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* $\Pi = \Pi_1, \dots, \Pi_n$
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. *$\mathcal{G}$ is satisfiable*
2. *$\mathcal{G}$ has a Herbrand model*
3. *$\forall$ finite $\mathcal{G}_0 \subseteq \mathrm{Gr}(\mathcal{G})$, $\mathcal{G}_0$ 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: $G(t_1^1, \dots, t_n^1) \vee \cdots \vee G(t_1^k, \dots, t_n^k)$

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

Background: Russel's paradox

Definition

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

Background: Russel's paradox

Definition

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

Proof.

Background: Russel's paradox

Definition

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

Proof.

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

Background: Russel's paradox

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

Background: Russel's paradox

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

$$R \in R \to R \notin R \qquad R \notin R \to R \in R$$

Background: Russel's paradox

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

$$R \in R \to R \notin R \qquad R \notin R \to R \in R$$

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

Background: Russel's paradox

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

$$R \in R \to R \notin R \qquad R \notin R \to R \in R$$

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 $A \vee \neg A$ is no longer valid

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
- 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)

A Problem with the Excluded Middle

Theorem

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

A Problem with the Excluded Middle

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

A Problem with the Excluded Middle

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

A Problem with the Excluded Middle

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
     $$x = \sqrt{2} \qquad y = \sqrt{2} \qquad z = \sqrt{2}^{\sqrt{2}}$$
   - $\sqrt{2}^{\sqrt{2}}$ is irrational, then

A Problem with the Excluded Middle

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
   $$x = \sqrt{2} \qquad y = \sqrt{2} \qquad z = \sqrt{2}^{\sqrt{2}}$$
   - $\sqrt{2}^{\sqrt{2}}$ is irrational, then
   $$x = \sqrt{2}^{\sqrt{2}} \qquad y = \sqrt{2} \qquad z = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = 2$$

A Problem with the Excluded Middle

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
     $$x = \sqrt{2} \qquad y = \sqrt{2} \qquad z = \sqrt{2}^{\sqrt{2}}$$
   - $\sqrt{2}^{\sqrt{2}}$ is irrational, then
     $$x = \sqrt{2}^{\sqrt{2}} \qquad y = \sqrt{2} \qquad z = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = 2$$

$\blacksquare$

A Problem with the Excluded Middle

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

     $$x = \sqrt{2} \qquad y = \sqrt{2} \qquad z = \sqrt{2}^{\sqrt{2}}$$

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

     $$x = \sqrt{2}^{\sqrt{2}} \qquad y = \sqrt{2} \qquad z = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2} \cdot \sqrt{2}} = 2$$

∎

prototypical example of a non-constructive proof

Intuitionistic Logic

	introduction	elimination
$\wedge$	$\dfrac{E \quad F}{E \wedge F}\ \wedge\colon \mathrm{i}$	$\dfrac{E \wedge F}{E}\ \wedge\colon \mathrm{e} \qquad \dfrac{E \wedge F}{F}\ \wedge\colon \mathrm{e}$
$\vee$	$\dfrac{E}{E \vee F}\ \vee\colon \mathrm{i} \qquad \dfrac{F}{F \vee F}\ \vee\colon \mathrm{i}$	$\dfrac{E \vee F \quad \boxed{\begin{array}{c} E \\ \vdots \\ G \end{array}} \quad \boxed{\begin{array}{c} F \\ \vdots \\ G \end{array}}}{G}\ \vee\colon \mathrm{e}$
$\rightarrow$	$\dfrac{\boxed{\begin{array}{c} E \\ \vdots \\ F \end{array}}}{E \rightarrow F}\ \rightarrow\colon \mathrm{i}$	$\dfrac{E \quad E \rightarrow F}{F}\ \rightarrow\colon \mathrm{e}$

	introduction	elimination
$\neg$	$\dfrac{\boxed{\begin{array}{c} E \\ \vdots \\ \bot \end{array}}}{\neg E}\ \neg\!:\mathrm{i}$	$\dfrac{F \quad \neg F}{\bot}\ \neg\!:\mathrm{e}$
$\bot$		$\dfrac{\bot}{F}\ \neg\!:\mathrm{e}$

	introduction	elimination
$\neg$	$\dfrac{\boxed{\begin{array}{c} E \\ \vdots \\ \bot \end{array}}}{\neg E}\ \neg\!:\mathrm{i}$	$\dfrac{F \quad \neg F}{\bot}\ \neg\!:\mathrm{e}$
$\bot$		$\dfrac{\bot}{F}\ \neg\!:\mathrm{e}$

Remark

note the absence of

$$\frac{\neg\neg F}{F}\ \neg\neg\!:\mathrm{e}$$

Definition (Alternative)

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 (Alternative)

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)

Definition (Alternative)

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

Definition (Alternative)

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 argument for $E \vee F$ is an argument of E or F

Definition (Alternative)

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 argument for $E \vee F$ is an argument of E or F
- an argument for $E \to F$ is a transformation of an argument for E into an argument for F

Definition (Alternative)

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 argument for $E \vee F$ is an argument of E or F
- an argument for $E \to F$ is a transformation of an argument for E into an argument for F
- $\neg E$ is interpreted as $E \to \perp$

Definition (Alternative)

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 argument for $E \vee F$ is an argument of E or F
- an argument for $E \to F$ is a transformation of an argument for E into an argument for F
- $\neg E$ is interpreted as $E \to \bot$
- no argument for $\bot$ can exist

Definition (Alternative)

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 argument for $E \vee F$ is an argument of E or F
- an argument for $E \to F$ is a transformation of an argument for E into an argument for F
- $\neg E$ is interpreted as $E \to \perp$
- no argument for $\perp$ can exist

the formal definition needs Kripke models

Kripke Models

Definition

- a frame $\mathcal{F}$ is a pair $(W, \leqslant)$, where W denotes a nonempty set of worlds and $\leqslant$ denotes a preorder on W

Kripke Models

Definition

- a frame $\mathcal{F}$ is a pair $(W, \leqslant)$, where W denotes a nonempty set of worlds and $\leqslant$ denotes a preorder on W
- a Kripke model on a frame $F = (W, \leqslant)$ is a triple

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

such that for all p:

1. $\mathcal{A}_p = (A_p, a_p)$
2. A_p is a non-empty set (the domain in world p)
3. A_p is a mapping that associates predicate constants to domains

Kripke Models

Definition

- a frame $\mathcal{F}$ is a pair $(W, \leqslant)$, where W denotes a nonempty set of worlds and $\leqslant$ denotes a preorder on W
- a Kripke model on a frame $F = (W, \leqslant)$ is a triple
$$\mathcal{K} = (W, \leqslant, (\mathcal{A}_p)_{p \in W})$$
such that for all p:
  1. $\mathcal{A}_p = (A_p, a_p)$
  2. A_p is a non-empty set (the domain in world p)
  3. A_p is a mapping that associates predicate constants to domains
- $\forall$ predicate symbols P, $p, q \in W$, $(a_1, \ldots, a_n) \in A_p^n$:
$$p \leqslant q, \mathcal{A}_p \models P(a_1, \ldots, a_n) \quad \text{implies} \quad \mathcal{A}_q \models P(a_1, \ldots, a_n)$$

Kripke Models

Definition

- a frame $\mathcal{F}$ is a pair $(W, \leqslant)$, where W denotes a nonempty set of worlds and $\leqslant$ denotes a preorder on W
- a Kripke model on a frame $F = (W, \leqslant)$ is a triple

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

  such that for all p:

  1. $\mathcal{A}_p = (A_p, a_p)$
  2. A_p is a non-empty set (the domain in world p)
  3. A_p is a mapping that associates predicate constants to domains

- $\forall$ predicate symbols P, $p, q \in W$, $(a_1, \ldots, a_n) \in A_p^n$:

$$p \leqslant q, \mathcal{A}_p \models P(a_1, \ldots, a_n) \quad \text{implies} \quad \mathcal{A}_q \models P(a_1, \ldots, a_n)$$

- 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 $F(x_1, \ldots, x_n)$ 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

Definition

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

$\mathcal{K}, p \Vdash \top$	$\mathcal{K}, p \nVdash \bot$
$\mathcal{K}, p \Vdash P(a_1, \ldots, a_n)$	if $\mathcal{A}_p \models P(a_1, \ldots, a_n)$
$\mathcal{K}, p \Vdash A \wedge B$	iff $\mathcal{K}, p \Vdash A$ and $\mathcal{K}, p \Vdash B$
$\mathcal{K}, p \Vdash A \vee B$	iff $\mathcal{K}, p \Vdash A$ or $\mathcal{K}, p \Vdash B$
$\mathcal{K}, p \Vdash A \rightarrow B$	iff for all $q \geqslant p$: $\mathcal{K}, q \Vdash A$ implies $\mathcal{K}, q \Vdash B$

Convention

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

Definition

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

$\mathcal{K}, p \Vdash \top$	$\mathcal{K}, p \nVdash \bot$
$\mathcal{K}, p \Vdash P(a_1, \dots, a_n)$	if $\mathcal{A}_p \models P(a_1, \dots, a_n)$
$\mathcal{K}, p \Vdash A \wedge B$	iff $\mathcal{K}, p \Vdash A$ and $\mathcal{K}, p \Vdash B$
$\mathcal{K}, p \Vdash A \vee B$	iff $\mathcal{K}, p \Vdash A$ or $\mathcal{K}, p \Vdash B$
$\mathcal{K}, p \Vdash A \rightarrow B$	iff for all $q \geqslant p$: $\mathcal{K}, q \Vdash A$ implies $\mathcal{K}, q \Vdash B$

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

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	elimination
	$A \Rightarrow A$	
$\wedge$	$\dfrac{\Gamma \Rightarrow E \quad \Gamma \Rightarrow F}{\Gamma \Rightarrow E \wedge F}$	$\dfrac{\Gamma \Rightarrow E \wedge F}{\Gamma \Rightarrow E}$ $\quad$ $\dfrac{\Gamma \Rightarrow E \wedge F}{\Gamma \Rightarrow F}$
$\vee$	$\dfrac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E \vee F}$ $\quad$ $\dfrac{\Gamma \Rightarrow F}{\Gamma \Rightarrow E \vee F}$	$\dfrac{\Gamma \Rightarrow E \vee F \quad \Gamma, E \Rightarrow G \quad \Gamma, F \Rightarrow G}{\Gamma \Rightarrow G}$
$\rightarrow$	$\dfrac{\Gamma, E \Rightarrow F}{\Gamma \Rightarrow E \rightarrow F}$	$\dfrac{\Gamma \Rightarrow E \quad \Gamma \Rightarrow E \rightarrow F}{\Gamma \Rightarrow F}$

A Sequent Calculus for Minimal Logic

	left	right
$\wedge$	$\dfrac{E, \Gamma \Rightarrow C}{E \wedge F, \Gamma \Rightarrow C}\ \wedge\!: \mathsf{l}$ $\dfrac{F, \Gamma \Rightarrow C}{E \wedge F, \Gamma \Rightarrow C}\ \wedge\!: \mathsf{l}$	$\dfrac{\Gamma_1 \Rightarrow E \quad \Gamma_2 \Rightarrow F}{\Gamma_1, \Gamma_2 \Rightarrow E \wedge F}\ \wedge\!: \mathsf{r}$
$\vee$	$\dfrac{E, \Gamma_1 \Rightarrow C \quad F, \Gamma_2 \Rightarrow C}{E \vee F, \Gamma_1, \Gamma_2 \Rightarrow C}\ \vee\!: \mathsf{l}$	$\dfrac{\Gamma \Rightarrow E}{\Gamma \Rightarrow E \vee F}\ \vee\!: \mathsf{r}$ $\dfrac{\Gamma \Rightarrow F}{\Gamma \Rightarrow E \vee F}\ \vee\!: \mathsf{r}$
$\rightarrow$	$\dfrac{\Gamma_1 \Rightarrow E \quad F, \Gamma_2 \Rightarrow C}{E \rightarrow F, \Gamma_1, \Gamma_2 \Rightarrow C}\ \rightarrow\!: \mathsf{l}$	$\dfrac{\Gamma, E \Rightarrow F}{\Gamma \Rightarrow E \rightarrow F}\ \rightarrow\!: \mathsf{l}$

Lemma

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*

Lemma

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 Π, i.e., on the number of sequents in Π

Lemma

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 Π, i.e., on the number of sequents in Π

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

Lemma

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 Π, 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:

$$\frac{\begin{matrix}\Pi_0 \\ \Gamma \Rightarrow E \wedge F\end{matrix}}{\Gamma \Rightarrow E}$$

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

Proof (cont'd).

3. the following is a correct proof:

$$\dfrac{\begin{array}{c}\Psi_0\\ \Gamma \Rightarrow E \wedge F\end{array} \quad \dfrac{E \Rightarrow E}{E \wedge F \Rightarrow E}\,\wedge : \mathsf{l}}{\Gamma \Rightarrow E}\,\mathit{cut}$$

Proof (cont'd).

3 the following is a correct proof:

$$\dfrac{\begin{array}{c}\Psi_0 \\ \Gamma \Rightarrow E \wedge F\end{array} \quad \dfrac{E \Rightarrow E}{E \wedge F \Rightarrow E}\,{\color{red}\wedge\colon \mathrm{l}}}{\Gamma \Rightarrow E}\,{\color{red}cut}$$

4 all other cases are similar

Proof (cont'd).

3. the following is a correct proof:

$$\dfrac{\dfrac{\Psi_0}{\Gamma \Rightarrow E \wedge F} \quad \dfrac{E \Rightarrow E}{E \wedge F \Rightarrow E}\ \wedge\!: \mathsf{l}}{\Gamma \Rightarrow E}\ \mathit{cut}$$

4. all other cases are similar

the other direction follows by induction on the length of Ψ ∎

Proof (cont'd).

3. the following is a correct proof:

$$\dfrac{\dfrac{\Psi_0}{\Gamma \Rightarrow E \wedge F} \quad \dfrac{E \Rightarrow E}{E \wedge F \Rightarrow E}\ \wedge\colon \mathrm{l}}{\Gamma \Rightarrow E}\ \mathit{cut}$$

4. all other cases are similar

the other direction follows by induction on the length of Ψ ■

Question

is this really correct?

Proof (cont'd).

3. the following is a correct proof:

$$\dfrac{\dfrac{\Psi_0}{\Gamma \Rightarrow E \wedge F} \quad \dfrac{E \Rightarrow E}{E \wedge F \Rightarrow E}\ \wedge\!: \mathsf{l}}{\Gamma \Rightarrow E}\ \mathit{cut}$$

4. all other cases are similar

the other direction follows by induction on the length of Ψ ∎

Question

is this really correct?

Answer

no, we forgot about the structural rules in the direction from right to left

"Natural Deduction" Structural Rules

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

"Natural Deduction" Structural Rules

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

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

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

Limits and Extensions of First Order Logic

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

Typed λ-Calculus

Definition (types and terms)

we define the set of types T and typed λ-terms as follows:

Typed λ-Calculus

Definition (types and terms)

we define the set of types T and typed λ-terms as follows:

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

Typed λ-Calculus

Definition (types and terms)

we define the set of types T and typed λ-terms as follows:

- 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 $(\sigma \to \tau)$ is a (function) type
- any (typed) variable $x : \sigma$ is a (typed) term

Typed λ-Calculus

Definition (types and terms)

we define the set of types T and typed λ-terms as follows:

- 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 $(\sigma \to \tau)$ 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 $\mathsf{fst}(M) : \sigma$ and $\mathsf{snd}(M) : \tau$ are terms

Typed λ-Calculus

Definition (types and terms)

we define the set of types T and typed λ-terms as follows:

- a variable type: $\alpha, \beta, \gamma, \ldots$
- if σ, τ are types, then $(\sigma \times \tau)$ is a (product) type
- if σ, τ are types, then $(\sigma \to \tau)$ 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 $\mathsf{fst}(M) : \sigma$ and $\mathsf{snd}(M) : \tau$ are terms
- if $M : \tau$ is a term, $x : \sigma$ a variable, then the abstraction $(\lambda x^\sigma.M) : \sigma \to \tau$ 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

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

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

Definition

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

- $\mathsf{FV}(x) = \{x\}$.
- $\mathsf{FV}(\lambda x.M) = \mathsf{FV}(M) - \{x\}$
- $\mathsf{FV}(MN) = \mathsf{FV}(\langle M, N \rangle) = \mathsf{FV}(M) \cup \mathsf{FV}(N)$.
- $\mathsf{FV}(\mathsf{fst}(M)) = \mathsf{FV}(\mathsf{snd}(M)) = \mathsf{FV}(M)$.

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!

Definition

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

- $\mathsf{FV}(x) = \{x\}$.
- $\mathsf{FV}(\lambda x.M) = \mathsf{FV}(M) - \{x\}$
- $\mathsf{FV}(MN) = \mathsf{FV}(\langle M, N \rangle) = \mathsf{FV}(M) \cup \mathsf{FV}(N)$.
- $\mathsf{FV}(\mathsf{fst}(M)) = \mathsf{FV}(\mathsf{snd}(M)) = \mathsf{FV}(M)$.

Definition (informal)

occurrences of x in the scope of λ are called bound

Definition (substitution)

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

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 $y \notin \mathrm{FV}(N)$
- $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$

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 $y \notin \mathrm{FV}(N)$
- $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$
- $\langle M_1, M_2\rangle[x := N] = \langle M_1[x := N], M_2[x := N]\rangle$

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 $y \notin \mathrm{FV}(N)$
- $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$
- $\langle M_1, M_2 \rangle[x := N] = \langle M_1[x := N], M_2[x := N] \rangle$
- $\mathsf{fst}(M)[x := N] = \mathsf{fst}(M[x := N])$

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 $y \notin \mathrm{FV}(N)$
- $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$
- $\langle M_1, M_2\rangle[x := N] = \langle M_1[x := N], M_2[x := N]\rangle$
- $\mathrm{fst}(M)[x := N] = \mathrm{fst}(M[x := N])$
- $\mathrm{snd}(M)[x := N] = \mathrm{snd}(M[x := N])$

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 $y \notin \mathsf{FV}(N)$
- $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$
- $\langle M_1, M_2 \rangle[x := N] = \langle M_1[x := N], M_2[x := N] \rangle$
- $\mathsf{fst}(M)[x := N] = \mathsf{fst}(M[x := N])$
- $\mathsf{snd}(M)[x := N] = \mathsf{snd}(M[x := N])$

Definition (β-reduction)

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

Lemma

β-reduction is closed under context:

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

Lemma

β-reduction is closed under context:

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

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

$$\frac{}{x : \sigma, \Gamma \Rightarrow x : \sigma}\ \text{ref}$$

$\times$

$$\frac{\Gamma \Rightarrow M : \sigma \quad \Gamma \Rightarrow N : \tau}{\Gamma \Rightarrow \langle M, N \rangle : \sigma \times \tau}\ \text{pair} \quad \frac{\Gamma \Rightarrow M : \sigma \times \tau}{\Gamma \Rightarrow \mathsf{fst}(M) : \sigma}\ \text{fst} \quad \frac{\Gamma \Rightarrow M : \sigma \times \tau}{\Gamma \Rightarrow \mathsf{snd}(M) : \tau}\ \text{snd}$$

$\to$

$$\frac{\Gamma, x : \sigma \Rightarrow M : \tau}{\Gamma \Rightarrow \lambda x.M : \sigma \to \tau}\ \text{abs} \qquad \frac{\Gamma \Rightarrow M : \sigma \to \tau \quad \Gamma \Rightarrow N : \sigma}{\Gamma \Rightarrow MN : \tau}\ \text{app}$$

Type Checking

$$\frac{}{x : \sigma, \Gamma \Rightarrow x : \sigma}\ \text{ref}$$

$\times$

$$\frac{\Gamma \Rightarrow M : \sigma \quad \Gamma \Rightarrow N : \tau}{\Gamma \Rightarrow \langle M, N\rangle : \sigma \times \tau}\ \text{pair} \quad \frac{\Gamma \Rightarrow M : \sigma \times \tau}{\Gamma \Rightarrow \mathsf{fst}(M) : \sigma}\ \text{fst} \quad \frac{\Gamma \Rightarrow M : \sigma \times \tau}{\Gamma \Rightarrow \mathsf{snd}(M) : \tau}\ \text{snd}$$

$\rightarrow$

$$\frac{\Gamma, x : \sigma \Rightarrow M : \tau}{\Gamma \Rightarrow \lambda x.M : \sigma \rightarrow \tau}\ \text{abs} \qquad \frac{\Gamma \Rightarrow M : \sigma \rightarrow \tau \quad \Gamma \Rightarrow N : \sigma}{\Gamma \Rightarrow MN : \tau}\ \text{app}$$

Remarks

1. different to type checking system in functional programming we have type assignment for product types
2. weakening is incorporated into the axiom, sequents arepresented as sets

Types as Formulas

Definition (Types as Formulas)

(ref)	$\sim$	(Ax) $+$ structural rules	(pair)	$\sim$	$(\wedge : \mathrm{i})$
(abs)	$\sim$	$(\rightarrow: \mathrm{i})$	(fst)	$\sim$	$(\wedge : \mathrm{e})$
(app)	$\sim$	$(\rightarrow: \mathrm{e})$	(snd)	$\sim$	$(\wedge : \mathrm{e})$

Types as Formulas

Definition (Types as Formulas)

$$
\begin{array}{lclclcl}
(\mathsf{ref}) & \sim & (\mathsf{Ax}) + \text{structural rules} & \qquad & (\mathsf{pair}) & \sim & (\wedge : \mathsf{i}) \\
(\mathsf{abs}) & \sim & (\rightarrow : \mathsf{i}) & & (\mathsf{fst}) & \sim & (\wedge : \mathsf{e}) \\
(\mathsf{app}) & \sim & (\rightarrow : \mathsf{e}) & & (\mathsf{snd}) & \sim & (\wedge : \mathsf{e})
\end{array}
$$

Question

what is the correspondence to $\vee$?

Types as Formulas

Definition (Types as Formulas)

$$\begin{array}{lll} (\mathsf{ref}) & \sim & (\mathsf{Ax}) + \text{structural rules} \\ (\mathsf{abs}) & \sim & (\rightarrow: \mathsf{i}) \\ (\mathsf{app}) & \sim & (\rightarrow: \mathsf{e}) \end{array} \qquad \begin{array}{lll} (\mathsf{pair}) & \sim & (\wedge : \mathsf{i}) \\ (\mathsf{fst}) & \sim & (\wedge : \mathsf{e}) \\ (\mathsf{snd}) & \sim & (\wedge : \mathsf{e}) \end{array}$$

Question

what is the correspondence to $\vee$?

Answer

sum types!

Types as Formulas

Definition (Types as Formulas)

$$\begin{array}{lcl@{\qquad\qquad}lcl}
(\mathsf{ref}) & \sim & (\mathsf{Ax}) + \text{structural rules} & (\mathsf{pair}) & \sim & (\wedge : \mathsf{i}) \\
(\mathsf{abs}) & \sim & (\rightarrow : \mathsf{i}) & (\mathsf{fst}) & \sim & (\wedge : \mathsf{e}) \\
(\mathsf{app}) & \sim & (\rightarrow : \mathsf{e}) & (\mathsf{snd}) & \sim & (\wedge : \mathsf{e})
\end{array}$$

Question

what is the correspondence to $\vee$?

Answer

sum types!

Definition

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

Type System for Sum Types

$$\vee \quad \frac{\Gamma \Rightarrow M : \sigma}{\Gamma \Rightarrow \mathsf{inl}(M) : \sigma + \tau} \qquad \frac{\Gamma \Rightarrow N : \tau}{\Gamma \Rightarrow \mathsf{inr}(N) : \sigma + \tau}$$

$$\frac{\Gamma \Rightarrow M : \sigma + \tau \quad \Gamma, x : \sigma \Rightarrow N_1 : \gamma \quad \Gamma, y : \tau \Rightarrow N_2 : \gamma}{\Gamma \Rightarrow \mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}(x) \longrightarrow N_1 \mid \mathsf{inr}(y) \longrightarrow N_2 : \gamma}$$

Type System for Sum Types

$$\vee \left| \begin{array}{c} \dfrac{\Gamma \Rightarrow M : \sigma}{\Gamma \Rightarrow \mathsf{inl}(M) : \sigma + \tau} \qquad \dfrac{\Gamma \Rightarrow N : \tau}{\Gamma \Rightarrow \mathsf{inr}(N) : \sigma + \tau} \\[2em] \dfrac{\Gamma \Rightarrow M : \sigma + \tau \quad \Gamma, x : \sigma \Rightarrow N_1 : \gamma \quad \Gamma, y : \tau \Rightarrow N_2 : \gamma}{\Gamma \Rightarrow \mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}(x) \longrightarrow N_1 \mid \mathsf{inr}(y) \longrightarrow N_2 : \gamma} \end{array} \right.$$

Definition (β-reduction, cont'd)

$$\begin{aligned} (\lambda x.M)N &\xrightarrow{\beta} M[x := N] \\ \mathsf{fst}(\langle M, N \rangle) &\xrightarrow{\beta} M \\ \mathsf{snd}(\langle M, N \rangle) &\xrightarrow{\beta} N \\ \mathsf{case}\ \mathsf{inl}(M)\ \mathsf{of}\ \mathsf{inl}(x) \longrightarrow N_1 \mid \mathsf{inr}(y) \longrightarrow N_2 &\xrightarrow{\beta} N_1[x := M] \\ \mathsf{case}\ \mathsf{inr}(N)\ \mathsf{of}\ \mathsf{inl}(x) \longrightarrow N_1 \mid \mathsf{inr}(y) \longrightarrow N_2 &\xrightarrow{\beta} N_2[y := N] \end{aligned}$$

Curry-Howard Correspondence

Definition (Types as Formulas (cont'd))

(ref)	$\sim$	(Ax) + structural rules	(pair)	$\sim$	$(\wedge : \mathrm{i})$
(abs)	$\sim$	$(\to: \mathrm{i})$	(fst)	$\sim$	$(\wedge : \mathrm{e})$
(app)	$\sim$	$(\to: \mathrm{e})$	(snd)	$\sim$	$(\wedge : \mathrm{e})$

Curry-Howard Correspondence

Definition (Types as Formulas (cont'd))

(ref)	$\sim$	(Ax) $+$ structural rules	(pair)	$\sim$	$(\wedge : \mathsf{i})$
(abs)	$\sim$	$(\rightarrow : \mathsf{i})$	(fst)	$\sim$	$(\wedge : \mathsf{e})$
(app)	$\sim$	$(\rightarrow : \mathsf{e})$	(snd)	$\sim$	$(\wedge : \mathsf{e})$
(inl)	$\sim$	$(\vee : \mathsf{i})$	(inr)	$\sim$	$(\vee : \mathsf{i})$
(case)	$\sim$	$(\vee : \mathsf{e})$			

Curry-Howard Correspondence

Definition (Types as Formulas (cont'd))

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

Definition (Curry-Howard)

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

1. formulas = types

Curry-Howard Correspondence

Definition (Types as Formulas (cont'd))

(ref)	$\sim$	(Ax) + structural rules	(pair)	$\sim$	$(\wedge : \mathrm{i})$
(abs)	$\sim$	$(\rightarrow : \mathrm{i})$	(fst)	$\sim$	$(\wedge : \mathrm{e})$
(app)	$\sim$	$(\rightarrow : \mathrm{e})$	(snd)	$\sim$	$(\wedge : \mathrm{e})$
(inl)	$\sim$	$(\vee : \mathrm{i})$	(inr)	$\sim$	$(\vee : \mathrm{i})$
(case)	$\sim$	$(\vee : \mathrm{e})$			

Definition (Curry-Howard)

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

2. proof = programs

Curry-Howard Correspondence

Definition (Types as Formulas (cont'd))

(ref)	$\sim$	(Ax) $+$ structural rules	(pair)	$\sim$	$(\wedge : \mathrm{i})$
(abs)	$\sim$	$(\rightarrow: \mathrm{i})$	(fst)	$\sim$	$(\wedge : \mathrm{e})$
(app)	$\sim$	$(\rightarrow: \mathrm{e})$	(snd)	$\sim$	$(\wedge : \mathrm{e})$
(inl)	$\sim$	$(\vee : \mathrm{i})$	(inr)	$\sim$	$(\vee : \mathrm{i})$
(case)	$\sim$	$(\vee : \mathrm{e})$			

Definition (Curry-Howard)

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

3. normalisation $=$ computation

Curry-Howard Correspondence

Definition (Types as Formulas (cont'd))

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

Definition (Curry-Howard)

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

1. formulas = types
2. proof = programs

Curry-Howard Correspondence

Definition (Types as Formulas (cont'd))

(ref)	$\sim$	(Ax) $+$ structural rules	(pair)	$\sim$	$(\wedge : \mathrm{i})$
(abs)	$\sim$	$(\to: \mathrm{i})$	(fst)	$\sim$	$(\wedge : \mathrm{e})$
(app)	$\sim$	$(\to: \mathrm{e})$	(snd)	$\sim$	$(\wedge : \mathrm{e})$
(inl)	$\sim$	$(\vee : \mathrm{i})$	(inr)	$\sim$	$(\vee : \mathrm{i})$
(case)	$\sim$	$(\vee : \mathrm{e})$			

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)

$$
\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \vdots\\ \Gamma \Rightarrow M : \sigma\end{array} \quad \begin{array}{c}\Pi_2\\ \vdots\\ \Gamma \Rightarrow N : \tau\end{array}}{\Gamma \Rightarrow \langle M, N\rangle : \sigma \times \tau}}{\Gamma \Rightarrow \mathsf{fst}(\langle M, N\rangle) : \sigma} \quad \Longrightarrow \quad \begin{array}{c}\Pi_1\\ \vdots\\ \Gamma \Rightarrow M : \sigma\end{array}
$$

Proofs as Programs

Definition (normalisation)

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \vdots \\ \Gamma \Rightarrow M : \sigma\end{array} \quad \begin{array}{c}\Pi_2\\ \vdots \\ \Gamma \Rightarrow N : \tau\end{array}}{\Gamma \Rightarrow \langle M, N\rangle : \sigma \times \tau}}{\Gamma \Rightarrow \mathsf{fst}(\langle M, N\rangle) : \sigma} \quad \Longrightarrow \quad \begin{array}{c}\Pi_1\\ \vdots \\ \Gamma \Rightarrow M : \sigma\end{array}$$

Definition (β-reduction)

$$\mathsf{fst}(\langle M, N\rangle) \xrightarrow{\beta} M$$

Definition (normalisation)

$$\dfrac{\dfrac{\begin{matrix}\Pi_1 \\ \vdots \\ \Gamma, x : \sigma \Rightarrow M : \tau\end{matrix}}{\Gamma \Rightarrow \lambda x.M : \sigma \to \tau} \quad \begin{matrix}\Pi_2 \\ \vdots \\ \Gamma \Rightarrow N : \sigma\end{matrix}}{\Gamma \Rightarrow (\lambda x.M)N : \tau} \quad \Longrightarrow \quad \begin{matrix}\Pi_1[x\backslash\Pi_2] \\ \vdots \\ \Gamma \Rightarrow M[x := N] : \tau\end{matrix}$$

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

Definition (normalisation)

$$\dfrac{\dfrac{\begin{matrix}\Pi_1 \\ \vdots \\ \Gamma, x : \sigma \Rightarrow M : \tau\end{matrix}}{\Gamma \Rightarrow \lambda x.M : \sigma \to \tau} \quad \begin{matrix}\Pi_2 \\ \vdots \\ \Gamma \Rightarrow N : \sigma\end{matrix}}{\Gamma \Rightarrow (\lambda x.M)N : \tau} \quad \Longrightarrow \quad \begin{matrix}\Pi_1[x\backslash\Pi_2] \\ \vdots \\ \Gamma \Rightarrow M[x := N] : \tau\end{matrix}$$

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

Definition (β-reduction)

$$(\lambda x M)N \quad \xrightarrow{\beta} \quad M$$

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

1. links 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

the Curry-Howard correspondence

1. links 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
4. ...

Example

- strong normalisation of simply typed λ-calculus is typically proved via strong normalisation of minimal logic

Example

- strong normalisation of simply typed λ-calculus is typically proved via strong normalisation of minimal logic
- similarily, undecidablilty of type inhabitation of dependent types follows from undeciabilty of intuitionistic predicate logic

Example

- strong normalisation of simply typed λ-calculus is typically proved via strong normalisation of minimal logic
- similarily, 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

Example

- strong normalisation of simply typed λ-calculus is typically proved via strong normalisation of minimal logic
- similarily, 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

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 $\mathsf{T_TT_2}$