

Automated Reasoning

Georg Moser



Institute of Computer Science @ UIBK

Winter 2013

Summary Last Lecture

Definition sequent calculus

Theorem (Normalisation and Strong Normalisation)

let Π be a proof in minimal logic

- **1** \exists a reduction sequence $\Pi = \Pi_1, \ldots, \Pi_n$
- **2** \exists computable upper bound *n* on the maximal length of any reduction sequence

Corollary

if T_0 is complete and T_1 , T_2 are satisfiable extensions of T_0 , then $T_1 \cup T_2$ is satisfiable

Theorem

let \mathcal{G} be a set of universal sentences (of \mathcal{L}) without =, then the following is equivalent

- **1** *G* is satisfiable
- **2** *G* has a Herbrand model
- **3** \forall finite $\mathcal{G}_0 \subseteq Gr(\mathcal{G})$, \mathcal{G}_0 has a Herbrand model

Corollary

 $\exists x_1 \cdots \exists x_n G(x_1, \dots, x_n)$ is valid iff there are ground terms t_1^k, \dots, t_n^k , $k \in \mathbb{N}$ and the following is valid: $G(t_1^1, \dots, t_n^1) \lor \cdots \lor G(t_1^k, \dots, t_n^k)$

Theorem

 \forall formula F, \exists formula G not containing individual, nor function constants, nor = such that F \approx G

Outline of the Lecture

Propositional Logic

short reminder of propositional logic, soundness and completeness theorem, natural deduction, propositional resolution

First Order Logic

introduction, syntax, semantics, undecidability of first-order, Löwenheim-Skolem, compactness, model existence theorem, natural deduction, completeness, sequent calculus, normalisation

Properties of First Order Logic

Craig's Interpolation Theorem, Robinson's Joint Consistency Theorem, Herbrand's Theorem

Limits and Extensions of First Order Logic

Intuitionistic Logic, Curry-Howard Isomorphism, Limits, Second-Order Logic

Outline of the Lecture

Propositional Logic

short reminder of propositional logic, soundness and completeness theorem, natural deduction, propositional resolution

First Order Logic

introduction, syntax, semantics, undecidability of first-order, Löwenheim-Skolem, compactness, model existence theorem, natural deduction, completeness, sequent calculus, normalisation

Properties of First Order Logic

Craig's Interpolation Theorem, Robinson's Joint Consistency Theorem, Herbrand's Theorem

Limits and Extensions of First Order Logic

Intuitionistic Logic, Curry-Howard Isomorphism, Limits, Second-Order Logic

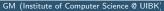
Definition

according to naive set theory, any definable collection is a set; this is not a good idea



Definition

according to naive set theory, any definable collection is a set; this is not a good idea



Definition

according to naive set theory, any definable collection is a set; this is not a good idea

Proof.

1 let $R := \{x \mid x \notin x\}$



Definition

according to naive set theory, any definable collection is a set; this is not a good idea

- **1** let $R := \{x \mid x \notin x\}$
- **2** as " $x \notin x$ " is a definition (= a predicate) this should be set

Definition

according to naive set theory, any definable collection is a set; this is not a good idea

- 1 let $R := \{x \mid x \notin x\}$
- **2** as " $x \notin x$ " is a definition (= a predicate) this should be set
- **3** so either $R \in R$, or $R \notin R$, but

$$R \in R \rightarrow R \notin R$$
 $R \notin R \rightarrow R \in R$

Definition

according to naive set theory, any definable collection is a set; this is not a good idea

Proof.

- 1 let $R := \{x \mid x \notin x\}$
- **2** as " $x \notin x$ " is a definition (= a predicate) this should be set
- **3** so either $R \in R$, or $R \notin R$, but

$$R \in R \rightarrow R \notin R$$
 $R \notin R \rightarrow R \in R$

4 hence $R \in R \leftrightarrow R \notin R$, which is a contradiction

Definition

according to naive set theory, any definable collection is a set; this is not a good idea

- 1 let $R := \{x \mid x \notin x\}$
- **2** as " $x \notin x$ " is a definition (= a predicate) this should be set
- **3** so either $R \in R$, or $R \notin R$, but

$$R \in R \rightarrow R \notin R$$
 $R \notin R \rightarrow R \in R$

- 4 hence $R \in R \leftrightarrow R \notin R$, which is a contradiction
- 5 thus naive set theory is inconsistent

Oops, what to do?

Brouwer's Way Out (1742)

Change Mathematics!





Oops, what to do?

Brouwer's Way Out (1742)

Change Mathematics!



Definition

• intuitionistic logic is a restriction of classical logic, where certain formulas are no longer derivable

Oops, what to do?

Brouwer's Way Out (1742)

Change Mathematics!



Definition

- intuitionistic logic is a restriction of classical logic, where certain formulas are no longer derivable
- for example $A \lor \neg A$ is no longer valid

Brouwer's Way Out (1742)

Change Mathematics!

Definition

- intuitionistic logic is a restriction of classical logic, where certain formulas are no longer derivable
- for example $A \lor \neg A$ is no longer valid
- its interpretation in intuitionistic logic is:

there is an argument for A or there is a argument for $\neg A$ (= from the assumption A we can prove a contradiction)



Theorem

 \exists solutions of the equation $x^y = z$ with x and y irrational and z rational



Theorem

 \exists solutions of the equation $x^y = z$ with x and y irrational and z rational

Proof.

1 $\sqrt{2}$ is an irrational number

Theorem

 \exists solutions of the equation $x^y = z$ with x and y irrational and z rational

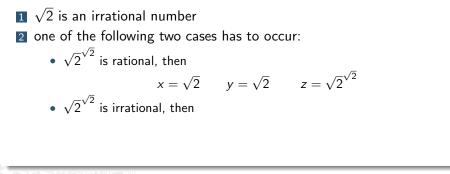
Proof.

- 1 $\sqrt{2}$ is an irrational number
- 2 one of the following two cases has to occur:
 - $\sqrt{2}^{\sqrt{2}}$ is rational, then

• $\sqrt{2}^{\sqrt{2}}$ is irrational, then

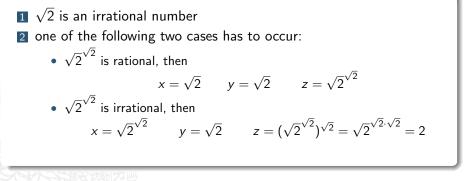
Theorem

 \exists solutions of the equation $x^y = z$ with x and y irrational and z rational



Theorem

 \exists solutions of the equation $x^y = z$ with x and y irrational and z rational



Theorem

 \exists solutions of the equation $x^y = z$ with x and y irrational and z rational

1
$$\sqrt{2}$$
 is an irrational number
2 one of the following two cases has to occur:
• $\sqrt{2}^{\sqrt{2}}$ is rational, then
 $x = \sqrt{2}$ $y = \sqrt{2}$ $z = \sqrt{2}^{\sqrt{2}}$
• $\sqrt{2}^{\sqrt{2}}$ is irrational, then
 $x = \sqrt{2}^{\sqrt{2}}$ $y = \sqrt{2}$ $z = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2} \cdot \sqrt{2}} = 2$

Theorem

 \exists solutions of the equation $x^y = z$ with x and y irrational and z rational

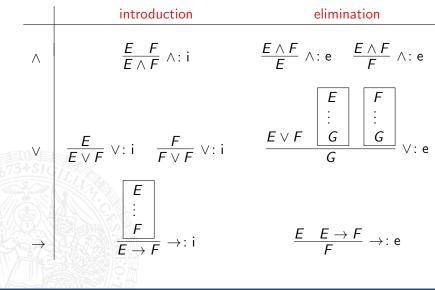
Proof.

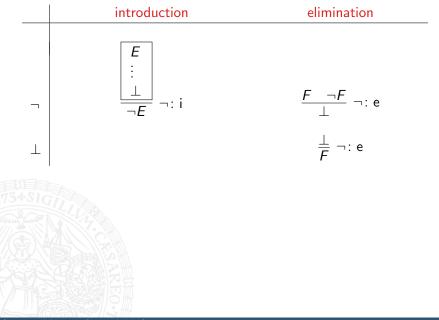
1
$$\sqrt{2}$$
 is an irrational number
2 one of the following two cases has to occur:
• $\sqrt{2}^{\sqrt{2}}$ is rational, then
 $x = \sqrt{2}$ $y = \sqrt{2}$ $z = \sqrt{2}^{\sqrt{2}}$
• $\sqrt{2}^{\sqrt{2}}$ is irrational, then
 $x = \sqrt{2}^{\sqrt{2}}$ $y = \sqrt{2}$ $z = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2} \cdot \sqrt{2}} = 2$

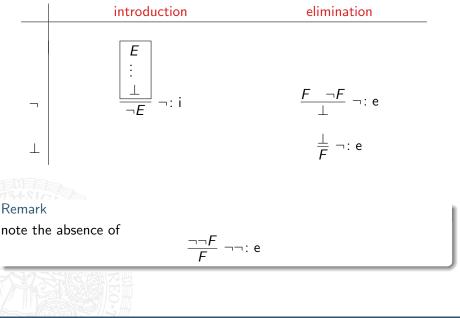
prototypical example of a non-constructive proof

GM (Institute of Computer Science @ UIBK)

Intuitionistic Logic







GM (Institute of Computer Science @ UIBK)

an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 \forall sequents $\Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$



an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 \forall sequents $\Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$

an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 $\forall \text{ sequents } \Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$

Definition (Brower, Heyting, Kolmogorow (Kreisel) Interpretation)

• an argument for $E \wedge F$ is an argument for E and F

an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 \forall sequents $\Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$

- an argument for $E \wedge F$ is an argument for E and F
- an argument for $E \lor F$ is an argument of E or F

an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 \forall sequents $\Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$

- an argument for $E \wedge F$ is an argument for E and F
- an argument for $E \lor F$ is an argument of E or F
- an argument for $E \rightarrow F$ is a transformation of an argument for E into an argument for F

an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 \forall sequents $\Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$

- an argument for $E \wedge F$ is an argument for E and F
- an argument for $E \lor F$ is an argument of E or F
- an argument for $E \rightarrow F$ is a transformation of an argument for E into an argument for F
- $\neg E$ is interpreted as $E \rightarrow \perp$

an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 \forall sequents $\Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$

- an argument for $E \wedge F$ is an argument for E and F
- an argument for $E \lor F$ is an argument of E or F
- an argument for $E \rightarrow F$ is a transformation of an argument for E into an argument for F
- $\neg E$ is interpreted as $E \rightarrow \perp$
- no argument for \perp can exist

an equivalent formalisation of intutitionistic logic is given by sequent calculus with the following restriction:

 \forall sequents $\Gamma \Rightarrow \Delta \colon |\Delta| \leqslant 1$

Definition (Brower, Heyting, Kolmogorow (Kreisel) Interpretation)

- an argument for $E \wedge F$ is an argument for E and F
- an argument for $E \lor F$ is an argument of E or F
- an argument for $E \rightarrow F$ is a transformation of an argument for E into an argument for F
- $\neg E$ is interpreted as $E \rightarrow \perp$
- no argument for \perp can exist

the formal definition needs Kripke models

Kripke Models

Definition

 a frame *F* is a pair (*W*, ≤), where *W* denotes a nonempty set of worlds and ≤ denotes a preorder on *W*

Kripke Models

Definition

- a frame *F* is a pair (*W*, ≤), where *W* denotes a nonempty set of worlds and ≤ denotes a preorder on *W*
- a Kripke model on a frame $F = (W, \leq)$ is a triple

$$\mathcal{K} = (W, \leqslant, (\mathcal{A}_p)_{p \in W})$$

such that for all p:

$$\mathcal{A}_{\rho} = (A_{\rho}, a_{\rho})$$

$$\mathcal{A}_{\rho} \text{ is a non-empty set (the domain in world ρ)}
$$\mathcal{A}_{\rho} \text{ is a mapping that associates predicate constants to domains}$$$$

Kripke Models

Definition

- a frame *F* is a pair (*W*, ≤), where *W* denotes a nonempty set of worlds and ≤ denotes a preorder on *W*
- a Kripke model on a frame $F = (W, \leq)$ is a triple

$$\mathcal{K} = (W, \leqslant, (\mathcal{A}_p)_{p \in W})$$

such that for all p:

- \$\mathcal{A}_p = (\mathcal{A}_p, \mathcal{a}_p)\$
 \$\mathcal{A}_p\$ is a non-empty set (the domain in world \$p\$)
 \$\mathcal{A}_p\$ is a mapping that associates predicate constants to domains
- \forall predicate symbols P, $p, q \in W$, $(a_1, \dots, a_n) \in A_p^n$: $p \leq q, A_p \models P(a_1, \dots, a_n)$ implies $A_q \models P(a_1, \dots, a_n)$

Kripke Models

Definition

- a frame *F* is a pair (*W*, ≤), where *W* denotes a nonempty set of worlds and ≤ denotes a preorder on *W*
- a Kripke model on a frame $F = (W, \leq)$ is a triple

$$\mathcal{K} = (W, \leqslant, (\mathcal{A}_p)_{p \in W})$$

such that for all p:

- \$\mathcal{A}_p = (\mathcal{A}_p, \mathcal{a}_p)\$
 \$\mathcal{A}_p\$ is a non-empty set (the domain in world \$p\$)
 \$\mathcal{A}_p\$ is a mapping that associates predicate constants to domains
- ∀ predicate symbols P, p, q ∈ W, (a₁,..., a_n) ∈ Aⁿ_p:
 p ≤ q, A_p ⊨ P(a₁,..., a_n) implies A_q ⊨ P(a₁,..., a_n)

• we set
$$A = \bigcup_{p \in W} A_p$$

Convention

suppose $F(x_1, \ldots, x_n)$ is formula with free variables x_1, \ldots, x_n ; we write $F(a_1, \ldots, a_n)$ for the "interpretation" of x_i by $a_i \in A$ in F



Convention

suppose $F(x_1, ..., x_n)$ is formula with free variables $x_1, ..., x_n$; we write $F(a_1, ..., a_n)$ for the "interpretation" of x_i by $a_i \in A$ in F

Definition

for a given Kripke model $\mathcal{K} = (W, \leq, (\mathcal{A}_p)_{p \in W})$ the satisfaction relation is defined as follows:

$$\begin{array}{lll} \mathcal{K}, p \Vdash \top & \mathcal{K}, p \nvDash \bot \\ \mathcal{K}, p \Vdash P(a_1, \dots, a_n) & \text{if } \mathcal{A}_p \models P(a_1, \dots, a_n) \\ \mathcal{K}, p \Vdash A \land B & \text{iff } \mathcal{K}, p \Vdash A \text{ and } \mathcal{K}, p \Vdash B \\ \mathcal{K}, p \Vdash A \lor B & \text{iff } \mathcal{K}, p \Vdash A \text{ or } \mathcal{K}, p \Vdash B \\ \mathcal{K}, p \Vdash A \to B & \text{iff for all } q \ge p: \mathcal{K}, q \Vdash A \text{ implies } \mathcal{K}, q \Vdash B \end{array}$$

Convention

suppose $F(x_1, ..., x_n)$ is formula with free variables $x_1, ..., x_n$; we write $F(a_1, ..., a_n)$ for the "interpretation" of x_i by $a_i \in A$ in F

Definition

for a given Kripke model $\mathcal{K} = (W, \leq, (\mathcal{A}_p)_{p \in W})$ the satisfaction relation is defined as follows:

$$\begin{array}{lll} \mathcal{K}, p \Vdash \top & \mathcal{K}, p \nvDash \bot \\ \mathcal{K}, p \Vdash P(a_1, \dots, a_n) & \text{if } \mathcal{A}_p \models P(a_1, \dots, a_n) \\ \mathcal{K}, p \Vdash A \land B & \text{iff } \mathcal{K}, p \Vdash A \text{ and } \mathcal{K}, p \Vdash B \\ \mathcal{K}, p \Vdash A \lor B & \text{iff } \mathcal{K}, p \Vdash A \text{ or } \mathcal{K}, p \Vdash B \\ \mathcal{K}, p \Vdash A \rightarrow B & \text{iff for all } q \ge p: \mathcal{K}, q \Vdash A \text{ implies } \mathcal{K}, q \Vdash B \end{array}$$

a formula *F* is valid in \mathcal{K} if $\mathcal{K}, p \Vdash F$ for all $p \in W$

GM (Institute of Computer Science @ UIBK

Some Transfer Results

Theorem

natural deduction (and the sequent calculus) for intuitionistic logic is sound and complete



Some Transfer Results

Theorem

natural deduction (and the sequent calculus) for intuitionistic logic is sound and complete

Theorem

- natural deduction is strongly normalising
- sequent calculus admits cut-eliminiation

Some Transfer Results

Theorem

natural deduction (and the sequent calculus) for intuitionistic logic is sound and complete

Theorem

- natural deduction is strongly normalising
- sequent calculus admits cut-eliminiation

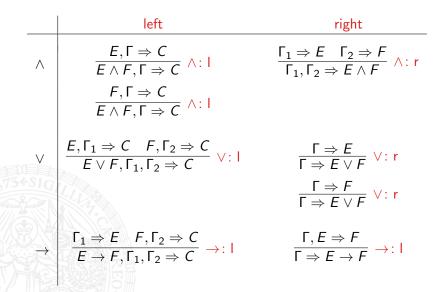
Theorem

Craig's interpolation theoremm holds for intutitionistic logic

"Natural Deduction" for Minimal Logic

	introduction	el	imination
		$A \Rightarrow A$	
\wedge	$\frac{\Gamma \Rightarrow E \Gamma \Rightarrow F}{\Gamma \Rightarrow E \land F}$	$\frac{\Gamma \Rightarrow E \land F}{\Gamma \Rightarrow E}$	$\frac{\Gamma \Rightarrow E \land F}{\Gamma \Rightarrow F}$
V 734	$\frac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E \lor F} \frac{\Gamma \Rightarrow F}{\Gamma \Rightarrow E \lor F}$		$\frac{\Gamma, E \Rightarrow G \Gamma, F \Rightarrow G}{\Gamma \Rightarrow G}$
\rightarrow	$\frac{\Gamma, E \Rightarrow F}{\Gamma \Rightarrow E \to F}$	$\Gamma \Rightarrow E$	$\frac{\Gamma \Rightarrow E \to F}{\Gamma \Rightarrow F}$
	REF		

A Sequent Calculus for Minimal Logic



let $S = (\Gamma \Rightarrow C)$ be a sequent; \exists proof Π of S in natural deduction iff \exists proof Ψ of S in the sequent calculus



let $S = (\Gamma \Rightarrow C)$ be a sequent; \exists proof Π of S in natural deduction iff \exists proof Ψ of S in the sequent calculus

Proof

direction from left to right is shown by induction on the length of $\Pi,$ i.e., on the number of sequents in Π

let $S = (\Gamma \Rightarrow C)$ be a sequent; \exists proof Π of S in natural deduction iff \exists proof Ψ of S in the sequent calculus

Proof

direction from left to right is shown by induction on the length of $\Pi,$ i.e., on the number of sequents in Π

1 the base case is immediate as $\Pi \vdash A \Rightarrow A$ iff $\Psi \vdash A \Rightarrow A$

let $S = (\Gamma \Rightarrow C)$ be a sequent; \exists proof Π of S in natural deduction iff \exists proof Ψ of S in the sequent calculus

Proof

direction from left to right is shown by induction on the length of $\Pi,$ i.e., on the number of sequents in Π

- **1** the base case is immediate as $\Pi \vdash A \Rightarrow A$ iff $\Psi \vdash A \Rightarrow A$
- **2** for the step case, consider the case that Π has the following form:

$$\frac{\prod_{0}}{\Gamma \Rightarrow E \land F}$$
$$\frac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E}$$

by induction hypothesis \exists a sequent calculus proof Ψ_0 of $\Gamma \Rightarrow E \wedge F$

3 the following is a correct proof:

$$\frac{\Psi_0}{\Gamma \Rightarrow E \land F} \xrightarrow{E \Rightarrow E}{E \land F \Rightarrow E} \land : \mathsf{I}$$

$$\frac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E} \land : \mathsf{I}$$



3 the following is a correct proof:

$$\frac{\Psi_0}{\Gamma \Rightarrow E \land F} \quad \frac{E \Rightarrow E}{E \land F \Rightarrow E} \quad \land : \mathsf{I}$$
$$\frac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E} \quad \mathsf{cut}$$

4 all other cases are similar



3 the following is a correct proof:

$$\frac{\Psi_0}{\Gamma \Rightarrow E \land F} \quad \frac{E \Rightarrow E}{E \land F \Rightarrow E} \quad \land: \mathsf{I}$$
$$\frac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E} \quad \mathsf{cut}$$

4 all other cases are similar

the other direction follows by induction on the length of $\boldsymbol{\Psi}$

3 the following is a correct proof:

$$\frac{\Psi_0}{\Gamma \Rightarrow E \land F} \quad \frac{E \Rightarrow E}{E \land F \Rightarrow E} \quad \land: \mathsf{I}$$
$$\frac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E} \quad \mathsf{cut}$$

4 all other cases are similar

the other direction follows by induction on the length of $\boldsymbol{\Psi}$

 Question

 is this really correct?

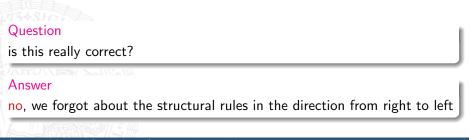
 GM (Institute of Computer Science @ UBK)

3 the following is a correct proof:

$$\frac{\Psi_0}{\Gamma \Rightarrow E \land F} \quad \frac{E \Rightarrow E}{E \land F \Rightarrow E} \quad \land: \mathsf{I}$$
$$\frac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E} \quad \mathsf{cut}$$

4 all other cases are similar

the other direction follows by induction on the length of $\boldsymbol{\Psi}$



"Natural Deduction" Structural Rules

contraction	$\frac{A, A, \Gamma \Rightarrow C}{A, \Gamma \Rightarrow C}$	
weakening	$\frac{\Gamma \Rightarrow C}{A, \Gamma \Rightarrow C}$	$\frac{\Gamma \Rightarrow}{\Gamma \Rightarrow C}$

"Natural Deduction" Structural Rules

contraction	$\frac{A, A, \Gamma \Rightarrow C}{A, \Gamma \Rightarrow C}$
weakening	$\frac{\Gamma \Rightarrow C}{A, \Gamma \Rightarrow C}$

Observations

- note the restriction to one formula in the succedent
- contraction and weakening can also be represented by changed axioms and representation of sequents

Outline of the Lecture

Propositional Logic

short reminder of propositional logic, soundness and completeness theorem, natural deduction, propositional resolution

First Order Logic

introduction, syntax, semantics, undecidability of first-order, Löwenheim-Skolem, compactness, model existence theorem, natural deduction, completeness, sequent calculus, normalisation

Properties of First Order Logic

Craig's Interpolation Theorem, Robinson's Joint Consistency Theorem, Herbrand's Theorem

VILLAN PURCHARM

Limits and Extensions of First Order Logic

Intuitionistic Logic, Curry-Howard Isomorphism, Second-Order Logic

Outline of the Lecture

Propositional Logic

short reminder of propositional logic, soundness and completeness theorem, natural deduction, propositional resolution

First Order Logic

introduction, syntax, semantics, undecidability of first-order, Löwenheim-Skolem, compactness, model existence theorem, natural deduction, completeness, sequent calculus, normalisation

Properties of First Order Logic

Craig's Interpolation Theorem, Robinson's Joint Consistency Theorem, Herbrand's Theorem

VILLAN DAVIE

Limits and Extensions of First Order Logic

Intuitionistic Logic, Curry-Howard Isomorphism, Second-Order Logic

Definition (types and terms)



Definition (types and terms)

- a variable type: α , β , γ , ...
- if σ , τ are types, then $(\sigma \times \tau)$ is a (product) type
- if σ , τ are types, then $(\sigma \rightarrow \tau)$ is a (function) type

Definition (types and terms)

- a variable type: α , β , γ , ...
- if σ , τ are types, then $(\sigma \times \tau)$ is a (product) type
- if σ , τ are types, then $(\sigma \rightarrow \tau)$ is a (function) type
- any (typed) variable $x : \sigma$ is a (typed) term

Definition (types and terms)

- a variable type: α , β , γ , ...
- if σ , τ are types, then $(\sigma \times \tau)$ is a (product) type
- if σ , τ are types, then $(\sigma \rightarrow \tau)$ is a (function) type
- any (typed) variable $x : \sigma$ is a (typed) term
- if $M : \sigma$, $N : \tau$ are terms, then $\langle M, N \rangle : \sigma \times \tau$ is a term
- if $M : \sigma \times \tau$ is a term, then $fst(M) : \sigma$ and $snd(M) : \tau$ are terms

Definition (types and terms)

- a variable type: α , β , γ , ...
- if σ , τ are types, then $(\sigma \times \tau)$ is a (product) type
- if σ , τ are types, then $(\sigma \rightarrow \tau)$ is a (function) type
- any (typed) variable $x : \sigma$ is a (typed) term
- if $M : \sigma$, $N : \tau$ are terms, then $\langle M, N \rangle : \sigma \times \tau$ is a term
- if $M : \sigma \times \tau$ is a term, then $fst(M) : \sigma$ and $snd(M) : \tau$ are terms
- if M : τ is a term, x : σ a variable, then the abstraction (λx^σ.M) : σ → τ is a term
- if $M : \sigma \to \tau$, $N : \sigma$ are terms, then the application $(MN) : \tau$ is a term.

Example

the following are (well-formed, typed) terms

 $\lambda fx.fx: (\sigma \to \tau) \to \sigma \to \tau \qquad \langle \lambda x.x, \lambda y.y \rangle : (\sigma \to \sigma) \times (\tau \to \tau)$

but $\lambda x.xx$ cannot be typed!



Example

the following are (well-formed, typed) terms

 $\lambda f x. f x: (\sigma \to \tau) \to \sigma \to \tau \qquad \langle \lambda x. x, \lambda y. y \rangle: (\sigma \to \sigma) \times (\tau \to \tau)$

but $\lambda x.xx$ cannot be typed!

Definition

the set of free variables of a term is defined as follows

• $FV(x) = \{x\}.$

•
$$FV(\lambda x.M) = FV(M) - \{x\}$$

- $FV(MN) = FV(\langle M, N \rangle) = FV(M) \cup FV(N).$
- FV(fst(M)) = FV(snd(M)) = FV(M).

Example

the following are (well-formed, typed) terms

 $\lambda f x. f x: (\sigma \to \tau) \to \sigma \to \tau \qquad \langle \lambda x. x, \lambda y. y \rangle: (\sigma \to \sigma) \times (\tau \to \tau)$

but $\lambda x.xx$ cannot be typed!

Definition

the set of free variables of a term is defined as follows

• $FV(x) = \{x\}.$

•
$$FV(\lambda x.M) = FV(M) - \{x\}$$

- $FV(MN) = FV(\langle M, N \rangle) = FV(M) \cup FV(N).$
- FV(fst(M)) = FV(snd(M)) = FV(M).

Definition (informal)

occurrences of x in the scope of λ are called bound

M[x := N] denotes the result of substituting N for x in M



M[x := N] denotes the result of substituting N for x in M

•
$$(\lambda x.M)[x := N] = \lambda x.M$$

- $(\lambda y.M)[x := N] = \lambda y.(M[x := N])$, if $x \neq y$ and $y \notin FV(N)$
- $(M_1M_2)[x := N] = (M_1[x := N])(M_2[x := N])$



M[x := N] denotes the result of substituting N for x in M

•
$$(\lambda x.M)[x := N] = \lambda x.M$$

•
$$(\lambda y.M)[x := N] = \lambda y.(M[x := N])$$
, if $x \neq y$ and $y \notin FV(N)$

•
$$(M_1M_2)[x := N] = (M_1[x := N])(M_2[x := N])$$

•
$$\langle M_1, M_2 \rangle [x := N] = \langle M_1[x := N], M_2[x := N] \rangle$$

M[x := N] denotes the result of substituting N for x in M

•
$$(\lambda x.M)[x := N] = \lambda x.M$$

•
$$(\lambda y.M)[x := N] = \lambda y.(M[x := N])$$
, if $x \neq y$ and $y \notin FV(N)$

•
$$(M_1M_2)[x := N] = (M_1[x := N])(M_2[x := N])$$

•
$$\langle M_1, M_2 \rangle [x := N] = \langle M_1[x := N], M_2[x := N] \rangle$$

•
$$fst(M)[x := N] = fst(M[x := N])$$

M[x := N] denotes the result of substituting N for x in M

•
$$(\lambda x.M)[x := N] = \lambda x.M$$

- $(\lambda y.M)[x := N] = \lambda y.(M[x := N])$, if $x \neq y$ and $y \notin FV(N)$
- $(M_1M_2)[x := N] = (M_1[x := N])(M_2[x := N])$
- $\langle M_1, M_2 \rangle [x := N] = \langle M_1[x := N], M_2[x := N] \rangle$

•
$$snd(M)[x := N] = snd(M[x := N])$$

Definition (substitution)

M[x := N] denotes the result of substituting N for x in M

• x[x := N] = N and if $x \neq y$, then y[x := N] = y

•
$$(\lambda x.M)[x := N] = \lambda x.M$$

•
$$(\lambda y.M)[x := N] = \lambda y.(M[x := N])$$
, if $x \neq y$ and $y \notin FV(N)$

•
$$(M_1M_2)[x := N] = (M_1[x := N])(M_2[x := N])$$

•
$$\langle M_1, M_2 \rangle [x := N] = \langle M_1[x := N], M_2[x := N] \rangle$$

•
$$fst(M)[x := N] = fst(M[x := N])$$

•
$$snd(M)[x := N] = snd(M[x := N])$$

Definition (β -reduction)

$$\begin{array}{ccc} (\lambda x.M)N & \xrightarrow{\beta} M[x := N] \\ \mathrm{fst}(\langle M, N \rangle) & \xrightarrow{\beta} M \\ \mathrm{snd}(\langle M, N \rangle) & \xrightarrow{\beta} N \end{array}$$

GM (Institute of Computer Science @ UIBK)

Lemma

 β -reduction is closed under context:

$$M \xrightarrow{\beta} N \Longrightarrow \begin{cases} LM \xrightarrow{\beta} LN \\ ML \xrightarrow{\beta} NL \\ \lambda x.M \xrightarrow{\beta} \lambda x.N \\ \langle M, L \rangle \xrightarrow{\beta} \langle N, L \rangle \\ \langle L, M \rangle \xrightarrow{\beta} \langle L, N \rangle \\ fst(M) \xrightarrow{\beta} fst(N) \\ snd(M) \xrightarrow{\beta} snd(N) \end{cases}$$

GM (Institute of Computer Science @ UIBK)

Lemma

 β -reduction is closed under context:

$$M \xrightarrow{\beta} N \Longrightarrow \begin{cases} LM \xrightarrow{\beta} LN \\ ML \xrightarrow{\beta} NL \\ \lambda x.M \xrightarrow{\beta} \lambda x.N \\ \langle M, L \rangle \xrightarrow{\beta} \langle N, L \rangle \\ \langle L, M \rangle \xrightarrow{\beta} \langle L, N \rangle \\ fst(M) \xrightarrow{\beta} fst(N) \\ snd(M) \xrightarrow{\beta} snd(N) \end{cases}$$

Example

$$(\lambda f.\lambda x.fx)(\lambda x.x+1)0 \xrightarrow{\beta} (\lambda x.(\lambda x.x+1)x)0 \xrightarrow{\beta} (\lambda x.x+1)0 \xrightarrow{\beta} 1$$

Type Checking

$$\begin{array}{c|c} \overline{x:\sigma,\Gamma\Rightarrow x:\sigma} & \mathrm{ref} \\ \\ \times & \left| \begin{array}{c} \frac{\Gamma\Rightarrow M:\sigma}{\Gamma\Rightarrow \langle M,N\rangle:\sigma\times\tau} \end{array} \mathsf{pair} \frac{\Gamma\Rightarrow M:\sigma\times\tau}{\Gamma\Rightarrow\mathsf{fst}(M):\sigma} & \mathrm{fst} \ \frac{\Gamma\Rightarrow M:\sigma\times\tau}{\Gamma\Rightarrow\mathsf{snd}(M):\tau} & \mathrm{snd} \\ \\ \rightarrow & \left| \begin{array}{c} \frac{\Gamma,x:\sigma\Rightarrow M:\tau}{\Gamma\Rightarrow\lambda x.M:\sigma\to\tau} & \mathrm{abs} \end{array} \right| \begin{array}{c} \frac{\Gamma\Rightarrow M:\sigma\to\tau}{\Gamma\Rightarrow MN:\sigma} & \mathrm{snd} \\ \end{array} \right| \end{array}$$

Type Checking

$$\begin{array}{c|c} \overline{x:\sigma,\Gamma\Rightarrow x:\sigma} & \operatorname{ref} \\ \\ \times & \left| \begin{array}{c} \frac{\Gamma\Rightarrow M:\sigma}{\Gamma\Rightarrow \langle M,N\rangle:\sigma\times\tau} & \operatorname{pair} \frac{\Gamma\Rightarrow M:\sigma\times\tau}{\Gamma\Rightarrow \operatorname{fst}(M):\sigma} & \operatorname{fst} \frac{\Gamma\Rightarrow M:\sigma\times\tau}{\Gamma\Rightarrow \operatorname{snd}(M):\tau} & \operatorname{snd} \\ \\ \rightarrow & \left| \begin{array}{c} \frac{\Gamma,x:\sigma\Rightarrow M:\tau}{\Gamma\Rightarrow\lambda x.M:\sigma\to\tau} & \operatorname{abs} \end{array} \right| \begin{array}{c} \frac{\Gamma\Rightarrow M:\sigma\to\tau}{\Gamma\Rightarrow MN:\tau} & \operatorname{app} \end{array} \end{array}$$

Remarks

- different to type checking system in functional programming we have type assignment for product types
- weakening is incorporated into the axiom, sequents are presented as sets

Definition (Types as Formulas)



Definition (Types as Formulas)

(ref)	\sim	(Ax) + structural rules
(abs)	\sim	(→: i)
(app)	\sim	$(\rightarrow: e)$

$$egin{array}{rcl} ({\sf pair}) &\sim & (\wedge:{\sf i}) \ ({\sf fst}) &\sim & (\wedge:{\sf e}) \ ({\sf snd}) &\sim & (\wedge:{\sf e}) \end{array}$$

Question

what is the correspondence to \lor ?



Definition (Types as Formulas)

(ref)	\sim	(Ax) + structural rules
(abs)	\sim	(→: i)
(app)	\sim	$(\rightarrow: e)$

Question

what is the correspondence to \lor ?

Answer

sum types!

Definition (Types as Formulas)

(ref)	\sim	(Ax) + structural rules
(abs)	\sim	(→: i)
(app)	\sim	(→: e)

Question

what is the correspondence to \lor ?

Answer

sum types!

Definition

a (binary) sum type describes a set of values drawn from exactly two given types

GM (Institute of Computer Science @ UIBK)

Type System for Sum Types

$$\vee \begin{vmatrix} \frac{\Gamma \Rightarrow M : \sigma}{\Gamma \Rightarrow \operatorname{inl}(M) : \sigma + \tau} & \frac{\Gamma \Rightarrow N : \tau}{\Gamma \Rightarrow \operatorname{inr}(N) : \sigma + \tau} \\ \frac{\Gamma \Rightarrow M : \sigma + \tau \quad \Gamma, x : \sigma \Rightarrow N_1 : \gamma \quad \Gamma, y : \tau \Rightarrow N_2 : \gamma}{\Gamma \Rightarrow \operatorname{case} M \text{ of } \operatorname{inl}(x) \longrightarrow N_1 \mid \operatorname{inr}(y) \longrightarrow N_2 : \gamma} \end{vmatrix}$$

Type System for Sum Types

$$\vee \left| \begin{array}{c} \frac{\Gamma \Rightarrow M : \sigma}{\Gamma \Rightarrow \mathsf{inl}(M) : \sigma + \tau} & \frac{\Gamma \Rightarrow N : \tau}{\Gamma \Rightarrow \mathsf{inr}(N) : \sigma + \tau} \\ \frac{\Gamma \Rightarrow M : \sigma + \tau \quad \Gamma, x : \sigma \Rightarrow N_1 : \gamma \quad \Gamma, y : \tau \Rightarrow N_2 : \gamma}{\Gamma \Rightarrow \mathsf{case} \ M \ \mathsf{of} \ \mathsf{inl}(x) \longrightarrow N_1 \mid \mathsf{inr}(y) \longrightarrow N_2 : \gamma} \right|$$

Definition (β -reduction, cont'd)

$$\begin{array}{ccc} (\lambda x.M)N & \stackrel{\beta}{\to} M[x := N] \\ & \operatorname{fst}(\langle M, N \rangle) & \stackrel{\beta}{\to} M \\ & \operatorname{snd}(\langle M, N \rangle) & \stackrel{\beta}{\to} N \end{array}$$

$$\begin{array}{ccc} \operatorname{case} & \operatorname{inl}(M) \text{ of } & \operatorname{inl}(x) \longrightarrow N_1 \mid \operatorname{inr}(y) \longrightarrow N_2 & \stackrel{\beta}{\to} N_1[x := M] \\ \operatorname{case} & \operatorname{inr}(N) \text{ of } & \operatorname{inl}(x) \longrightarrow N_1 \mid \operatorname{inr}(y) \longrightarrow N_2 & \stackrel{\beta}{\to} N_2[y := N] \end{array}$$

GM (Institute of Computer Science @ UIBK)

Definition (Types as Formulas (cont'd))



Definition (Types as Formulas (cont'd))



Definition (Types as Formulas (cont'd))

(ref)	\sim	(Ax) + structural rules	(pair)	\sim	(∧ : i)
(abs)	\sim	(→: i)	(fst)	\sim	(∧ : e)
(app)	\sim	$(\rightarrow:e)$	(snd)	\sim	$(\wedge : e)$
(inl)	\sim	(∨ : i)	(inr)	\sim	(∨ : i)
(case)	\sim	(∨ : e)			

Definition (Curry-Howard)

the Curry-Howard correspondence (aka Curry-Howard isomorophism) consists of the following parts:

formulas = types

Definition (Types as Formulas (cont'd))

(ref)	\sim	(Ax) + structural rules	(pair)	\sim	(∧ : i)
(abs)	\sim	(→: i)	(fst)	\sim	(∧ : e)
(app)	\sim	$(\rightarrow:e)$	(snd)	\sim	(∧ : e)
(inl)	\sim	(∨ : i)	(inr)	\sim	(∨ : i)
(case)	\sim	(∨ : e)			

Definition (Curry-Howard)

the Curry-Howard correspondence (aka Curry-Howard isomorophism) consists of the following parts:

2 proof = programs

Definition (Types as Formulas (cont'd))

(ref)	\sim	(Ax) + structural rules	(pair)	\sim	(∧ : i)
(abs)	\sim	$(\rightarrow:i)$	(fst)	\sim	(∧ : e)
(app)	\sim	(→: e)	(snd)	\sim	$(\land : e)$
(inl)	\sim	(∨ : i)	(inr)	\sim	(∨ : i)
(case)	\sim	(∨ : e)			

Definition (Curry-Howard)

the Curry-Howard correspondence (aka Curry-Howard isomorophism) consists of the following parts:

3 normalisation = computation

Definition (Types as Formulas (cont'd))

(ref)	\sim	(Ax) + structural rules	(pair)	\sim	(∧ : i)
(abs)	\sim	(→: i)	(fst)	\sim	(∧ : e)
(app)	\sim	$(\rightarrow: e)$	(snd)	\sim	(∧ : e)
(inl)	\sim	(∨ : i)	(inr)	\sim	(∨:i)
(case)	\sim	(∨ : e)			

Definition (Curry-Howard)

the Curry-Howard correspondence (aka Curry-Howard isomorophism) consists of the following parts:

- 1 formulas = types
- **2** proof = programs

Definition (Types as Formulas (cont'd))

(ref)	\sim	(Ax) + structural rules	(pair)	\sim	(∧ : i)
(abs)	\sim	(→: i)	(fst)	\sim	(∧ : e)
(app)	\sim	$(\rightarrow:e)$	(snd)	\sim	(∧ : e)
(inl)	\sim	(∨ : i)	(inr)	\sim	(∨ : i)
(case)	\sim	(∨ : e)			

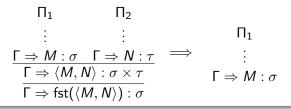
Definition (Curry-Howard)

the Curry-Howard correspondence (aka Curry-Howard isomorophism) consists of the following parts:

- 1 formulas = types
- 2 proof = programs
- 3 normalisation = computation

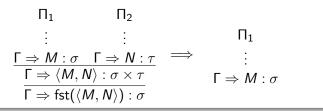
Proofs as Programs

Definition (normalisation)



Proofs as Programs

Definition (normalisation)





Definition (normalisation)

$$\begin{array}{ccccccc}
\Pi_{1} & & & \Pi_{2} & & \Pi_{1}[x \setminus \Pi_{2}] \\
\vdots & & & & \vdots & & & \vdots \\
\hline \Gamma \Rightarrow \lambda x.M : \sigma \to \tau & \Gamma \Rightarrow N : \sigma & & & \Gamma \Rightarrow M[x := N] : \tau \\
\hline \Gamma \Rightarrow (\lambda x.M)N : \tau & & & & \Gamma \Rightarrow M[x := N] : \tau
\end{array}$$

the proof $\Pi_1[x \setminus \Pi_2]$ represents the proof obtained from Π_1 by substituting Π_2 into Π_1 instead of the use of ref wrt x



Definition (normalisation)

the proof $\Pi_1[x \setminus \Pi_2]$ represents the proof obtained from Π_1 by substituting Π_2 into Π_1 instead of the use of ref wrt x



Discussion

Fact

the Curry-Howard correspondence extends to many systems, for example

- intuitionistic logic and λ-calculus
- Hilbert axioms and combinatory logic
- linear logic and interaction nets



Discussion

Fact

the Curry-Howard correspondence extends to many systems, for example

- intuitionistic logic and λ -calculus
- Hilbert axioms and combinatory logic
- linear logic and interaction nets

Observations

the Curry-Howard correspondence

- Inks logic with programming, i.e., provides an explanation for the sucess of logic in computer science
- 2 allows to mutual enrich both areas

Discussion

Fact

the Curry-Howard correspondence extends to many systems, for example

- intuitionistic logic and λ -calculus
- Hilbert axioms and combinatory logic
- linear logic and interaction nets

Observations

4 . . .

the Curry-Howard correspondence

- Inks logic with programming, i.e., provides an explanation for the sucess of logic in computer science
- 2 allows to mutual enrich both areas
- 3 provides a formally verified form of programming

- strong normalisation of simply typed $\lambda\text{-}calculus$ is typically proved via strong normalisation of minimal logic



- strong normalisation of simply typed $\lambda\text{-}calculus$ is typically proved via strong normalisation of minimal logic
- similarly, undecidablilty of type inhabitation of dependent types follows from undeciability of intuitionistic predicate logic



- strong normalisation of simply typed $\lambda\text{-}calculus$ is typically proved via strong normalisation of minimal logic
- similarly, undecidablilty of type inhabitation of dependent types follows from undeciability of intuitionistic predicate logic

Example

correspondencence between interaction nets and linear logic provides type system to interaction nets

- strong normalisation of simply typed $\lambda\text{-}calculus$ is typically proved via strong normalisation of minimal logic
- similarly, undecidablilty of type inhabitation of dependent types follows from undeciabilty of intuitionistic predicate logic

Example

correspondencence between interaction nets and linear logic provides type system to interaction nets

13+SIGT

Example

- formalisation of the theory of forbidden patterns for rewrite strategies in Isabelle provides a machine-checked theory
- code export from Isabelle provides OCaml code that has been integrated into $\mathsf{T}_T\mathsf{T}_2$