C} \vee [\mathbf{A}\ \mathsf{X}_{\alpha}]^{\mathsf{T}}}\ \Pi^{\mathsf{T}} \qquad \frac{\mathbf{C} \vee [\Pi^{\alpha}\mathbf{A}]^{\mathsf{F}}}{\mathbf{C} \vee [\mathbf{A}\ \mathsf{s}_{\alpha}]^{\mathsf{F}}}\ \Pi^{\mathsf{F}}$$

  X_{α} is a new free variable and s_{α} is a new Skolem term

- additionally Andrews presents rules addressing commutativity and associativity of the $\vee$-operator connecting the clauses literals; we have already mentioned the implicit treatment of these aspects here

- we refer with $\mathsf{Cnf}(\mathbf{A})$ to the set of clauses obtained from formula $\mathbf{A}$ by exhaustive clause normalisation

Andrews' Higher-Order Resolution $\mathcal{R}$

Resolution & Factorisation

- Instead of a resolution and a factorisation rule — which work in connection with unification — Andrews presents a simplification and a cut rule. The cut rule is only applicable to clauses with two complementary literals which have identical atoms. Similarly Sim is defined only for clauses with two identical literals. In order to generate identical literal atoms during the refutation process these two rules have to be combined with the substitution rule Sub presented below.

Andrews' Higher-Order Resolution $\mathcal{R}$

Resolution & Factorisation

- Instead of a resolution and a factorisation rule — which work in connection with unification — Andrews presents a simplification and a cut rule. The cut rule is only applicable to clauses with two complementary literals which have identical atoms. Similarly Sim is defined only for clauses with two identical literals. In order to generate identical literal atoms during the refutation process these two rules have to be combined with the substitution rule Sub presented below.

- Simplification: $$\frac{[\mathbf{A}]^{\mu} \vee [\mathbf{A}]^{\mu} \vee \mathbf{C}}{[\mathbf{A}]^{\mu} \vee \mathbf{C}}\ \text{Sim}$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Resolution & Factorisation

- Instead of a resolution and a factorisation rule — which work in connection with unification — Andrews presents a simplification and a cut rule. The cut rule is only applicable to clauses with two complementary literals which have identical atoms. Similarly Sim is defined only for clauses with two identical literals. In order to generate identical literal atoms during the refutation process these two rules have to be combined with the substitution rule Sub presented below.
- Simplification: $$\frac{[\mathbf{A}]^{\mu} \vee [\mathbf{A}]^{\mu} \vee \mathbf{C}}{[\mathbf{A}]^{\mu} \vee \mathbf{C}}\ \text{Sim}$$
- Cut: $$\frac{[\mathbf{A}]^{\mu} \vee \mathbf{C} \quad [\mathbf{A}]^{\nu} \vee \mathbf{D}}{\mathbf{C} \vee \mathbf{D}}\ \text{Cut}$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Unification & Primitive Substitution

- As higher-order unification was still an open problem in 1971 calculus $\mathcal{R}$ employs the British museum method instead, i.e. it provides a substitution rule that allows to blindly instantiate free variables by arbitrary terms. As the instantiated terms may contain logical constants, instantiation of variables in proper clauses may lead to pre-clauses, which must be normalised again with the clause normalisation rules.

Andrews' Higher-Order Resolution $\mathcal{R}$

Unification & Primitive Substitution

- As higher-order unification was still an open problem in 1971 calculus $\mathcal{R}$ employs the British museum method instead, i.e. it provides a substitution rule that allows to blindly instantiate free variables by arbitrary terms. As the instantiated terms may contain logical constants, instantiation of variables in proper clauses may lead to pre-clauses, which must be normalised again with the clause normalisation rules.

- Substitution of arbitrary terms: $$\frac{\mathcal{C}}{\mathcal{C}_{[\mathbf{T}_\alpha/\mathrm{X}_\alpha]}}\ \mathrm{Sub}$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Unification & Primitive Substitution

- As higher-order unification was still an open problem in 1971 calculus $\mathcal{R}$ employs the British museum method instead, i.e. it provides a substitution rule that allows to blindly instantiate free variables by arbitrary terms. As the instantiated terms may contain logical constants, instantiation of variables in proper clauses may lead to pre-clauses, which must be normalised again with the clause normalisation rules.
- Substitution of arbitrary terms:

$$\frac{\mathcal{C}}{\mathcal{C}_{[\mathbf{T}_\alpha/\mathrm{X}_\alpha]}}\ \mathrm{Sub}$$

X_α is a free variable occurring in $\mathcal{C}$.

Andrews' Higher-Order Resolution $\mathcal{R}$

Extensionality Treatment

- Calculus $\mathcal{R}$ does not provide rules addressing the functional and/or Boolean extensionality principles.

Andrews' Higher-Order Resolution $\mathcal{R}$

Extensionality Treatment

- Calculus $\mathcal{R}$ does not provide rules addressing the functional and/or Boolean extensionality principles.
- Instead $\mathcal{R}$ assumes that the following extensionality axioms are (in form of respective clauses) explicitly added to the search space. And since the functional extensionality principle is parameterised over arbitrary functional types infinitely many functional extensionality axioms are required.

Andrews' Higher-Order Resolution $\mathcal{R}$

Extensionality Treatment

- Calculus $\mathcal{R}$ does not provide rules addressing the functional and/or Boolean extensionality principles.
- Instead $\mathcal{R}$ assumes that the following extensionality axioms are (in form of respective clauses) explicitly added to the search space. And since the functional extensionality principle is parameterised over arbitrary functional types infinitely many functional extensionality axioms are required.
- Extensionality axioms

$$\mathbf{EXT}^{\doteq}_{\alpha\to\beta}: \quad \forall F_{\alpha\to\beta}. \forall G_{\alpha\to\beta}. (\forall X_\beta. F\,X \doteq G\,X) \Rightarrow F \doteq G$$

$$\mathbf{EXT}^{\doteq}_{o}: \quad \forall A_o. \forall B_o. (A \Leftrightarrow B) \Rightarrow A \doteq^o B$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Extensionality Treatment (contd.)

- The extensionality clauses derived from the extensionality axioms have the following form (note the many free variables, especially at literal head position, that are introduced into the search space – they heavily increase the amount of blind search in any attempt to automate the calculus):

Andrews' Higher-Order Resolution $\mathcal{R}$

Extensionality Treatment (contd.)

- The extensionality clauses derived from the extensionality axioms have the following form (note the many free variables, especially at literal head position, that are introduced into the search space – they heavily increase the amount of blind search in any attempt to automate the calculus):

$$\mathcal{E}_{\mathbf{1}}^{\alpha\to\beta} : [\mathsf{p}\ (\mathbf{F}\ \mathsf{s})]^{\mathsf{T}} \vee [\mathbf{Q}\ \mathbf{F}]^{\mathsf{F}} \vee [\mathbf{Q}\ \mathbf{G}]^{\mathsf{T}}$$
$$\mathcal{E}_{\mathbf{2}}^{\alpha\to\beta} : [\mathsf{p}\ (\mathbf{G}\ \mathsf{s})]^{\mathsf{F}} \vee [\mathbf{Q}\ \mathbf{F}]^{\mathsf{F}} \vee [\mathbf{Q}\ \mathbf{G}]^{\mathsf{T}}$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Extensionality Treatment (contd.)

- The extensionality clauses derived from the extensionality axioms have the following form (note the many free variables, especially at literal head position, that are introduced into the search space – they heavily increase the amount of blind search in any attempt to automate the calculus):

$$\mathcal{E}_{\mathbf{1}}^{\alpha\to\beta} : [\mathsf{p}\ (\mathbf{F}\ \mathsf{s})]^{\mathsf{T}} \lor [\mathbf{Q}\ \mathbf{F}]^{\mathsf{F}} \lor [\mathbf{Q}\ \mathbf{G}]^{\mathsf{T}}$$
$$\mathcal{E}_{\mathbf{2}}^{\alpha\to\beta} : [\mathsf{p}\ (\mathbf{G}\ \mathsf{s})]^{\mathsf{F}} \lor [\mathbf{Q}\ \mathbf{F}]^{\mathsf{F}} \lor [\mathbf{Q}\ \mathbf{G}]^{\mathsf{T}}$$

$$\mathcal{E}_{\mathbf{1}}^{\mathbf{o}} : [\mathbf{A}]^{\mathsf{F}} \lor [\mathbf{B}]^{\mathsf{F}} \lor [\mathbf{P}\ \mathbf{A}]^{\mathsf{F}} \lor [\mathbf{P}\ \mathbf{B}]^{\mathsf{T}}$$
$$\mathcal{E}_{\mathbf{2}}^{\mathbf{o}} : [\mathbf{A}]^{\mathsf{T}} \lor [\mathbf{B}]^{\mathsf{T}} \lor [\mathbf{P}\ \mathbf{A}]^{\mathsf{F}} \lor [\mathbf{P}\ \mathbf{B}]^{\mathsf{T}}$$

Andrews' Higher-Order Resolution $\mathcal{R}$

Extensionality Treatment (contd.)

- The extensionality clauses derived from the extensionality axioms have the following form (note the many free variables, especially at literal head position, that are introduced into the search space – they heavily increase the amount of blind search in any attempt to automate the calculus):

$$\mathcal{E}_{\mathbf{1}}^{\alpha\to\beta} : [\mathsf{p}\ (\mathbf{F}\ \mathsf{s})]^{\mathsf{T}} \vee [\mathbf{Q}\ \mathbf{F}]^{\mathsf{F}} \vee [\mathbf{Q}\ \mathbf{G}]^{\mathsf{T}}$$
$$\mathcal{E}_{\mathbf{2}}^{\alpha\to\beta} : [\mathsf{p}\ (\mathbf{G}\ \mathsf{s})]^{\mathsf{F}} \vee [\mathbf{Q}\ \mathbf{F}]^{\mathsf{F}} \vee [\mathbf{Q}\ \mathbf{G}]^{\mathsf{T}}$$

$$\mathcal{E}_{\mathbf{1}}^{\mathbf{o}} : [\mathbf{A}]^{\mathsf{F}} \vee [\mathbf{B}]^{\mathsf{F}} \vee [\mathbf{P}\ \mathbf{A}]^{\mathsf{F}} \vee [\mathbf{P}\ \mathbf{B}]^{\mathsf{T}}$$
$$\mathcal{E}_{\mathbf{2}}^{\mathbf{o}} : [\mathbf{A}]^{\mathsf{T}} \vee [\mathbf{B}]^{\mathsf{T}} \vee [\mathbf{P}\ \mathbf{A}]^{\mathsf{F}} \vee [\mathbf{P}\ \mathbf{B}]^{\mathsf{T}}$$

$\mathsf{p}_{\beta\to o}$, s_α are Skolem terms and A_o, B_o, $\mathsf{P}_{o\to o}$, $\mathsf{Q}_{(\alpha\to\beta)\to o}$ are new free variables.

Andrews' Higher-Order Resolution $\mathcal{R}$

Proof Search

- initially the proof problem is negated and normalised

Andrews' Higher-Order Resolution $\mathcal{R}$

Proof Search

- initially the proof problem is negated and normalised
- proof search then starts with the normalised clauses and applies the cut and simplification rule in close connection with the substitution rule

Andrews' Higher-Order Resolution $\mathcal{R}$

Proof Search

- initially the proof problem is negated and normalised
- proof search then starts with the normalised clauses and applies the cut and simplification rule in close connection with the substitution rule
- intermediate applications of the clause normalisation rules may be needed to normalise temporarily generated pre-clauses

Andrews' Higher-Order Resolution $\mathcal{R}$

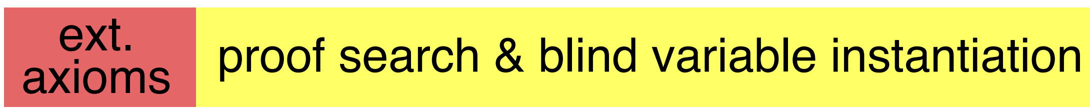
Proof Search

- initially the proof problem is negated and normalised
- proof search then starts with the normalised clauses and applies the cut and simplification rule in close connection with the substitution rule
- intermediate applications of the clause normalisation rules may be needed to normalise temporarily generated pre-clauses
- the extensionality treatment in $\mathcal{R}$ simply assumes to add at the beginning of the refutation process the above clauses obtained from the extensionality axioms

Andrews' Higher-Order Resolution $\mathcal{R}$

Proof Search

- initially the proof problem is negated and normalised
- proof search then starts with the normalised clauses and applies the cut and simplification rule in close connection with the substitution rule
- intermediate applications of the clause normalisation rules may be needed to normalise temporarily generated pre-clauses
- the extensionality treatment in $\mathcal{R}$ simply assumes to add at the beginning of the refutation process the above clauses obtained from the extensionality axioms
- the proof search can be graphically illustrated as follows:



Andrews' Higher-Order Resolution $\mathcal{R}$

Completeness

- [Andrews71] gives a completeness proof for calculus $\mathcal{R}$ with respect to the semantical notion of V-complexes (corresponds to our weakest model class $\mathfrak{M}_\beta(\Sigma)$)

Andrews' Higher-Order Resolution $\mathcal{R}$

Completeness

- [Andrews71] gives a completeness proof for calculus $\mathcal{R}$ with respect to the semantical notion of V-complexes (corresponds to our weakest model class $\mathfrak{M}_\beta(\Sigma)$)
- as the extensionality principles are not valid in this rather weak semantical structures, the extensionality axioms are not needed in this completeness proof

Andrews' Higher-Order Resolution $\mathcal{R}$

Completeness

- [Andrews71] gives a completeness proof for calculus $\mathcal{R}$ with respect to the semantical notion of V-complexes (corresponds to our weakest model class $\mathfrak{M}_\beta(\Sigma)$)
- as the extensionality principles are not valid in this rather weak semantical structures, the extensionality axioms are not needed in this completeness proof
- Theorem: (V-completeness of $\mathcal{R}$) The calculus $\mathcal{R}$ is (sound and) complete with respect to the notion of V-complexes.

  Proof: [Andrews71].

Andrews' Higher-Order Resolution $\mathcal{R}$

Henkin Completeness

- We can also prove Henkin completeness of calculus $\mathcal{R}$.

Andrews' Higher-Order Resolution $\mathcal{R}$

Henkin Completeness

- We can also prove Henkin completeness of calculus $\mathcal{R}$.
- Theorem: (Henkin completeness of $\mathcal{R}$) The calculus $\mathcal{R}$ is (sound and) complete with respect to Henkin semantics provided that the infinitely many extensionality axioms are given.

  Proof: exercise

Andrews' Higher-Order Resolution $\mathcal{R}$

Henkin Completeness

- We can also prove Henkin completeness of calculus $\mathcal{R}$.
- Theorem: (Henkin completeness of $\mathcal{R}$) The calculus $\mathcal{R}$ is (sound and) complete with respect to Henkin semantics provided that the infinitely many extensionality axioms are given.

  Proof: exercise

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{R}$?

- Leibniz equality and η-equality:

$$\mathsf{f}_{\iota\to\iota} \doteq \lambda \mathsf{X}_{\iota}\text{.}\mathsf{f}\ \mathsf{X}$$

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{R}$?

- Leibniz equality and η-equality:

$$\mathsf{f}_{\iota\to\iota} \doteq \lambda \mathsf{X}_\iota.\mathsf{f}\ \mathsf{X}$$

- The set of all red balls equals the set of all balls that are red: $\{X|\text{red } X \land \text{ball } X\} = \{X|\text{ball } X \land \text{red } X\}$. This problem can be encoded as

$$(\lambda X_\iota.\text{red } X \land \text{ball } X) = (\lambda X_\iota.\text{ball } X \land \text{red } X)$$

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{R}$?

- All unary logical operators $O_{o \to o}$ which map the propositions a and b to $\top$ consequently also map $a \wedge b$ to $\top$:

$$\forall O_{o \to o} \centerdot (O\ a_o) \wedge (O\ b_o) \Rightarrow (O\ (a_o \wedge b_o))$$

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{R}$?

- In Henkin semantics the domain $\mathcal{D}_o$ of all Booleans contains exactly the truth values $\bot$ and $\top$. Consequently the domain of all mappings from Booleans to Booleans contains exactly contains in each Henkin model at most four elements. And because of the requirement, that the function domains in Henkin models must be rich enough such that every term has a denotation, it follows that $\mathcal{D}_{o\to o}$ contains exactly the pairwise distinct denotations of the following four terms: $\lambda X_o . X_o$, $\lambda X_o . \neg X_o$, $\lambda X_o . \bot$, and $\lambda X_o . \top$. This theorem can be formulated as follows (where $f_{o\to o}$ is a constant):

$$(f = \lambda X_o . X_o) \lor (f = \lambda X_o . \neg X_o) \lor (f = \lambda X_o . \bot) \lor (f = \lambda X_o . \top)$$

Approaches to Higher-Order Resolution: $\mathcal{CR}$

Huet's Constrained Resolution $\mathcal{CR}$

We transform Huet's constrained resolution approach [Huet72,Huet73] in our uniform notation. The calculus here is the unsorted fragment of the variant of Huet's approach as presented in [Kohlhase94]. In the remainder of this paper we refer to this calculus with $\mathcal{CR}$.

λ-Conversion

- Calculus $\mathcal{CR}$ assumes that terms, literals, and clauses are implicitly reduced to β-normal form.

Huet's Constrained Resolution $\mathcal{CR}$

We transform Huet's constrained resolution approach [Huet72,Huet73] in our uniform notation. The calculus here is the unsorted fragment of the variant of Huet's approach as presented in [Kohlhase94]. In the remainder of this paper we refer to this calculus with $\mathcal{CR}$.

λ-Conversion

- Calculus $\mathcal{CR}$ assumes that terms, literals, and clauses are implicitly reduced to β-normal form.
- Furthermore, we assume that α-equality is treated implicitly, i.e. we identify all terms that differ only with respect to the names of bound variables.

Huet's Constrained Resolution $\mathcal{CR}$

Clause Normalisation

- [Huet72] does not explicitly present clause normalisation rules but assumes that they are given. Here we employ the rules $\neg^{\mathsf{T}}$, $\neg^{\mathsf{F}}$, $\vee^{\mathsf{T}}$, $\vee^{\mathsf{F}}_{\mathsf{l}}$, $\vee^{\mathsf{F}}_{\mathsf{r}}$, Π^{T}, and Π^{F} as already defined for calculus $\mathcal{R}$ before.

Huet's Constrained Resolution $\mathcal{CR}$

Clause Normalisation

- [Huet72] does not explicitly present clause normalisation rules but assumes that they are given. Here we employ the rules $\neg^{T}$, $\neg^{F}$, $\vee^{T}$, $\vee^{F}_{l}$, $\vee^{F}_{r}$, Π^{T}, and Π^{F} as already defined for calculus $\mathcal{R}$ before.

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation

- As first-order unification is decidable and unitary it can be employed as a strong filter in first-order resolution [Robinson65].

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation

- As first-order unification is decidable and unitary it can be employed as a strong filter in first-order resolution [Robinson65].
- Unfortunately higher-order unification is not decidable (cf. [Lucchesi72,Huet73,Goldfarb81]) and thus it can not be applied in the sense of a terminating side computation in higher-order theorem proving.

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation

- As first-order unification is decidable and unitary it can be employed as a strong filter in first-order resolution [Robinson65].
- Unfortunately higher-order unification is not decidable (cf. [Lucchesi72,Huet73,Goldfarb81]) and thus it can not be applied in the sense of a terminating side computation in higher-order theorem proving.
- Huet therefore suggests in [Huet72,Huet73] to delay the unification process and to explicitly encode unification problems occurring during the refutation search as unification constraints.

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation (contd.)

- In his original approach Huet presented a hyper-resolution rule which simultaneously resolves on the resolution literals $\mathbf{A}^1, \ldots \mathbf{A}^n$ $(1 \leq n)$ and $\mathbf{B}^1, \ldots \mathbf{B}^m$ $(1 \leq m)$ of two given clauses and adds the unification constraint $[\neq^? (\mathbf{A}^1, \ldots \mathbf{A}^n, \mathbf{B}^1, \ldots \mathbf{B}^m)]$ to the resolvent:

$$\frac{[\mathbf{A}^1]^\mu \vee \ldots \vee [\mathbf{A}^n]^\mu \vee \mathbf{C} \quad [\mathbf{B}^1]^\nu \vee \ldots \vee [\mathbf{B}^m]^\nu \vee \mathbf{D}}{\mathbf{C} \vee \mathbf{D} \vee [\neq^? (\mathbf{A}^1, \ldots \mathbf{A}^n, \mathbf{B}^1, \ldots \mathbf{B}^m)]} \text{ Hres}$$

(where $\mu \neq \nu$).

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation (contd.)

- In order to ease the comparison with the two other approaches discussed in this paper we instead employ a resolution rule Res and a factorisation rule Fac.

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation (contd.)

- In order to ease the comparison with the two other approaches discussed in this paper we instead employ a resolution rule Res and a factorisation rule Fac.
- Like Hres both rules encode the unification problem to be solved as a unification constraint:

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation (contd.)

- In order to ease the comparison with the two other approaches discussed in this paper we instead employ a resolution rule Res and a factorisation rule Fac.
- Like Hres both rules encode the unification problem to be solved as a unification constraint:
- Constrained resolution: $$\frac{[\mathbf{A}]^{\mu} \vee \mathbf{C} \quad [\mathbf{B}]^{\nu} \vee \mathbf{D}}{\mathbf{C} \vee \mathbf{D} \vee [\mathbf{A} \neq^{?} \mathbf{B}]} \text{ Res}$$

(where $\mu \neq \nu$).

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation (contd.)

- In order to ease the comparison with the two other approaches discussed in this paper we instead employ a resolution rule Res and a factorisation rule Fac.
- Like Hres both rules encode the unification problem to be solved as a unification constraint:
- Constrained resolution:

$$\frac{[\mathbf{A}]^{\mu} \vee \mathbf{C} \quad [\mathbf{B}]^{\nu} \vee \mathbf{D}}{\mathbf{C} \vee \mathbf{D} \vee [\mathbf{A} \neq^{?} \mathbf{B}]}\ \text{Res}$$

(where $\mu \neq \nu$).

- Constrained factorisation:

$$\frac{[\mathbf{A}]^{\mu} \vee [\mathbf{B}]^{\mu} \vee \mathbf{C}}{[\mathbf{A}]^{\mu} \vee \mathbf{C} \vee [\mathbf{A} \neq^{?} \mathbf{B}]^{\mathsf{F}}}\ \text{Fac}$$

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation (contd.)

- One can easily prove by induction on $n + m$ that each proof step applying rule Hres can be replaced by a corresponding derivation employing Res and Fac.

Huet's Constrained Resolution $\mathcal{CR}$

Resolution & Factorisation (contd.)

- One can easily prove by induction on $n + m$ that each proof step applying rule Hres can be replaced by a corresponding derivation employing Res and Fac.
- For a formal proof note that the unification constraint $[\neq^? (\mathbf{A}^1, \ldots \mathbf{A}^n, \mathbf{B}^1, \ldots \mathbf{B}^m)]$ is equivalent to $[\mathbf{A}^1 \neq^? \mathbf{A}^2] \vee [\mathbf{A}^2 \neq^? \mathbf{A}^3] \vee \ldots \vee [\mathbf{A}^{n-1} \neq^? \mathbf{A}^n] \vee [\mathbf{A}^n \neq^? \mathbf{B}^1] \vee [\mathbf{B}^1 \neq^? \mathbf{B}^2] \vee [\mathbf{B}^2 \neq^? \mathbf{B}^3] \vee \ldots \vee [\mathbf{B}^{n-1} \neq^? \mathbf{B}^n]$.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting

- [Huet75] introduces higher-order unification and higher-order pre-unification and shows that higher-order pre-unification is sufficient to verify the soundness of a refutation in which the occurring unification problems have been delayed until the end.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting

- [Huet75] introduces higher-order unification and higher-order pre-unification and shows that higher-order pre-unification is sufficient to verify the soundness of a refutation in which the occurring unification problems have been delayed until the end.
- The higher-order pre-unification rules presented here are discussed in detail in [Benzmüller-PhD-99]. They furthermore closely reflect the rules as presented in [SnyderGallier89].

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting

- [Huet75] introduces higher-order unification and higher-order pre-unification and shows that higher-order pre-unification is sufficient to verify the soundness of a refutation in which the occurring unification problems have been delayed until the end.
- The higher-order pre-unification rules presented here are discussed in detail in [Benzmüller-PhD-99]. They furthermore closely reflect the rules as presented in [SnyderGallier89].
- Elimination of trivial pairs:

$$\frac{\mathbf{C} \vee [\mathbf{A} \neq^{?} \mathbf{A}]}{\mathbf{C}}\ \text{Triv}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting

- [Huet75] introduces higher-order unification and higher-order pre-unification and shows that higher-order pre-unification is sufficient to verify the soundness of a refutation in which the occurring unification problems have been delayed until the end.
- The higher-order pre-unification rules presented here are discussed in detail in [Benzmüller-PhD-99]. They furthermore closely reflect the rules as presented in [SnyderGallier89].
- Elimination of trivial pairs: $$\frac{\mathbf{C} \vee [\mathbf{A} \neq^? \mathbf{A}]}{\mathbf{C}}\ \mathsf{Triv}$$
- Decomposition $$\frac{\mathsf{C} \vee [\mathsf{h}\overline{\mathbf{U}^n} \neq^? \mathsf{h}\overline{\mathbf{V}^n}]}{\mathsf{C} \vee [\mathbf{U}^1 \neq^? \mathbf{V}^1] \vee \ldots \vee [\mathbf{U}^n \neq^? \mathbf{V}^n]}\ \mathsf{Dec}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Elimination of λ-binders: (weak functional extensionality)

$$\frac{\mathbf{C} \vee [\mathbf{M}_{\alpha\rightarrow\beta} \neq^? \mathbf{N}_{\alpha\rightarrow\beta}]}{\mathbf{C} \vee [\mathbf{M}\ \mathsf{s}_\alpha \neq^? \mathbf{N}\ \mathsf{s}_\alpha]}\ \text{Func}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Elimination of λ-binders: (weak functional extensionality)

$$\frac{\mathbf{C} \vee [\mathbf{M}_{\alpha\rightarrow\beta} \neq^? \mathbf{N}_{\alpha\rightarrow\beta}]}{\mathbf{C} \vee [\mathbf{M}\ \mathsf{s}_\alpha \neq^? \mathbf{N}\ \mathsf{s}_\alpha]}\ \text{Func}$$

s_α is a new Skolem term.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Elimination of λ-binders: (weak functional extensionality)

$$\frac{\mathbf{C} \vee [\mathbf{M}_{\alpha\to\beta} \neq^? \mathbf{N}_{\alpha\to\beta}]}{\mathbf{C} \vee [\mathbf{M}\ \mathsf{s}_\alpha \neq^? \mathbf{N}\ \mathsf{s}_\alpha]}\ \text{Func}$$

s_α is a new Skolem term.

- Imitation of rigid heads:

$$\frac{\mathbf{C} \vee [\mathsf{F}_\gamma\ \overline{\mathbf{U}^\mathsf{n}} \neq^? \mathsf{h}\ \overline{\mathbf{V}^\mathsf{m}}] \quad \mathbf{G} \in \mathcal{AB}^\mathsf{h}_\gamma}{\mathbf{C} \vee [\mathsf{F} \neq^? \mathbf{G}] \vee [\mathsf{F}\ \overline{\mathbf{U}^\mathsf{n}} \neq^? \mathsf{h}\ \overline{\mathbf{V}^\mathsf{m}}]}\ \text{FlexRigid}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Elimination of λ-binders: (weak functional extensionality)

$$\frac{\mathbf{C} \vee [\mathbf{M}_{\alpha\to\beta} \neq^? \mathbf{N}_{\alpha\to\beta}]}{\mathbf{C} \vee [\mathbf{M}\ \mathsf{s}_\alpha \neq^? \mathbf{N}\ \mathsf{s}_\alpha]}\ \text{Func}$$

s_α is a new Skolem term.

- Imitation of rigid heads:

$$\frac{\mathbf{C} \vee [\mathsf{F}_\gamma\ \overline{\mathbf{U}^n} \neq^? \mathsf{h}\ \overline{\mathbf{V}^m}] \quad \mathbf{G} \in \mathcal{AB}^{\mathsf{h}}_\gamma}{\mathbf{C} \vee [\mathsf{F} \neq^? \mathbf{G}] \vee [\mathsf{F}\ \overline{\mathbf{U}^n} \neq^? \mathsf{h}\ \overline{\mathbf{V}^m}]}\ \text{FlexRigid}$$

$\mathcal{AB}^{\mathsf{h}}_\gamma$ is the set of general bindings of type γ for head h.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Huet points to the usefulness of eager unification to filter out clauses with non-unifiable unification constraints or to back-propagate the solutions of easily solvable constraints (e.g., in case of first-order unification problems occurring during the proof search): many of the higher-order unification problems occurring in practice are decidable and have only finitely many solutions.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Huet points to the usefulness of eager unification to filter out clauses with non-unifiable unification constraints or to back-propagate the solutions of easily solvable constraints (e.g., in case of first-order unification problems occurring during the proof search): many of the higher-order unification problems occurring in practice are decidable and have only finitely many solutions.
- Hence, even though higher-order unification is generally not decidable it is sensible in practice to apply the unification algorithm with a particular resource, such that only those unification problems which may have further solutions beyond this bound need to be delayed.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- In our presentation of calculus $\mathcal{CR}$ we explicitly address the aspect of eager unification and substitution by rule Subst. This rule back-propagates eagerly computed unifiers to the literal part of a clause.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- In our presentation of calculus $\mathcal{CR}$ we explicitly address the aspect of eager unification and substitution by rule Subst. This rule back-propagates eagerly computed unifiers to the literal part of a clause.
- Eager unification & substitution:

$$\frac{\mathbf{C} \vee [\mathsf{X} \neq^{?} \mathbf{A}] \quad \mathsf{X} \notin \mathsf{free}(\mathbf{A})}{\mathbf{C}_{[\mathbf{A}/\mathsf{X}]}}\ \mathsf{Subst}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- The literal heads of our clauses may consist of set variables and it may be necessary to instantiate them with terms introducing new logical constant at head position in order to find a refutation.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- The literal heads of our clauses may consist of set variables and it may be necessary to instantiate them with terms introducing new logical constant at head position in order to find a refutation.
- Unfortunately not all appropriate instantiations can be computed with the calculus rules presented so far.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- The literal heads of our clauses may consist of set variables and it may be necessary to instantiate them with terms introducing new logical constant at head position in order to find a refutation.
- Unfortunately not all appropriate instantiations can be computed with the calculus rules presented so far.
- To address this problem Huet's approach provides the following splitting rules:

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Instantiate
  set variables:

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Instantiate set variables:

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{T}} \vee [\mathsf{R}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} (\mathsf{Q}_{o} \vee \mathsf{R}_{o})]}\ \mathsf{S}^{\mathsf{T}}_{\vee}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Instantiate set variables:

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{T}} \vee [\mathsf{R}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} (\mathsf{Q}_{o} \vee \mathsf{R}_{o})]}\ \mathsf{S}_{\vee}^{\mathsf{T}}$$

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{F}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} (\mathsf{Q}_{o} \vee \mathsf{R}_{o})] \quad [\mathsf{R}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} (\mathsf{Q}_{o} \vee \mathsf{R}_{o})]}\ \mathsf{S}_{\vee}^{\mathsf{F}}$$

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mu} \vee \mathbf{C}}{[\mathsf{Q}]^{\nu} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} \neg \mathsf{Q}_{o}]}\ \mathsf{S}_{\neg}^{\mathsf{TF}}$$

(where $\mu \neq \nu$)

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Instantiate set variables:

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{T}} \vee [\mathsf{R}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A} \ \neq^? (\mathsf{Q_o} \vee \mathsf{R_o})]}\ \mathsf{S}^{\mathsf{T}}_{\vee}$$

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mu} \vee \mathbf{C}}{[\mathsf{Q}]^{\nu} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A} \ \neq^? \neg \mathsf{Q_o}]}\ \mathsf{S}^{\mathsf{TF}}_{\neg}$$

(where $\mu \neq \nu$)

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{F}} \vee \mathbf{C}}{\begin{array}{c}[\mathsf{Q}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A} \ \neq^? (\mathsf{Q_o} \vee \mathsf{R_o})] \\ [\mathsf{R}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A} \ \neq^? (\mathsf{Q_o} \vee \mathsf{R_o})]\end{array}}\ \mathsf{S}^{\mathsf{F}}_{\vee}$$

$$\frac{[\mathsf{P}\ \mathbf{A}_{\alpha\rightarrow o}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{M}_{\alpha\rightarrow o}\ \mathsf{Z}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A} \ \neq^? \Pi^{\alpha}\mathsf{M}]}\ \mathsf{S}^{\mathsf{T}}_{\Pi}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Instantiate set variables:

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{T}} \vee [\mathsf{R}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} (\mathsf{Q}_{\mathsf{o}} \vee \mathsf{R}_{\mathsf{o}})]}\ \mathsf{S}^{\mathsf{T}}_{\vee}$$

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{F}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} (\mathsf{Q}_{\mathsf{o}} \vee \mathsf{R}_{\mathsf{o}})] \quad [\mathsf{R}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} (\mathsf{Q}_{\mathsf{o}} \vee \mathsf{R}_{\mathsf{o}})]}\ \mathsf{S}^{\mathsf{F}}_{\vee}$$

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mu} \vee \mathbf{C}}{[\mathsf{Q}]^{\nu} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} \neg \mathsf{Q}_{\mathsf{o}}]}\ \mathsf{S}^{\mathsf{TF}}_{\neg}$$

(where $\mu \neq \nu$)

$$\frac{[\mathsf{P}\ \mathbf{A}_{\alpha\rightarrow\mathsf{o}}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{M}_{\alpha\rightarrow\mathsf{o}}\ \mathsf{Z}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} \Pi^{\alpha}\mathsf{M}]}\ \mathsf{S}^{\mathsf{T}}_{\Pi}$$

$$\frac{[\mathsf{P}\ \mathbf{A}_{\alpha\rightarrow\mathsf{o}}]^{\mathsf{F}} \vee \mathbf{C}}{[\mathsf{M}_{\alpha\rightarrow\mathsf{o}}\ \mathsf{s}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \neq^{?} \Pi^{\alpha}\mathsf{M}]}\ \mathsf{S}^{\mathsf{F}}_{\Pi}$$

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- Instantiate set variables:

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{T}} \vee [\mathsf{R}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \ \neq^{?} (\mathsf{Q}_{o} \vee \mathsf{R}_{o})]}\ S_{\vee}^{\mathsf{T}}$$

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mathsf{F}} \vee \mathbf{C}}{[\mathsf{Q}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \ \neq^{?} (\mathsf{Q}_{o} \vee \mathsf{R}_{o})] \quad [\mathsf{R}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \ \neq^{?} (\mathsf{Q}_{o} \vee \mathsf{R}_{o})]}\ S_{\vee}^{\mathsf{F}}$$

$$\frac{[\mathsf{P}\ \mathbf{A}]^{\mu} \vee \mathbf{C}}{[\mathsf{Q}]^{\nu} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \ \neq^{?} \neg \mathsf{Q}_{o}]}\ S_{\neg}^{\mathsf{TF}}$$

(where $\mu \neq \nu$)

$$\frac{[\mathsf{P}\ \mathbf{A}_{\alpha \to o}]^{\mathsf{T}} \vee \mathbf{C}}{[\mathsf{M}_{\alpha \to o}\ \mathsf{Z}]^{\mathsf{T}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \ \neq^{?} \Pi^{\alpha}\mathsf{M}]}\ S_{\Pi}^{\mathsf{T}}$$

$$\frac{[\mathsf{P}\ \mathbf{A}_{\alpha \to o}]^{\mathsf{F}} \vee \mathbf{C}}{[\mathsf{M}_{\alpha \to o}\ \mathsf{s}]^{\mathsf{F}} \vee \mathbf{C} \vee [\mathsf{P}\ \mathbf{A}\ \ \neq^{?} \Pi^{\alpha}\mathsf{M}]}\ S_{\Pi}^{\mathsf{F}}$$

- S_{Π}^{T} and S_{Π}^{F} are infinitely branching as they are parameterised over type α. $\mathsf{Q}_{o}, \mathsf{R}_{o}, \mathsf{M}_{\alpha \to o}, \mathsf{Z}_{\alpha}$ are new variables and s_{α} is a new Skolem constant.

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- A theorem which is not refutable in $\mathcal{CR}$ if the splitting rules are not available is $\exists A_o . A$:

Huet's Constrained Resolution $\mathcal{CR}$

Unification & Splitting (contd.)

- A theorem which is not refutable in $\mathcal{CR}$ if the splitting rules are not available is $\exists A_o.A$:
- After negation this statement normalises to clause $\mathcal{C}_1 : [A]^F$, such that none but the splitting rules are applicable. With the help of rule $S_{\neg}^{TF}$ and eager unification, however, we can derive $\mathcal{C}_2 : [A']^T$ which is then successfully resolvable against $\mathcal{C}_1$.

Huet's Constrained Resolution $\mathcal{CR}$

Extensionality Treatment

- On the one hand η-convertibility is built-in in higher-order unification, such that calculus $\mathcal{CR}$ already supports functional extensionality reasoning to a certain extend.

Huet's Constrained Resolution $\mathcal{CR}$

Extensionality Treatment

- On the one hand η-convertibility is built-in in higher-order unification, such that calculus $\mathcal{CR}$ already supports functional extensionality reasoning to a certain extend.
- On the other hand $\mathcal{CR}$ nevertheless fails to address full extensionality as it does not realise the required subtle interplay between the functional and Boolean extensionality principles.

Huet's Constrained Resolution $\mathcal{CR}$

Extensionality Treatment

- On the one hand η-convertibility is built-in in higher-order unification, such that calculus $\mathcal{CR}$ already supports functional extensionality reasoning to a certain extend.
- On the other hand $\mathcal{CR}$ nevertheless fails to address full extensionality as it does not realise the required subtle interplay between the functional and Boolean extensionality principles.
- Without employing additional (Boolean and functional!) extensionality axioms $\mathcal{CR}$ is, e.g., not able to prove the rather simple examples presented before.

Huet's Constrained Resolution $\mathcal{CR}$

Proof Search

- Initially the proof problem is negated and normalised. The main proof search then operates on the generated clauses by applying the resolution, factorisation, and splitting rules.

Huet's Constrained Resolution $\mathcal{CR}$

Proof Search

- Initially the proof problem is negated and normalised. The main proof search then operates on the generated clauses by applying the resolution, factorisation, and splitting rules.
- Despite the possibility of eager unification $\mathcal{CR}$ generally foresees to delay the higher-order unification process in order to overcome the undecidability problem.

Huet's Constrained Resolution $\mathcal{CR}$

Proof Search

- Initially the proof problem is negated and normalised. The main proof search then operates on the generated clauses by applying the resolution, factorisation, and splitting rules.
- Despite the possibility of eager unification $\mathcal{CR}$ generally foresees to delay the higher-order unification process in order to overcome the undecidability problem.
- When deriving a potentially empty clause (no normal literals), $\mathcal{CR}$ then tests whether the accumulated unification constraints justifying this particular refutation are solvable.

Huet's Constrained Resolution $\mathcal{CR}$

Proof Search (contd.)

- Like $\mathcal{R}$, the extensionality treatment of $\mathcal{CR}$ requires to add infinitely many extensionality axioms to the search space.

Huet's Constrained Resolution $\mathcal{CR}$

Proof Search (contd.)

- Like $\mathcal{R}$, the extensionality treatment of $\mathcal{CR}$ requires to add infinitely many extensionality axioms to the search space.
- The following figure graphically illustrates the main ideas of the proof search in $\mathcal{CR}$.

Huet's Constrained Resolution $\mathcal{CR}$

Completeness Results

- [Huet72,Huet73] analyses completeness of $\mathcal{CR}$ formally only with respect to Andrews V-complexes, i.e. Huet verifies that the set of non-refutable sentences in $\mathcal{CR}$ is an abstract consistency class for V-complexes.

Huet's Constrained Resolution $\mathcal{CR}$

Completeness Results

- [Huet72,Huet73] analyses completeness of $\mathcal{CR}$ formally only with respect to Andrews V-complexes, i.e. Huet verifies that the set of non-refutable sentences in $\mathcal{CR}$ is an abstract consistency class for V-complexes.
- Theorem (V-completeness of $\mathcal{CR}$): The calculus $\mathcal{CR}$ is complete with respect to the notion of V-complexes.

  Proof: [Huet72,Huet73]

Huet's Constrained Resolution $\mathcal{CR}$

Completeness Results

- [Huet72,Huet73] analyses completeness of $\mathcal{CR}$ formally only with respect to Andrews V-complexes, i.e. Huet verifies that the set of non-refutable sentences in $\mathcal{CR}$ is an abstract consistency class for V-complexes.
- Theorem (V-completeness of $\mathcal{CR}$): The calculus $\mathcal{CR}$ is complete with respect to the notion of V-complexes.

  Proof: [Huet72,Huet73]
- Theorem (Henkin completeness of $\mathcal{CR}$): The calculus $\mathcal{CR}$ is complete wrt. Henkin semantics provided that the infinitely many extensionality axioms are given.

  Proof: exercise

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{CR}$?

- Leibniz equality and η-equality:

$$\mathsf{f}_{\iota\to\iota} \doteq \lambda \mathsf{X}_\iota \mathbf{.} \mathsf{f}\ \mathsf{X}$$

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{CR}$?

- Leibniz equality and η-equality:

$$\mathsf{f}_{\iota\to\iota} \doteq \lambda \mathsf{X}_{\iota} \mathbf{.} \mathsf{f}\ \mathsf{X}$$

- The set of all red balls equals the set of all balls that are red: $\{\mathsf{X} | \mathsf{red}\ \mathsf{X} \wedge \mathsf{ball}\ \mathsf{X}\} = \{\mathsf{X} | \mathsf{ball}\ \mathsf{X} \wedge \mathsf{red}\ \mathsf{X}\}$. This problem can be encoded as

$$(\lambda \mathsf{X}_{\iota} \mathbf{.} \mathsf{red}\ \mathsf{X} \wedge \mathsf{ball}\ \mathsf{X}) = (\lambda \mathsf{X}_{\iota} \mathbf{.} \mathsf{ball}\ \mathsf{X} \wedge \mathsf{red}\ \mathsf{X})$$

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{CR}$?

- All unary logical operators $O_{o \to o}$ which map the propositions a and b to $\top$ consequently also map $a \wedge b$ to $\top$:

$$\forall O_{o \to o} \centerdot (O\ a_o) \wedge (O\ b_o) \Rightarrow (O\ (a_o \wedge b_o))$$

Example Proofs

Exercise: How are the following theorems proved in calculus $\mathcal{CR}$?

- In Henkin semantics the domain $\mathcal{D}_o$ of all Booleans contains exactly the truth values $\bot$ and $\top$. Consequently the domain of all mappings from Booleans to Booleans contains exactly contains in each Henkin model at most four elements. And because of the requirement, that the function domains in Henkin models must be rich enough such that every term has a denotation, it follows that $\mathcal{D}_{o\to o}$ contains exactly the pairwise distinct denotations of the following four terms: $\lambda X_o . X_o$, $\lambda X_o . \neg X_o$, $\lambda X_o . \bot$, and $\lambda X_o . \top$. This theorem can be formulated as follows (where $f_{o\to o}$ is a constant):

$$(f = \lambda X_o . X_o) \lor (f = \lambda X_o . \neg X_o) \lor (f = \lambda X_o . \bot) \lor (f = \lambda X_o . \top)$$

Approaches to Higher-Order Resolution: $\mathcal{ER}$

Extensional HO Resolution $\mathcal{ER}$

Clause normalization

$$\frac{C \vee [A \vee B]^T}{C \vee [A]^T \vee [B]^T} \vee^T \qquad \frac{C \vee [A \vee B]^F}{C \vee [A]^F} \vee^F_l \qquad \frac{C \vee [A \vee B]^F}{C \vee [B]^F} \vee^F_r$$

$$\frac{C \vee [\neg A]^T}{C \vee [A]^F} \neg^T \qquad \frac{C \vee [\neg A]^F}{C \vee [A]^T} \neg^F$$

$$\frac{C \vee [\Pi^\alpha A]^T \quad X_\alpha \text{ new variable}}{C \vee [A\ X]^T} \Pi^T$$

$$\frac{C \vee [\Pi^\alpha A]^F \quad sk_\alpha \text{ Skolem term}}{C \vee [A\ sk_\alpha]^F} \Pi^F$$

This rules may be combined into a single rule Cnf.

Extensional HO Resolution $\mathcal{ER}$

Resolution and Factorisation

$$\frac{[\mathsf{N}]^\alpha \vee \mathsf{C} \quad [\mathsf{M}]^\beta \vee \mathsf{D} \quad \alpha \neq \beta}{\mathsf{C} \vee \mathsf{D} \vee [\mathsf{N} \neq^? \mathsf{M}]}\ \mathsf{Res}$$

$$\frac{[\mathsf{N}]^\alpha \vee [\mathsf{M}]^\alpha \vee \mathsf{C} \quad \alpha \in \{\mathsf{T}, \mathsf{F}\}}{[\mathsf{N}]^\alpha \vee \mathsf{C} \vee [\mathsf{N} \neq^? \mathsf{M}]}\ \mathsf{Fac}$$

$$\frac{[\mathsf{Q}_\gamma \overline{\mathsf{U}^\mathsf{k}}]^\alpha \vee \mathsf{C} \quad \mathsf{P} \in \mathcal{GB}_\gamma^{\{\neg,\vee\}\cup\{\Pi^\beta|\beta\in\mathcal{T}^\mathsf{k}\}}}{[\mathsf{Q}_\gamma \overline{\mathsf{U}^\mathsf{k}}]^\alpha \vee \mathsf{C} \vee [\mathsf{Q} \neq^? \mathsf{P}]}\ \mathsf{Prim}^\mathsf{k}$$

Extensional HO Resolution $\mathcal{ER}$

(Pre-)unification rules

$$\frac{C \vee [M_{\alpha\to\beta} \neq^? N_{\alpha\to\beta}]^F \quad s_\alpha \text{ Skolem-Term}}{C \vee [M\ s \neq^? N\ s]}\ \text{Func}$$

$$\frac{C \vee [h\overline{U^n} \neq^? h\overline{V^n}]}{C \vee [U^1 \neq^? V^1] \vee \ldots \vee [U^n \neq^? V^n]}\ \text{Dec} \qquad \frac{C \vee [A \neq^? A]}{C}\ \text{Triv}$$

$$\frac{C \vee [F_\gamma \overline{U^n} \neq^? h\overline{V^n}] \quad G \in \mathcal{GB}^h_\gamma}{C \vee [F \neq^? G] \vee [F\overline{U^n} \neq^? h\overline{V^n}]}\ \text{Flex/Rigid}$$

$$\frac{C \vee E \quad E \text{ solved for } C}{Cnf(subst_E(C))}\ \text{Subst}$$

Extensional HO Resolution $\mathcal{ER}$

Extensionality rules

$$\frac{C \vee [M_o \neq^? N_o]^F}{Cnf(C \vee [M_o \Leftrightarrow N_o]^F)}\ \text{Equiv}$$

$$\frac{C \vee [M_\alpha \neq^? N_\alpha]^F \quad \alpha \in \{o, \iota\}}{Cnf(C \vee [\forall P_{\alpha \to o} . PM \Rightarrow PN]^F)}\ \text{Leib}$$

Extensional HO Resolution $\mathcal{ER}$

Extensionality Treatment

- Instead of adding infinitely many extensionality axioms to the search space $\mathcal{CR}$ provides two new extensionality rules which closely connect refutation search and eager unification.

Extensional HO Resolution $\mathcal{ER}$

Extensionality Treatment

- Instead of adding infinitely many extensionality axioms to the search space $\mathcal{CR}$ provides two new extensionality rules which closely connect refutation search and eager unification.
- The idea is to allow for recursive calls from higher-order unification to the overall refutation process.

Extensional HO Resolution $\mathcal{ER}$

Extensionality Treatment

- Instead of adding infinitely many extensionality axioms to the search space $\mathcal{CR}$ provides two new extensionality rules which closely connect refutation search and eager unification.
- The idea is to allow for recursive calls from higher-order unification to the overall refutation process.
- This turns the rather weak syntactical higher-order unification approach considered so far into a most general approach for *dynamic* higher-order theory unification.

Extensional HO Resolution $\mathcal{ER}$

Proof Search

- Initially the proof problem is negated and normalised. The main proof search then closely interleaves the refutation process on resolution layer and unification, i.e. the main proof search rules Res, Fac, and Prim and the unification rules are integrated at a common conceptual level. The calls from unification to the overall refutation process with rules *Leib* and *Equiv* introduce new clauses into the search space which can be resolved against already given ones.

Extensional HO Resolution $\mathcal{ER}$

Proof Search

- Initially the proof problem is negated and normalised. The main proof search then closely interleaves the refutation process on resolution layer and unification, i.e. the main proof search rules Res, Fac, and Prim and the unification rules are integrated at a common conceptual level. The calls from unification to the overall refutation process with rules *Leib* and *Equiv* introduce new clauses into the search space which can be resolved against already given ones.
- The following figure graphically illustrates the main ideas of the proof search in $\mathcal{ER}$.

interleaved proof search & unification

Ex.: Extensional HO Resolution $\mathcal{ER}$

$$\forall B_{\alpha\to o}, C_{\alpha\to o}, D_{\alpha\to o}. B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Negation and definition expansion with

$\cup = \lambda A_{\alpha\to o}, B_{\alpha\to o}, X_{\alpha}. (A\ X) \vee (B\ X) \qquad \cap = \lambda A_{\alpha\to o}, B_{\alpha\to o}, X_{\alpha}. (A\ X) \wedge (B\ X)$

leads to:

$$C_1 : [\lambda X_{\alpha}. (b\ X) \vee ((c\ X) \wedge (d\ X)) \neq^{?} \lambda X_{\alpha}. ((b\ X) \vee (c\ X)) \wedge ((b\ X) \vee (d\ X)))]$$

Goal directed functional and Boolean extensionality treatment:

$$C_2 : [(b\ x) \vee ((c\ x) \wedge (d\ x)) \Leftrightarrow ((b\ x) \vee (c\ x)) \wedge ((b\ x) \vee (d\ x)))]^{F}$$

Clause normalization results then in a pure propositional, i.e. decidable, set of clauses. Only these clauses are still in the search space of LEO(in total there are 33 clauses generated and LEO finds the proof on a 2,5GHz PC in 820ms).

Similar proof in case of embedded propositions:

$$\forall P_{(\alpha\to o)\to o}, B_{\alpha\to o}, C_{\alpha\to o}, D_{\alpha\to o}. P(B \cup (C \cap D)) \Rightarrow P((B \cup C) \cap (B \cup D))$$

Ex.: Extensional HO Resolution $\mathcal{ER}$

$$\forall P_{o\to o}. (P\ a_o) \land (P\ b_o) \Rightarrow (P\ (a_o \land b_o))$$

Negation and clause normalization

$$\mathcal{C}_1 : [p\ a]^T \quad \mathcal{C}_2 : [p\ b]^T \quad \mathcal{C}_3 : [p\ (a \land b)]^F$$

Resolution between $\mathcal{C}_1$ and $\mathcal{C}_3$ and between $\mathcal{C}_2$ and $\mathcal{C}_3$

$$\mathcal{C}_4 : [p\ a \neq^? p\ (a \land b)] \quad \mathcal{C}_5 : [p\ b \neq^? p\ (a \land b)]$$

Decomposition

$$\mathcal{C}_6 : [a \neq^? (a \land b)] \quad \mathcal{C}_7 : [b \neq^? (a \land b)]$$

Recursive call of proof process with rules Equiv and Cnf

$$\mathcal{C}_8 : [a]^F \lor [b]^F \quad \mathcal{C}_9 : [a]^T \lor [b]^T \quad \mathcal{C}_{10} : [a]^T \quad \mathcal{C}_{11} : [b]^T$$

Ex.: Extensional HO Resolution $\mathcal{ER}$

Further small examples which test Henkin completeness:

$$\forall F_{o \to o} . (F \doteq \lambda X_o . X_o) \vee (F \doteq \lambda X_o . \neg X_o) \vee (F \doteq \lambda X_o . \bot) \vee (F \doteq \lambda X_o . \top)$$

$$\forall H_{o \to o} . H \; \bot \doteq H \; (H \; \top \doteq H \; \bot)$$

$$\ldots$$

Higher-Order Sequent Calculi, Cut, and Saturation

UNIVERSITÄT DES SAARLANDES

Def.: Sequent Calculi

- A sequent calculus $\mathcal{G}$ provides an inductive definition for when $\Vdash_{\mathcal{G}} \Delta$ holds.

Def.: Sequent Calculi

- A sequent calculus $\mathcal{G}$ provides an inductive definition for when $\Vdash_{\mathcal{G}} \Delta$ holds.
- Here we consider a one-sided sequent calculus, thus a sequent is defined as a finite set Δ of β-normal sentences from $cwff_o(\Sigma)$.

Def.: Sequent Calculi

- A sequent calculus $\mathcal{G}$ provides an inductive definition for when $\Vdash_{\mathcal{G}} \Delta$ holds.
- Here we consider a one-sided sequent calculus, thus a sequent is defined as a finite set Δ of β-normal sentences from $cwff_o(\Sigma)$.
- Let's compare our one-sided rules to standard two-sided sequent calculus rules (we use $\Delta * \mathbf{A}$ to denote the set $\Delta \cup \{\mathbf{A}\}$ (which is simply Δ if $\mathbf{A} \in \Delta$)):

Def.: Sequent Calculi

- A sequent calculus $\mathcal{G}$ provides an inductive definition for when $\Vdash_{\mathcal{G}} \Delta$ holds.
- Here we consider a one-sided sequent calculus, thus a sequent is defined as a finite set Δ of β-normal sentences from $cwff_o(\Sigma)$.
- Let's compare our one-sided rules to standard two-sided sequent calculus rules (we use $\Delta * \mathbf{A}$ to denote the set $\Delta \cup \{\mathbf{A}\}$ (which is simply Δ if $\mathbf{A} \in \Delta$)):

$$\frac{}{\Delta * (\mathbf{A} \vee \mathbf{B})}\ \mathcal{G}(\vee_+) \qquad \text{vs.} \qquad \frac{}{\Gamma \Longrightarrow \Delta * (\mathbf{A} \vee \mathbf{B})}\ \vee R$$

Def.: Sequent Calculi

- A sequent calculus $\mathcal{G}$ provides an inductive definition for when $\Vdash_{\mathcal{G}} \Delta$ holds.
- Here we consider a one-sided sequent calculus, thus a sequent is defined as a finite set Δ of β-normal sentences from $cwff_o(\Sigma)$.
- Let's compare our one-sided rules to standard two-sided sequent calculus rules (we use $\Delta * \mathbf{A}$ to denote the set $\Delta \cup \{\mathbf{A}\}$ (which is simply Δ if $\mathbf{A} \in \Delta$)):

$$\frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})}\, \mathcal{G}(\vee_+) \qquad \text{vs.} \qquad \frac{\Gamma \Longrightarrow \Delta * \mathbf{A} * \mathbf{B}}{\Gamma \Longrightarrow \Delta * (\mathbf{A} \vee \mathbf{B})}\, \vee R$$

Def.: Sequent Calculi

- A sequent calculus $\mathcal{G}$ provides an inductive definition for when $\Vdash_{\mathcal{G}} \Delta$ holds.
- Here we consider a one-sided sequent calculus, thus a sequent is defined as a finite set Δ of β-normal sentences from *cwff*$_{o}(\Sigma)$.
- Let's compare our one-sided rules to standard two-sided sequent calculus rules (we use $\Delta * \mathbf{A}$ to denote the set $\Delta \cup \{\mathbf{A}\}$ (which is simply Δ if $\mathbf{A} \in \Delta$)):

$$\frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})} \mathcal{G}(\vee_+) \quad \text{vs.} \quad \frac{\Gamma \Longrightarrow \Delta * \mathbf{A} * \mathbf{B}}{\Gamma \Longrightarrow \Delta * (\mathbf{A} \vee \mathbf{B})} \vee R$$

$$\frac{}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})} \mathcal{G}(\vee_-) \quad \text{vs.} \quad \frac{}{\Gamma * \mathbf{A} \vee \mathbf{B} \Longrightarrow \Delta} \vee L$$

Def.: Sequent Calculi

- A sequent calculus $\mathcal{G}$ provides an inductive definition for when $\Vdash_{\mathcal{G}} \Delta$ holds.
- Here we consider a one-sided sequent calculus, thus a sequent is defined as a finite set Δ of β-normal sentences from $cwff_o(\Sigma)$.
- Let's compare our one-sided rules to standard two-sided sequent calculus rules (we use $\Delta * \mathbf{A}$ to denote the set $\Delta \cup \{\mathbf{A}\}$ (which is simply Δ if $\mathbf{A} \in \Delta$)):

$$\frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})}\ \mathcal{G}(\vee_+) \qquad \text{vs.} \qquad \frac{\Gamma \Longrightarrow \Delta * \mathbf{A} * \mathbf{B}}{\Gamma \Longrightarrow \Delta * (\mathbf{A} \vee \mathbf{B})}\ \vee\mathrm{R}$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\ \mathcal{G}(\vee_-) \qquad \text{vs.} \qquad \frac{\Gamma * \mathbf{B} \Longrightarrow \Delta \quad \Gamma * \mathbf{A} \Longrightarrow \Delta}{\Gamma * \mathbf{A} \vee \mathbf{B} \Longrightarrow \Delta}\ \vee\mathrm{L}$$

Def.: Validity of Sequents

- Given a sequent Δ, a model $\mathcal{M}$, and a class $\mathfrak{M}$ of models, we say

Def.: Validity of Sequents

- Given a sequent Δ, a model $\mathcal{M}$, and a class $\mathfrak{M}$ of models, we say

$$\Delta \text{ is valid for } \mathcal{M} \text{ (or valid for } \mathfrak{M}\text{),}$$

Def.: Validity of Sequents

- Given a sequent Δ, a model $\mathcal{M}$, and a class $\mathfrak{M}$ of models, we say

$$\Delta \text{ is valid for } \mathcal{M} \text{ (or valid for } \mathfrak{M}\text{)},$$

if $\mathcal{M} \models \mathbf{D}$ for some $\mathbf{D} \in \Delta$ (or Δ is valid for every $\mathcal{M} \in \mathfrak{M}$).

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

$$\frac{\Delta_1 \quad \cdots \quad \Delta_n}{\Delta}\, r$$

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

$$\frac{\Delta_1 \quad \cdots \quad \Delta_n}{\Delta}\, r$$

is admissible in $\mathcal{G}$

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

$$\frac{\Delta_1 \quad \cdots \quad \Delta_n}{\Delta}\, r$$

is admissible in $\mathcal{G}$

if $\Vdash_{\mathcal{G}} \Delta$ holds whenever $\Vdash_{\mathcal{G}} \Delta_i$ for all $1 \leq i \leq n$.

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

$$\frac{\Delta_1 \quad \cdots \quad \Delta_n}{\Delta}\, r$$

is admissible in $\mathcal{G}$

if $\Vdash_{\mathcal{G}} \Delta$ holds whenever $\Vdash_{\mathcal{G}} \Delta_i$ for all $1 \leq i \leq n$.

- For any $k \geq 0$, we call an admissible rule r

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

$$\frac{\Delta_1 \quad \cdots \quad \Delta_n}{\Delta}\, r$$

is admissible in $\mathcal{G}$

if $\Vdash_{\mathcal{G}} \Delta$ holds whenever $\Vdash_{\mathcal{G}} \Delta_i$ for all $1 \leq i \leq n$.

- For any $k \geq 0$, we call an admissible rule r

k-admissible

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

$$\frac{\Delta_1 \quad \cdots \quad \Delta_n}{\Delta}\, r$$

is admissible in $\mathcal{G}$

if $\Vdash_{\mathcal{G}} \Delta$ holds whenever $\Vdash_{\mathcal{G}} \Delta_i$ for all $1 \leq i \leq n$.

- For any $k \geq 0$, we call an admissible rule r

k-admissible

if any instance of r can be replaced by a derivation with at most k additional proof steps.

Def.: k-Admissibility of Rules

- We say a sequent calculus rule

$$\frac{\Delta_1 \quad \cdots \quad \Delta_n}{\Delta}\, r$$

is admissible in $\mathcal{G}$

if $\Vdash_{\mathcal{G}} \Delta$ holds whenever $\Vdash_{\mathcal{G}} \Delta_i$ for all $1 \leq i \leq n$.

- For any $k \geq 0$, we call an admissible rule r

k-admissible

if any instance of r can be replaced by a derivation with at most k additional proof steps.

- Remark: We use admissibility to obtain more general results. In fact all rules that are later shown to be k-admissible are actually even k-derivable.

Def.: Sequent Calculus Rules

Basic Rules

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{}{\mathbf{\Delta} * \mathbf{A} * \neg\mathbf{A}}\ \mathcal{G}(init)$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\; \mathcal{G}(init)$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad\qquad \frac{}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

$$\frac{}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_{-})$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_{-})$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_-) \qquad \frac{}{\Delta * (\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_+)$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_-) \qquad \frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_+)$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_-) \qquad \frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_+)$$

$$\frac{}{\Delta * \neg\Pi^\alpha\mathbf{A}}\,\mathcal{G}(\mathit{\Pi}_-^{\mathbf{C}})$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\,\mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\,\mathcal{G}(\neg)$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_-) \qquad \frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})}\,\mathcal{G}(\vee_+)$$

$$\frac{\Delta * \neg(\mathbf{A}\mathbf{C}){\downarrow_\beta} \quad \mathbf{C} \in \mathit{cwff}_\alpha(\Sigma)}{\Delta * \neg\Pi^\alpha\mathbf{A}}\,\mathcal{G}(\Pi_-^{\mathbf{C}})$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}}\ \mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}}\ \mathcal{G}(\neg)$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})}\ \mathcal{G}(\vee_-) \qquad \frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})}\ \mathcal{G}(\vee_+)$$

$$\frac{\Delta * \neg(\mathbf{A}\mathbf{C}){\downarrow_\beta} \quad \mathbf{C} \in \mathit{cwff}_\alpha(\Sigma)}{\Delta * \neg\Pi^\alpha\mathbf{A}}\ \mathcal{G}(\Pi^{\mathbf{C}}_-) \qquad \frac{}{\Delta * \Pi^\alpha\mathbf{A}}\ \mathcal{G}(\Pi^c_+)$$

Def.: Sequent Calculus Rules

Basic Rules

$$\frac{\mathbf{A} \text{ atomic (and } \beta\text{-normal)}}{\Delta * \mathbf{A} * \neg\mathbf{A}} \mathcal{G}(init) \qquad \frac{\Delta * \mathbf{A}}{\Delta * \neg\neg\mathbf{A}} \mathcal{G}(\neg)$$

$$\frac{\Delta * \neg\mathbf{A} \quad \Delta * \neg\mathbf{B}}{\Delta * \neg(\mathbf{A} \vee \mathbf{B})} \mathcal{G}(\vee_-) \qquad \frac{\Delta * \mathbf{A} * \mathbf{B}}{\Delta * (\mathbf{A} \vee \mathbf{B})} \mathcal{G}(\vee_+)$$

$$\frac{\Delta * \neg(\mathbf{A}\mathbf{C}){\downarrow_\beta} \quad \mathbf{C} \in \mathit{cwff}_\alpha(\Sigma)}{\Delta * \neg\Pi^\alpha\mathbf{A}} \mathcal{G}(\Pi_-^{\mathbf{C}}) \qquad \frac{\Delta * (\mathbf{A}\mathsf{c}){\downarrow_\beta} \quad \mathsf{c}_\alpha \in \Sigma \text{ new}}{\Delta * \Pi^\alpha\mathbf{A}} \mathcal{G}(\Pi_+^c)$$

Def.: Sequent Calculus Rules

Inversion Rule

Weakening and Cut Rules

Def.: Sequent Calculus Rules

Inversion Rule

$$\frac{}{\Delta * \mathbf{A}}\ \mathcal{G}(Inv^{\neg})$$

Weakening and Cut Rules

Def.: Sequent Calculus Rules

Inversion Rule

$$\frac{\mathbf{\Delta} * \neg\neg\mathbf{A}}{\mathbf{\Delta} * \mathbf{A}}\,\mathcal{G}(Inv^{\neg})$$

Weakening and Cut Rules

Def.: Sequent Calculus Rules

Inversion Rule

$$\frac{\Delta * \neg\neg \mathbf{A}}{\Delta * \mathbf{A}}\ \mathcal{G}(Inv^{\neg})$$

Weakening and Cut Rules

$$\frac{}{\Delta \cup \Delta'}\ \mathcal{G}(weak)$$

Def.: Sequent Calculus Rules

Inversion Rule

$$\frac{\Delta * \neg\neg\mathbf{A}}{\Delta * \mathbf{A}}\,\mathcal{G}(Inv^{\neg})$$

Weakening and Cut Rules

$$\frac{\Delta}{\Delta \cup \Delta'}\,\mathcal{G}(weak)$$

Def.: Sequent Calculus Rules

Inversion Rule

$$\frac{\mathbf{\Delta} * \neg\neg\mathbf{A}}{\mathbf{\Delta} * \mathbf{A}}\ \mathcal{G}(Inv^{\neg})$$

Weakening and Cut Rules

$$\frac{\mathbf{\Delta}}{\mathbf{\Delta} \cup \mathbf{\Delta}'}\ \mathcal{G}(weak)$$

$$\frac{}{\mathbf{\Delta}}\ \mathcal{G}(cut)$$

Def.: Sequent Calculus Rules

Inversion Rule

$$\frac{\mathbf{\Delta} * \neg\neg\mathbf{A}}{\mathbf{\Delta} * \mathbf{A}}\,\mathcal{G}(Inv^{\neg})$$

Weakening and Cut Rules

$$\frac{\mathbf{\Delta}}{\mathbf{\Delta} \cup \mathbf{\Delta}'}\,\mathcal{G}(weak)$$

$$\frac{\mathbf{\Delta} * \mathbf{C} \quad \mathbf{\Delta} * \neg\mathbf{C}}{\mathbf{\Delta}}\,\mathcal{G}(cut)$$

ACC for Sequent Calculi

- For any sequent calculus $\mathcal{G}$ we can define a class $\Gamma_{\Sigma}^{\mathcal{G}}$ of sets of sentences.

ACC for Sequent Calculi

- For any sequent calculus $\mathcal{G}$ we can define a class $\Gamma_\Sigma^{\mathcal{G}}$ of sets of sentences.
- Under certain assumptions, $\Gamma_\Sigma^{\mathcal{G}}$ is an abstract consistency class.

ACC for Sequent Calculi

- For any sequent calculus $\mathcal{G}$ we can define a class $\Gamma_{\Sigma}^{\mathcal{G}}$ of sets of sentences.
- Under certain assumptions, $\Gamma_{\Sigma}^{\mathcal{G}}$ is an abstract consistency class.
- First we adopt the notation $\neg\Phi$ and $\Phi{\downarrow}_{\beta}$ for the sets $\{\neg\mathbf{A} | \mathbf{A} \in \Phi\}$ and $\{\mathbf{A}{\downarrow}_{\beta} | \mathbf{A} \in \Phi\}$, resp., where $\Phi \subseteq \mathit{cwff}_{o}(\Sigma)$.

ACC for Sequent Calculi

- For any sequent calculus $\mathcal{G}$ we can define a class $\Gamma_\Sigma^{\mathcal{G}}$ of sets of sentences.
- Under certain assumptions, $\Gamma_\Sigma^{\mathcal{G}}$ is an abstract consistency class.
- First we adopt the notation $\neg\Phi$ and $\Phi\big\downarrow_\beta$ for the sets $\{\neg\mathbf{A}|\mathbf{A}\in\Phi\}$ and $\{\mathbf{A}\big\downarrow_\beta|\mathbf{A}\in\Phi\}$, resp., where $\Phi\subseteq \mathit{cwff}_o(\Sigma)$.
- Furthermore, we assume this use of $\neg$ binds more strongly than $\cup$ or $*$, so that $\neg\Phi\cup\Delta$ means $(\neg\Phi)\cup\Delta$ and $\neg\Phi * \mathbf{A}$ means $(\neg\Phi) * \mathbf{A}$.

Def.: ACC for Sequent Calculi

Let $\mathcal{G}$ be a sequent calculus.

Def.: ACC for Sequent Calculi

Let $\mathcal{G}$ be a sequent calculus. We define $\Gamma_\Sigma^\mathcal{G}$ to be the class of all finite $\Phi \subset cwff_o(\Sigma)$ such that $\Vdash_\mathcal{G} \neg\Phi\downarrow_\beta$ does not hold.

Lemma: Consequence of $\mathcal{G}(Inv^{\neg})$

Let $\mathcal{G}$ be a sequent calculus such that $\mathcal{G}(Inv^{\neg})$ is admissible.

Lemma: Consequence of $\mathcal{G}(Inv^{\neg})$

Let $\mathcal{G}$ be a sequent calculus such that $\mathcal{G}(Inv^{\neg})$ is admissible. For any finite sets Φ and Δ of sentences, if $\Phi \cup \neg\Delta \notin \Gamma_{\Sigma}^{\mathcal{G}}$, then $\Vdash_{\mathcal{G}} \neg\Phi\downarrow_{\beta} \cup \Delta\downarrow_{\beta}$ holds.

Lemma: Consequence of $\mathcal{G}(Inv^{\neg})$

Let $\mathcal{G}$ be a sequent calculus such that $\mathcal{G}(Inv^{\neg})$ is admissible. For any finite sets Φ and Δ of sentences, if $\Phi \cup \neg\Delta \notin \Gamma_{\Sigma}^{\mathcal{G}}$, then $\Vdash_{\mathcal{G}} \neg\Phi\downarrow_{\beta} \cup \Delta\downarrow_{\beta}$ holds.

Proof:

Lemma: Consequence of $\mathcal{G}(Inv^{\neg})$

Let $\mathcal{G}$ be a sequent calculus such that $\mathcal{G}(Inv^{\neg})$ is admissible. For any finite sets Φ and Δ of sentences, if $\Phi \cup \neg\Delta \notin \Gamma_{\Sigma}^{\mathcal{G}}$, then $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_{\beta}} \cup \Delta{\downarrow_{\beta}}$ holds.

Proof: Suppose $\Phi \cup \neg\Delta \notin \Gamma_{\Sigma}^{\mathcal{G}}$.

Lemma: Consequence of $\mathcal{G}(Inv^{\neg})$

Let $\mathcal{G}$ be a sequent calculus such that $\mathcal{G}(Inv^{\neg})$ is admissible. For any finite sets Φ and Δ of sentences, if $\Phi \cup \neg\Delta \notin \Gamma_{\Sigma}^{\mathcal{G}}$, then $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_{\beta}} \cup \Delta{\downarrow_{\beta}}$ holds.

Proof: Suppose $\Phi \cup \neg\Delta \notin \Gamma_{\Sigma}^{\mathcal{G}}$. By definition, $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_{\beta}} \cup \neg\neg\Delta{\downarrow_{\beta}}$ holds.

Lemma: Consequence of $\mathcal{G}(Inv^\neg)$

Let $\mathcal{G}$ be a sequent calculus such that $\mathcal{G}(Inv^\neg)$ is admissible. For any finite sets Φ and Δ of sentences, if $\Phi \cup \neg\Delta \notin \Gamma_\Sigma^\mathcal{G}$, then $\Vdash_\mathcal{G} \neg\Phi{\downarrow_\beta} \cup \Delta{\downarrow_\beta}$ holds.

Proof: Suppose $\Phi \cup \neg\Delta \notin \Gamma_\Sigma^\mathcal{G}$. By definition, $\Vdash_\mathcal{G} \neg\Phi{\downarrow_\beta} \cup \neg\neg\Delta{\downarrow_\beta}$ holds. Applying $\mathcal{G}(Inv^\neg)$ to each member of $\Delta{\downarrow_\beta}$, we have $\Vdash_\mathcal{G} \neg\Phi{\downarrow_\beta} \cup \Delta{\downarrow_\beta}$.

Thm.: Sufficient Conditions for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Let $\mathcal{G}$ be a sequent calculus.

Thm.: Sufficient Conditions for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Let $\mathcal{G}$ be a sequent calculus. If the rules $\mathcal{G}(Inv^\neg)$, $\mathcal{G}(\neg)$, $\mathcal{G}(weak)$, $\mathcal{G}(init)$, $\mathcal{G}(\vee_-)$, $\mathcal{G}(\vee_+)$, $\mathcal{G}(\Pi_-^{\mathbf{C}})$ and $\mathcal{G}(\Pi_+^c)$ are admissible in $\mathcal{G}$, then $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$.

Thm.: Sufficient Conditions for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Let $\mathcal{G}$ be a sequent calculus. If the rules $\mathcal{G}(Inv^\neg)$, $\mathcal{G}(\neg)$, $\mathcal{G}(weak)$, $\mathcal{G}(init)$, $\mathcal{G}(\vee_-)$, $\mathcal{G}(\vee_+)$, $\mathcal{G}(\Pi_-^{\mathbf{C}})$ and $\mathcal{G}(\Pi_+^c)$ are admissible in $\mathcal{G}$, then $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$.

Proof: We prove $\Gamma_\Sigma^{\mathcal{G}}$ is closed under subsets and satisfies ∇_c, $\nabla_\neg$, $\nabla_\vee$, $\nabla_\wedge$ and ∇_β. The remaining conditions are proven analogously.

Thm.: Sufficient Conditions for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Let $\mathcal{G}$ be a sequent calculus. If the rules $\mathcal{G}(Inv^\neg)$, $\mathcal{G}(\neg)$, $\mathcal{G}(weak)$, $\mathcal{G}(init)$, $\mathcal{G}(\vee_-)$, $\mathcal{G}(\vee_+)$, $\mathcal{G}(\Pi_-^{\mathbf{C}})$ and $\mathcal{G}(\Pi_+^c)$ are admissible in $\mathcal{G}$, then $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$.

Proof: We prove $\Gamma_\Sigma^{\mathcal{G}}$ is closed under subsets and satisfies ∇_c, $\nabla_\neg$, $\nabla_\vee$, $\nabla_\wedge$ and ∇_β. The remaining conditions are proven analogously.

- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, If $\Phi_0 \subseteq \Phi$ and $\Phi_0 \notin \Gamma_\Sigma^{\mathcal{G}}$, then $\Vdash_{\mathcal{G}} \neg\Phi_0{\downarrow}_\beta$ and so $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow}_\beta$ by admissibility of $\mathcal{G}(weak)$. Hence $\Gamma_\Sigma^{\mathcal{G}}$ is closed under subsets.

Thm.: Sufficient Conditions for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Let $\mathcal{G}$ be a sequent calculus. If the rules $\mathcal{G}(Inv^\neg)$, $\mathcal{G}(\neg)$, $\mathcal{G}(weak)$, $\mathcal{G}(init)$, $\mathcal{G}(\vee_-)$, $\mathcal{G}(\vee_+)$, $\mathcal{G}(\Pi_-^{\mathbf{C}})$ and $\mathcal{G}(\Pi_+^c)$ are admissible in $\mathcal{G}$, then $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$.

Proof: We prove $\Gamma_\Sigma^{\mathcal{G}}$ is closed under subsets and satisfies ∇_c, $\nabla_\neg$, $\nabla_\vee$, $\nabla_\wedge$ and ∇_β. The remaining conditions are proven analogously.

- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, If $\Phi_0 \subseteq \Phi$ and $\Phi_0 \notin \Gamma_\Sigma^{\mathcal{G}}$, then $\Vdash_{\mathcal{G}} \neg\Phi_0{\downarrow_\beta}$ and so $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta}$ by admissibility of $\mathcal{G}(weak)$. Hence $\Gamma_\Sigma^{\mathcal{G}}$ is closed under subsets.
- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$ and $\mathbf{A}, \neg\mathbf{A} \in \Phi$ where $\mathbf{A}$ is atomic. By admissibility of $\mathcal{G}(init)$, $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta} * \mathbf{A}{\downarrow_\beta}$ since $\neg\mathbf{A}{\downarrow_\beta} \in \neg\Phi{\downarrow_\beta}$. By admissibility of $\mathcal{G}(\neg)$, $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta}$ since $\neg\neg\mathbf{A}{\downarrow_\beta} \in \neg\Phi{\downarrow_\beta}$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus ∇_c holds.

Thm.: Sufficient Condition for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Proof (contd.):

Thm.: Sufficient Condition for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Proof (contd.):

- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A} \notin \Gamma_\Sigma^{\mathcal{G}}$. Hence $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta} * \neg\mathbf{A}{\downarrow_\beta}$ and so $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta} * \neg\neg\neg\mathbf{A}{\downarrow_\beta}$ by admissibility of $\mathcal{G}(\neg)$. Since $\neg\neg\mathbf{A} \in \Phi$, we know $\neg\Phi{\downarrow_\beta}$ is equal to $\neg\Phi{\downarrow_\beta} * \neg\neg\neg\mathbf{A}{\downarrow_\beta}$. Hence $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta}$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus $\nabla_\neg$ holds.

Thm.: Sufficient Condition for $\Gamma_\Sigma^\mathcal{G} \in \mathfrak{Acc}_\beta$

Proof (contd.):

- Suppose $\Phi \in \Gamma_\Sigma^\mathcal{G}$, $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A} \notin \Gamma_\Sigma^\mathcal{G}$. Hence $\Vdash_\mathcal{G} \neg\Phi\downarrow_\beta * \neg\mathbf{A}\downarrow_\beta$ and so $\Vdash_\mathcal{G} \neg\Phi\downarrow_\beta * \neg\neg\neg\mathbf{A}\downarrow_\beta$ by admissibility of $\mathcal{G}(\neg)$. Since $\neg\neg\mathbf{A} \in \Phi$, we know $\neg\Phi\downarrow_\beta$ is equal to $\neg\Phi\downarrow_\beta * \neg\neg\neg\mathbf{A}\downarrow_\beta$. Hence $\Vdash_\mathcal{G} \neg\Phi\downarrow_\beta$, contradicting $\Phi \in \Gamma_\Sigma^\mathcal{G}$. Thus $\nabla_\neg$ holds.
- Suppose $\Phi \in \Gamma_\Sigma^\mathcal{G}$, $(\mathbf{A} \vee \mathbf{B}) \in \Phi$, $\Phi * \mathbf{A} \notin \Gamma_\Sigma^\mathcal{G}$ and $\Phi * \mathbf{B} \notin \Gamma_\Sigma^\mathcal{G}$. Hence $\Vdash_\mathcal{G} \neg\Phi\downarrow_\beta * \neg\mathbf{A}\downarrow_\beta$ and $\Vdash_\mathcal{G} \neg\Phi\downarrow_\beta * \neg\mathbf{B}\downarrow_\beta$. Applying $\mathcal{G}(\vee_-)$, we have $\Vdash_\mathcal{G} \neg\Phi\downarrow_\beta$ since $\neg(\mathbf{A} \vee \mathbf{B})\downarrow_\beta \in \neg\Phi\downarrow_\beta$, contradicting $\Phi \in \Gamma_\Sigma^\mathcal{G}$. Thus $\nabla_\vee$ holds.

Thm.: Sufficient Condition for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Proof (contd.):

- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A} \notin \Gamma_\Sigma^{\mathcal{G}}$. Hence $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta} * \neg\mathbf{A}{\downarrow_\beta}$ and so $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta} * \neg\neg\neg\mathbf{A}{\downarrow_\beta}$ by admissibility of $\mathcal{G}(\neg)$. Since $\neg\neg\mathbf{A} \in \Phi$, we know $\neg\Phi{\downarrow_\beta}$ is equal to $\neg\Phi{\downarrow_\beta} * \neg\neg\neg\mathbf{A}{\downarrow_\beta}$. Hence $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta}$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus $\nabla_\neg$ holds.
- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $(\mathbf{A} \vee \mathbf{B}) \in \Phi$, $\Phi * \mathbf{A} \notin \Gamma_\Sigma^{\mathcal{G}}$ and $\Phi * \mathbf{B} \notin \Gamma_\Sigma^{\mathcal{G}}$. Hence $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta} * \neg\mathbf{A}{\downarrow_\beta}$ and $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta} * \neg\mathbf{B}{\downarrow_\beta}$. Applying $\mathcal{G}(\vee_-)$, we have $\Vdash_{\mathcal{G}} \neg\Phi{\downarrow_\beta}$ since $\neg(\mathbf{A} \vee \mathbf{B}){\downarrow_\beta} \in \neg\Phi{\downarrow_\beta}$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus $\nabla_\vee$ holds.
- By a similar argument, admissibility of $\mathcal{G}(\Pi_-^{\mathbf{C}})$ implies $\nabla_\forall$.

Thm.: Sufficient Condition for $\Gamma_{\Sigma}^{\mathcal{G}} \in \mathfrak{Acc}_{\beta}$

Proof (contd.):

Thm.: Sufficient Condition for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Proof (contd.):

- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \notin \Gamma_\Sigma^{\mathcal{G}}$. By Lemma 'Consequence of $\mathcal{G}(Inv^{\neg})$', $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta\!*\mathbf{A}\!\downarrow_\beta\!*\mathbf{B}\!\downarrow_\beta$. Applying $\mathcal{G}(\vee_+)$, we have $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta\!*(\mathbf{A} \vee \mathbf{B})\!\downarrow_\beta$. Applying $\mathcal{G}(\neg)$, we have $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta$ since $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus $\nabla_\wedge$ holds.

Thm.: Sufficient Condition for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Proof (contd.):

- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \notin \Gamma_\Sigma^{\mathcal{G}}$. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta\!*\mathbf{A}\!\downarrow_\beta\!*\mathbf{B}\!\downarrow_\beta$. Applying $\mathcal{G}(\lor_+)$, we have $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta\!*(\mathbf{A} \lor \mathbf{B})\!\downarrow_\beta$. Applying $\mathcal{G}(\neg)$, we have $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta$ since $\neg(\mathbf{A} \lor \mathbf{B}) \in \Phi$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus $\nabla_\land$ holds.
- By a similar argument, admissibility of $\mathcal{G}(\Pi_+^c)$, $\mathcal{G}(Inv^\neg)$ and $\mathcal{G}(\neg)$ imply $\nabla_\exists$.

Thm.: Sufficient Condition for $\Gamma_\Sigma^{\mathcal{G}} \in \mathfrak{Acc}_\beta$

Proof (contd.):

- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$ and $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \notin \Gamma_\Sigma^{\mathcal{G}}$. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta\!*\mathbf{A}\!\downarrow_\beta\!*\mathbf{B}\!\downarrow_\beta$. Applying $\mathcal{G}(\vee_+)$, we have $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta\!*(\mathbf{A} \vee \mathbf{B})\!\downarrow_\beta$. Applying $\mathcal{G}(\neg)$, we have $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta$ since $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus $\nabla_\wedge$ holds.
- By a similar argument, admissibility of $\mathcal{G}(\Pi_+^c)$, $\mathcal{G}(Inv^\neg)$ and $\mathcal{G}(\neg)$ imply $\nabla_\exists$.
- Suppose $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $\mathbf{A} \in \Phi$, $\mathbf{A} =_\beta \mathbf{B}$ and $\Phi * \mathbf{B} \notin \Gamma_\Sigma^{\mathcal{G}}$. Hence $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta\!*\neg\mathbf{B}\!\downarrow_\beta$, contradicting $\mathbf{A}\!\downarrow_\beta \in \Phi\!\downarrow_\beta$ and $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$. Thus ∇_β holds.

Thm.: Saturation and Cut

Let $\mathcal{G}$ be a sequent calculus.

Thm.: Saturation and Cut

Let $\mathcal{G}$ be a sequent calculus.

1. If $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$, then $\Gamma_\Sigma^{\mathcal{G}}$ is saturated.

Thm.: Saturation and Cut

Let $\mathcal{G}$ be a sequent calculus.

1. If $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$, then $\Gamma_{\Sigma}^{\mathcal{G}}$ is saturated.
2. If $\mathcal{G}(\neg)$ and $\mathcal{G}(Inv^{\neg})$ are admissible in $\mathcal{G}$ and $\Gamma_{\Sigma}^{\mathcal{G}}$ is saturated, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Thm.: Saturation and Cut

Let $\mathcal{G}$ be a sequent calculus.

1. If $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$, then $\Gamma_\Sigma^{\mathcal{G}}$ is saturated.
2. If $\mathcal{G}(\neg)$ and $\mathcal{G}(Inv^\neg)$ are admissible in $\mathcal{G}$ and $\Gamma_\Sigma^{\mathcal{G}}$ is saturated, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\mathcal{G}(cut)$ is admissible, $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$, $\mathbf{A} \in \mathit{cwff}_o(\Sigma)$, $\Phi * \mathbf{A} \notin \Gamma_\Sigma^{\mathcal{G}}$ and $\Phi * \neg\mathbf{A} \notin \Gamma_\Sigma^{\mathcal{G}}$. Hence $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta * \neg\mathbf{A}\!\downarrow_\beta$ and $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta * \neg\neg\mathbf{A}\!\downarrow_\beta$. Using $\mathcal{G}(cut)$, we have $\Vdash_{\mathcal{G}} \neg\Phi\!\downarrow_\beta$, contradicting $\Phi \in \Gamma_\Sigma^{\mathcal{G}}$.

Thm.: Saturation and Cut

Let $\mathcal{G}$ be a sequent calculus.

1. If $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$, then $\Gamma_\Sigma^\mathcal{G}$ is saturated.
2. If $\mathcal{G}(\neg)$ and $\mathcal{G}(Inv^\neg)$ are admissible in $\mathcal{G}$ and $\Gamma_\Sigma^\mathcal{G}$ is saturated, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\mathcal{G}(cut)$ is admissible, $\Phi \in \Gamma_\Sigma^\mathcal{G}$, $\mathbf{A} \in \mathit{cwff}_o(\Sigma)$, $\Phi * \mathbf{A} \notin \Gamma_\Sigma^\mathcal{G}$ and $\Phi * \neg\mathbf{A} \notin \Gamma_\Sigma^\mathcal{G}$. Hence $\Vdash_\mathcal{G} \neg\Phi\!\downarrow_\beta * \neg\mathbf{A}\!\downarrow_\beta$ and $\Vdash_\mathcal{G} \neg\Phi\!\downarrow_\beta * \neg\neg\mathbf{A}\!\downarrow_\beta$. Using $\mathcal{G}(cut)$, we have $\Vdash_\mathcal{G} \neg\Phi\!\downarrow_\beta$, contradicting $\Phi \in \Gamma_\Sigma^\mathcal{G}$.

Suppose $\Gamma_\Sigma^\mathcal{G}$ is saturated, $\Vdash_\mathcal{G} \Delta * \mathbf{C}$ and $\Vdash_\mathcal{G} \Delta * \neg\mathbf{C}$ hold but $\Vdash_\mathcal{G} \Delta$ does not. Applying $\mathcal{G}(\neg)$ to every member of Δ and to $\mathbf{C}$ we have $\Vdash_\mathcal{G} \neg\neg\Delta * \neg\neg\mathbf{C}$ and $\Vdash_\mathcal{G} \neg\neg\Delta * \neg\mathbf{C}$. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', we know $\neg\Delta \in \Gamma_\Sigma^\mathcal{G}$. By saturation, we must have $\neg\Delta * \mathbf{C} \in \Gamma_\Sigma^\mathcal{G}$ or $\neg\Delta * \neg\mathbf{C} \in \Gamma_\Sigma^\mathcal{G}$. The first case contradicts $\Vdash_\mathcal{G} \neg\neg\Delta * \neg\mathbf{C}$ while the second case contradicts $\Vdash_\mathcal{G} \neg\neg\Delta * \neg\neg\mathbf{C}$.

Def.: Saturated Extension

Since saturation is equivalent to admissibility of cut, we need (are interested in) weaker conditions than saturation. A natural condition to consider is the existence of saturated extensions.

Def.: Saturated Extension

Since saturation is equivalent to admissibility of cut, we need (are interested in) weaker conditions than saturation. A natural condition to consider is the existence of saturated extensions.

Def. (Saturated Extension): Let $\Gamma_\Sigma, \Gamma'_\Sigma \in \mathfrak{Acc}_*$ be abstract consistency classes. We say Γ'_Σ is an extension of Γ_Σ if $\Phi \in \Gamma'_\Sigma$ for every sufficiently Σ-pure $\Phi \in \Gamma_\Sigma$. We say Γ'_Σ is a saturated extension of Γ_Σ if Γ'_Σ is saturated and an extension of Γ_Σ.

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Example:

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Example:
Let $a_o, b_o, q_{o\to o} \in \Sigma$ and $\Phi := \{a, b, (qa), \neg(qb)\}$. We construct an abstract consistency class Γ_Σ from Φ by first building the closure Φ' of Φ under relation $=_\beta$ and then taking the power set of Φ'. It is easy to check that this Γ_Σ is in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$.

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Example:
Let $a_o, b_o, q_{o\to o} \in \Sigma$ and $\Phi := \{a, b, (qa), \neg(qb)\}$. We construct an abstract consistency class Γ_Σ from Φ by first building the closure Φ' of Φ under relation $=_\beta$ and then taking the power set of Φ'. It is easy to check that this Γ_Σ is in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Suppose we have a saturated extension Γ'_Σ of Γ_Σ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$.

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Example:

Let $a_o, b_o, q_{o\to o} \in \Sigma$ and $\Phi := \{a, b, (qa), \neg(qb)\}$. We construct an abstract consistency class Γ_Σ from Φ by first building the closure Φ' of Φ under relation $=_\beta$ and then taking the power set of Φ'. It is easy to check that this Γ_Σ is in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Suppose we have a saturated extension Γ'_Σ of Γ_Σ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Then $\Phi \in \Gamma'_\Sigma$ since Φ is finite (hence sufficiently pure).

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Example:
Let $a_o, b_o, q_{o\to o} \in \Sigma$ and $\Phi := \{a, b, (qa), \neg(qb)\}$. We construct an abstract consistency class Γ_Σ from Φ by first building the closure Φ' of Φ under relation $=_\beta$ and then taking the power set of Φ'. It is easy to check that this Γ_Σ is in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Suppose we have a saturated extension Γ'_Σ of Γ_Σ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Then $\Phi \in \Gamma'_\Sigma$ since Φ is finite (hence sufficiently pure). By saturation, $\Phi * (a \doteq^o b) \in \Gamma'_\Sigma$ or $\Phi * \neg(a \doteq^o b) \in \Gamma'_\Sigma$.

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Example:
Let $a_o, b_o, q_{o\to o} \in \Sigma$ and $\Phi := \{a, b, (qa), \neg(qb)\}$. We construct an abstract consistency class Γ_Σ from Φ by first building the closure Φ' of Φ under relation $=_\beta$ and then taking the power set of Φ'. It is easy to check that this Γ_Σ is in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Suppose we have a saturated extension Γ'_Σ of Γ_Σ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Then $\Phi \in \Gamma'_\Sigma$ since Φ is finite (hence sufficiently pure). By saturation, $\Phi * (a \doteq^o b) \in \Gamma'_\Sigma$ or $\Phi * \neg(a \doteq^o b) \in \Gamma'_\Sigma$. In the first case, applying $\nabla_\forall$ with the constant q, ∇_β, $\nabla_\vee$ and ∇_c contradicts $(qa), \neg(qb) \in \Phi$.

Ex.: ACC without Saturated Extension

There exist abstract consistency classes Γ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$ which have no saturated extension.

Example:
Let $a_o, b_o, q_{o \to o} \in \Sigma$ and $\Phi := \{a, b, (qa), \neg(qb)\}$. We construct an abstract consistency class Γ_Σ from Φ by first building the closure Φ' of Φ under relation $=_\beta$ and then taking the power set of Φ'. It is easy to check that this Γ_Σ is in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Suppose we have a saturated extension Γ'_Σ of Γ_Σ in $\mathfrak{Acc}_{\beta\mathfrak{fb}}$. Then $\Phi \in \Gamma'_\Sigma$ since Φ is finite (hence sufficiently pure). By saturation, $\Phi * (a \doteq^o b) \in \Gamma'_\Sigma$ or $\Phi * \neg(a \doteq^o b) \in \Gamma'_\Sigma$. In the first case, applying $\nabla_\forall$ with the constant q, ∇_β, $\nabla_\vee$ and ∇_c contradicts $(qa), \neg(qb) \in \Phi$. In the second case, $\nabla_\mathfrak{b}$ and ∇_c contradict $a, b \in \Phi$.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^{\mathcal{G}}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^{\mathcal{G}}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\Gamma'_\Sigma \in \mathfrak{Acc}_*$ is a saturated extension of $\Gamma_\Sigma^{\mathcal{G}}$.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^{\mathcal{G}}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\Gamma'_\Sigma \in \mathfrak{Acc}_*$ is a saturated extension of $\Gamma_\Sigma^{\mathcal{G}}$. Assume $\Vdash_{\mathcal{G}} \Delta * \mathbf{C}$ and $\Vdash_{\mathcal{G}} \Delta * \neg\mathbf{C}$ hold and $\Vdash_{\mathcal{G}} \Delta$ does not.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^{\mathcal{G}}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\Gamma'_\Sigma \in \mathfrak{Acc}_*$ is a saturated extension of $\Gamma_\Sigma^{\mathcal{G}}$. Assume $\Vdash_{\mathcal{G}} \Delta * \mathbf{C}$ and $\Vdash_{\mathcal{G}} \Delta * \neg\mathbf{C}$ hold and $\Vdash_{\mathcal{G}} \Delta$ does not. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', we know $\neg\Delta \in \Gamma_\Sigma^{\mathcal{G}}$.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^{\mathcal{G}}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\Gamma'_\Sigma \in \mathfrak{Acc}_*$ is a saturated extension of $\Gamma_\Sigma^{\mathcal{G}}$. Assume $\Vdash_{\mathcal{G}} \Delta * \mathbf{C}$ and $\Vdash_{\mathcal{G}} \Delta * \neg\mathbf{C}$ hold and $\Vdash_{\mathcal{G}} \Delta$ does not. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', we know $\neg\Delta \in \Gamma_\Sigma^{\mathcal{G}}$. Since $\neg\Delta$ is finite (hence sufficiently pure), $\neg\Delta \in \Gamma'_\Sigma$.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^\mathcal{G}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\Gamma'_\Sigma \in \mathfrak{Acc}_*$ is a saturated extension of $\Gamma_\Sigma^\mathcal{G}$. Assume $\Vdash_\mathcal{G} \Delta * \mathbf{C}$ and $\Vdash_\mathcal{G} \Delta * \neg\mathbf{C}$ hold and $\Vdash_\mathcal{G} \Delta$ does not. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', we know $\neg\Delta \in \Gamma_\Sigma^\mathcal{G}$. Since $\neg\Delta$ is finite (hence sufficiently pure), $\neg\Delta \in \Gamma'_\Sigma$. By the model existence theorem for saturated abstract consistency classes, there is a model $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ such that $\mathcal{M} \models \neg\Delta$.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^{\mathcal{G}}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\Gamma'_\Sigma \in \mathfrak{Acc}_*$ is a saturated extension of $\Gamma_\Sigma^{\mathcal{G}}$. Assume $\Vdash_{\mathcal{G}} \Delta * \mathbf{C}$ and $\Vdash_{\mathcal{G}} \Delta * \neg\mathbf{C}$ hold and $\Vdash_{\mathcal{G}} \Delta$ does not. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', we know $\neg\Delta \in \Gamma_\Sigma^{\mathcal{G}}$. Since $\neg\Delta$ is finite (hence sufficiently pure), $\neg\Delta \in \Gamma'_\Sigma$. By the model existence theorem for saturated abstract consistency classes, there is a model $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ such that $\mathcal{M} \models \neg\Delta$. By soundness of $\mathcal{G}$, we know both $\Delta * \mathbf{C}$ and $\Delta * \neg\mathbf{C}$ must be valid in $\mathcal{M}$.

Existence of Saturated Extensions and Cut

Existence of any saturated extension of a sound sequent calculus $\mathcal{G}$ implies admissibility of cut:

Let $\mathcal{G}$ be a sequent calculus which is sound for $\mathfrak{M}_*(\Sigma)$. If $\Gamma_\Sigma^\mathcal{G}$ has a saturated extension $\Gamma'_\Sigma \in \mathfrak{Acc}_*$, then $\mathcal{G}(cut)$ is admissible in $\mathcal{G}$.

Proof: Suppose $\Gamma'_\Sigma \in \mathfrak{Acc}_*$ is a saturated extension of $\Gamma_\Sigma^\mathcal{G}$. Assume $\Vdash_\mathcal{G} \Delta * \mathbf{C}$ and $\Vdash_\mathcal{G} \Delta * \neg\mathbf{C}$ hold and $\Vdash_\mathcal{G} \Delta$ does not. By Lemma 'Consequence of $\mathcal{G}(Inv^\neg)$', we know $\neg\Delta \in \Gamma_\Sigma^\mathcal{G}$. Since $\neg\Delta$ is finite (hence sufficiently pure), $\neg\Delta \in \Gamma'_\Sigma$. By the model existence theorem for saturated abstract consistency classes, there is a model $\mathcal{M} \in \mathfrak{M}_*(\Sigma)$ such that $\mathcal{M} \models \neg\Delta$. By soundness of $\mathcal{G}$, we know both $\Delta * \mathbf{C}$ and $\Delta * \neg\mathbf{C}$ must be valid in $\mathcal{M}$. Since $\mathcal{M} \models \neg\Delta$, we must have $\mathcal{M} \models \mathbf{C}$ and $\mathcal{M} \models \neg\mathbf{C}$, a contradiction.