Dependent Type Theory

Russell O'Connor

K∩W

2015-01-20

Dependent Type Theory

Dependent Type Theory unifies Logic,
Mathematics and Programming.

Dependent Type Theory

- There is a price to be paid from each discipline.
  - Logic must give up unrestricted proof by contradiction and logical completeness.
  - Programming must give up unbounded search and Turing completeness.
  - Mathematics must give up the axiom of choice.

Natural Deduction

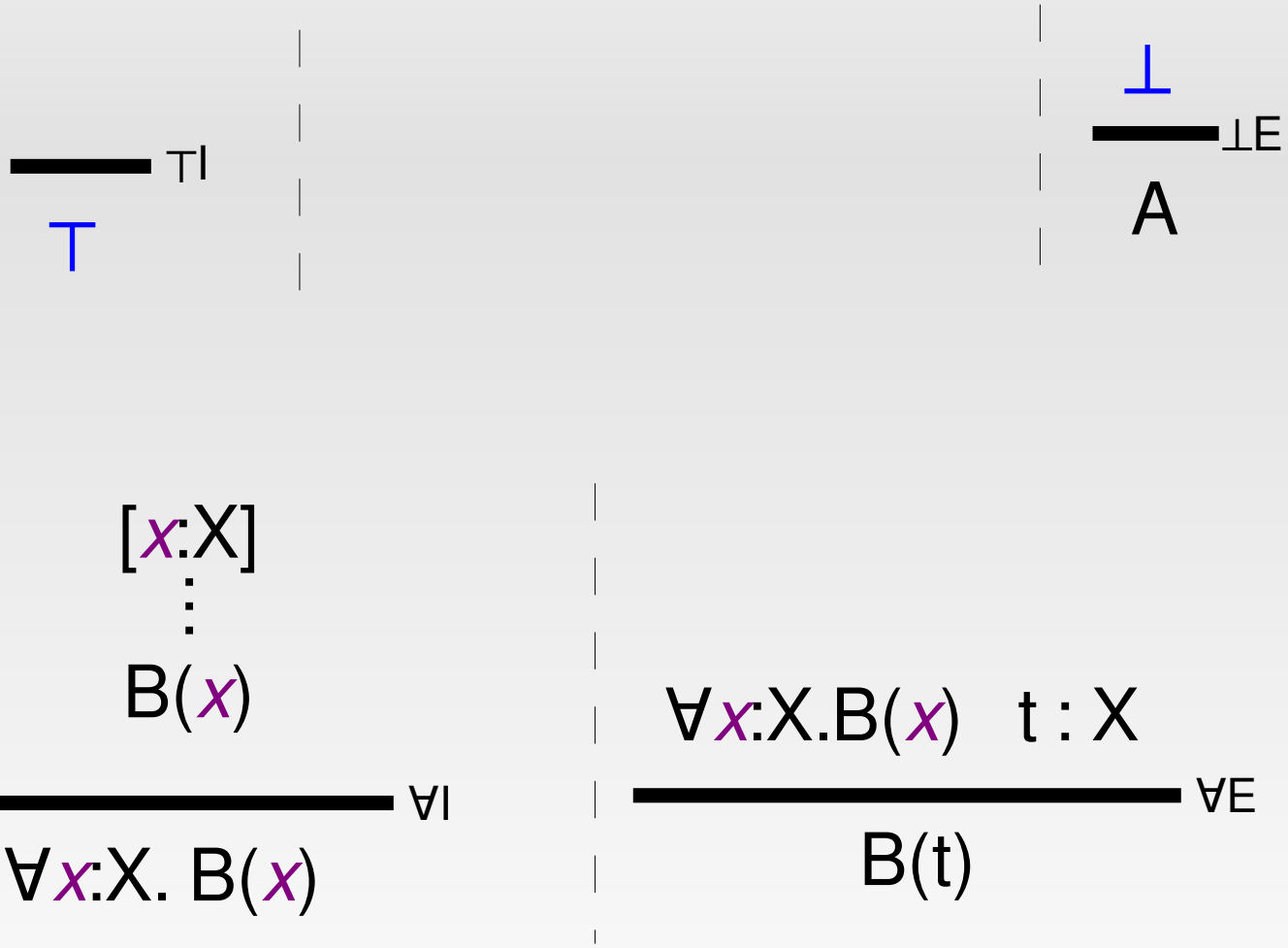
- For the moment, consider only the following logical connectives:
  - Universal quantification ($\forall$)
  - Conjunction ($\wedge$)
  - Implication ($\Rightarrow$)
  - True ($\top$)
  - False ($\bot$)

Natural Deduction

$$\frac{A \quad B}{A \wedge B}\,\wedge I \qquad \frac{A \wedge B}{A}\,\wedge E_1 \qquad \frac{A \wedge B}{B}\,\wedge E_2$$

$$\frac{\begin{array}{c}[a{:}A]\\ \vdots \\ B\end{array}}{A \Rightarrow B}\,\Rightarrow I(a) \qquad \frac{A \Rightarrow B \quad A}{B}\,\Rightarrow E$$

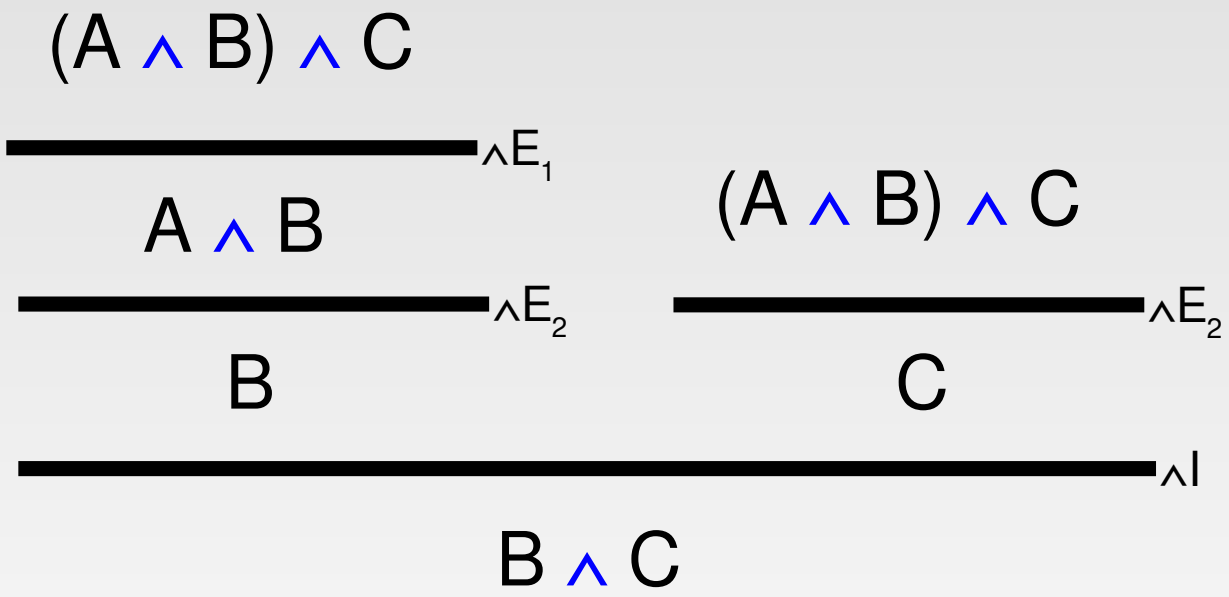
Natural Deduction



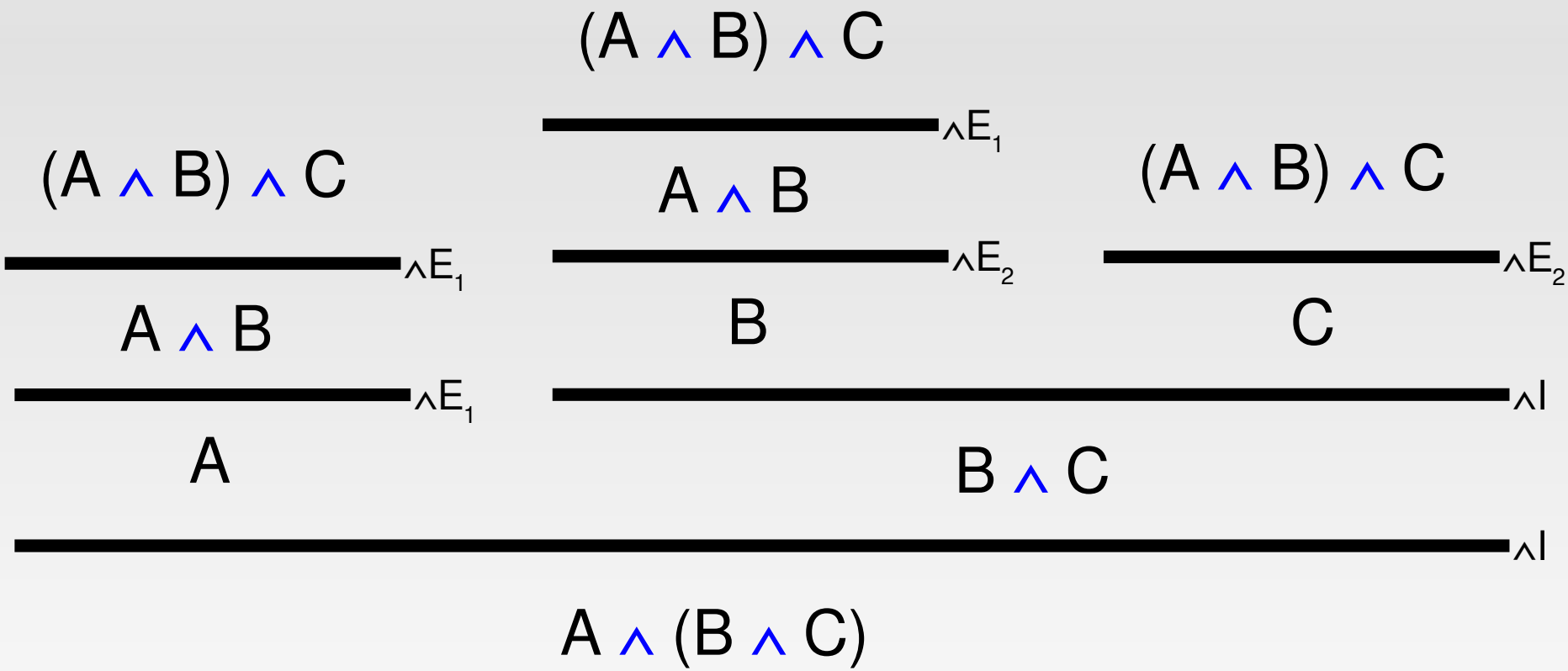
Example Deduction

$$\dfrac{\dfrac{(A \wedge B) \wedge C}{A \wedge B}\,\wedge E_1}{A}\,\wedge E_1$$

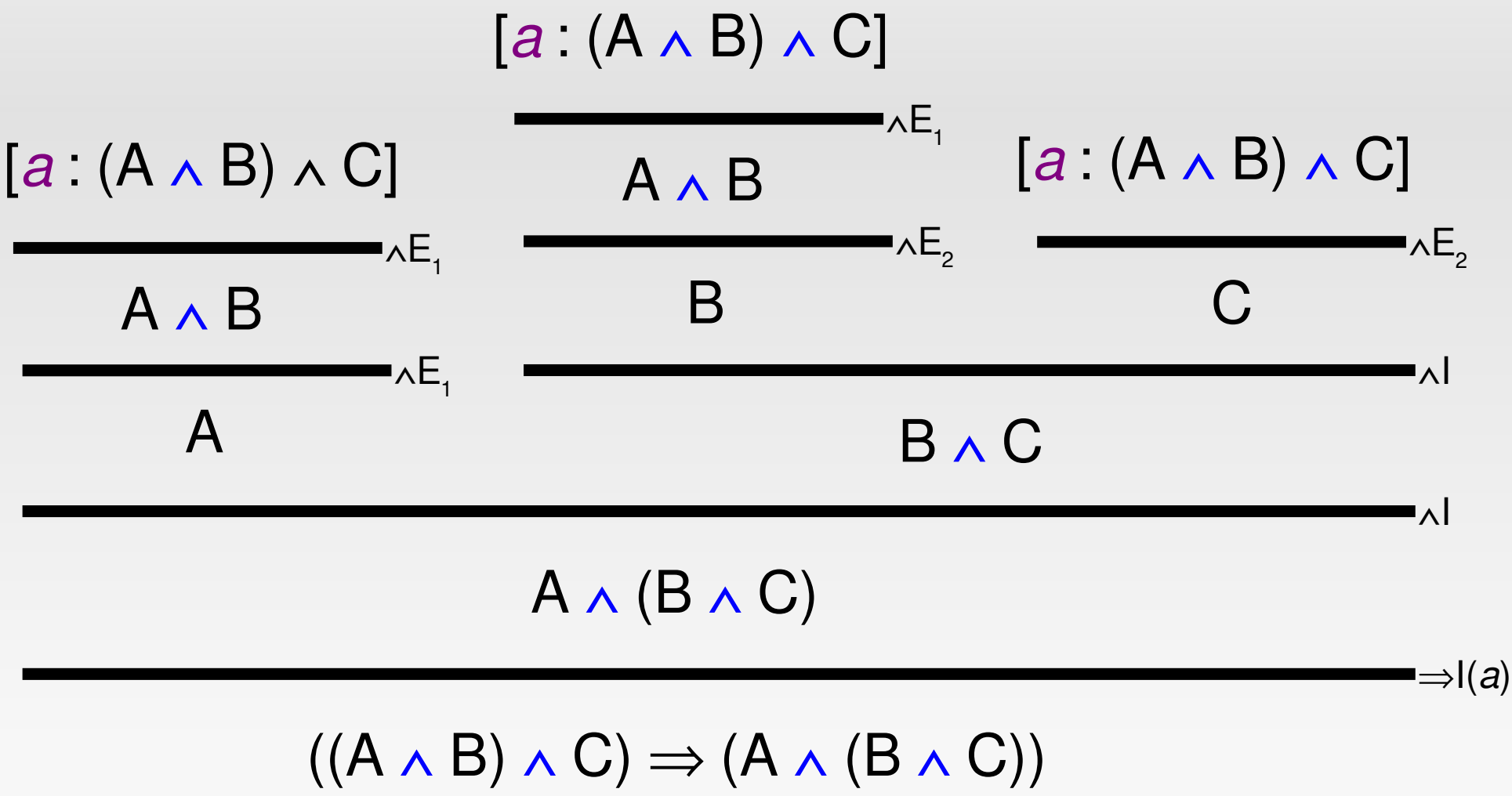
Example Deduction



Example Deduction



Example Deduction



Example Deduction

$$\dfrac{\dfrac{[b{:}\mathrm{B}]}{\mathrm{A} \Rightarrow \mathrm{B}}\Rightarrow\mathrm{I}(a)}{\mathrm{B} \Rightarrow (\mathrm{A} \Rightarrow \mathrm{B})}\Rightarrow\mathrm{I}(b)$$

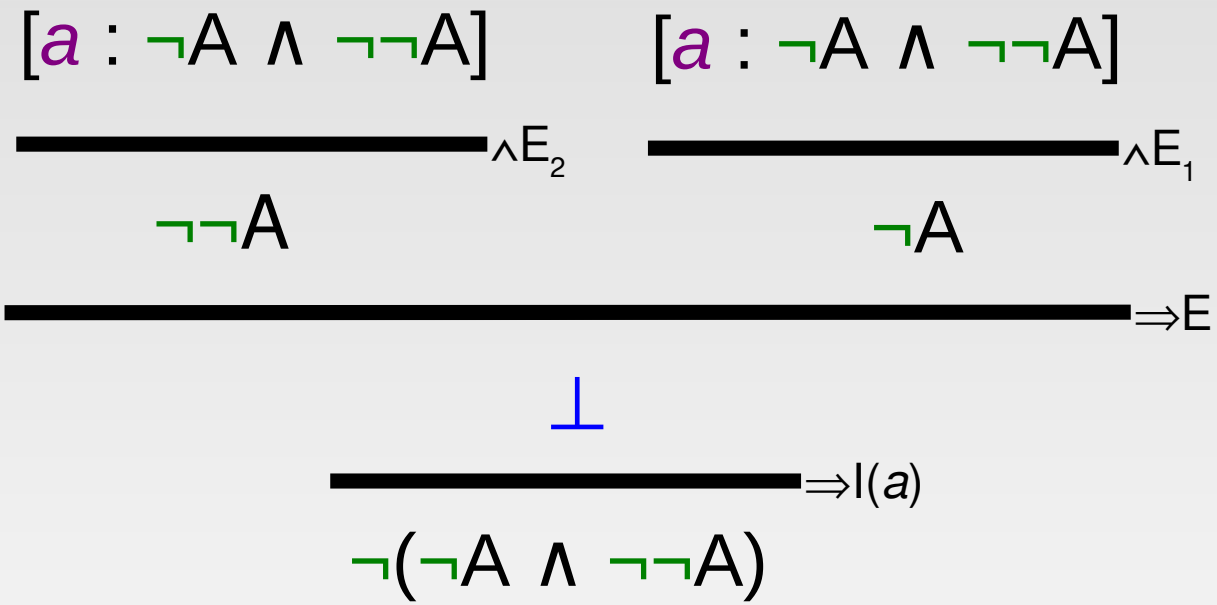
Connectives by Definition

- If and only if:
  - $A \Leftrightarrow B := (A \Rightarrow B) \wedge (B \Rightarrow A)$
- Negation:
  - $\neg A := A \Rightarrow \bot$
- Classical disjunction:
  - $A \vee_c B := \neg(\neg A \wedge \neg B)$
- Classical existential quantifier:
  - $\exists_c x{:}X.B(x) := \neg \forall x{:}X.\neg B(x)$

Proof by Contradiction

- A formula B is *stable* when:
  - $\neg\neg B \Rightarrow B$
- Stability is preserved by all connectives
  - ... so far
- This logic is classical if we assume all propositional variables are stable.
- Many atomic propositions are stable:
  - e.g. Equality between integers or real numbers

Excluded Middle



Constructive Connectives

$$\frac{A}{A \vee B}\,\vee I_1 \qquad \frac{B}{A \vee B}\,\vee I_2$$

$$\frac{A \vee B \qquad \begin{array}{c}[a{:}A]\\ \vdots\\ C\end{array} \qquad \begin{array}{c}[b{:}B]\\ \vdots\\ C\end{array}}{C}\,\vee E(a,b)$$

$$\frac{t : X \quad B(t)}{\exists x{:}X.B(x)}\,\exists I$$

$$\frac{\exists x{:}X.B(x) \qquad \begin{array}{c}[x{:}X;\ b{:}B(x)]\\ \vdots\\ C\end{array}}{C}\,\exists E(x,b)$$

Constructive Connectives

- The constructive disjuction and constructive existential create non-stable formula from stable formula.
  - Proof by contradition no longer works in general.
- Excluded middle cannot be derived for the constructive disjunction.

Proof Objects

$$\frac{p : A \quad q : B}{(p,q) : A \wedge B} \qquad \frac{r : A \wedge B}{(\pi_1\ r) : A} \qquad \frac{r : A \wedge B}{(\pi_2\ r) : B}$$

$$\frac{\begin{array}{c}[a{:}A] \\ \vdots \\ q : B\end{array}}{\lambda a{:}A.q : A \Rightarrow B} \qquad \frac{f : A \Rightarrow B \quad p : A}{(f\ p) : B}$$

Proof Objects

$$\frac{}{* : \top} \qquad \frac{p : \bot}{\mathbf{abort}_A(p) : A}$$

$$\frac{\begin{array}{c}[x{:}X] \\ \vdots \\ q : B(x)\end{array}}{\lambda x{:}X.q : \forall x{:}X.\ B(x)} \qquad \frac{f : \forall x{:}A.B(x) \quad t : A}{(f\ t) : B(t)}$$

Proof Objects

$$\frac{p : A}{(\sigma_{1,B}\ p) : A \vee B} \qquad \frac{q : B}{(\sigma_{2,A}\ q) : A \vee B}$$

$$\frac{r : A \vee B \qquad \begin{array}{c}[a{:}A] \\ \vdots \\ p : C\end{array} \qquad \begin{array}{c}[b{:}B] \\ \vdots \\ q : C\end{array}}{\mathbf{case}_{\vee}\ r\ \mathbf{of}\ \{\ a \rightarrow p;\ b \rightarrow q\ \} : C}$$

$$\frac{t : X \qquad p : B(t)}{(t,p)_B : \exists x{:}X.\ B(x)}$$

$$\frac{r : \exists x{:}X.B(x) \qquad \begin{array}{c}[x{:}X;\ b{:}B(x)] \\ \vdots \\ p : C\end{array}}{\mathbf{case}_{\exists}\ r\ \mathbf{of}\ \{\ x,b \rightarrow p\ \} : C}$$

Example Deduction

$$\dfrac{\dfrac{\dfrac{[a : (A \wedge B) \wedge C]}{(\pi_1\, a) : A \wedge B}}{(\pi_1\,(\pi_1\, a)) : A} \qquad \dfrac{\dfrac{[a : (A \wedge B) \wedge C]}{(\pi_1\, a) : A \wedge B} \qquad \dfrac{[a : (A \wedge B) \wedge C]}{(\pi_2\, a) : C}}{(\pi_2\,(\pi_1\, a)) : B}}{(\pi_2\,(\pi_1\, a), \pi_2\, a) : B \wedge C}$$

$$\dfrac{(\pi_1\,(\pi_1\, a), (\pi_2\,(\pi_1\, a), \pi_2\, a)) : A \wedge (B \wedge C)}{\lambda a{:}(A \wedge B) \wedge C.(\pi_1\,(\pi_1\, a), (\pi_2\,(\pi_1\, a), \pi_2\, a)) : ((A \wedge B) \wedge C) \Rightarrow (A \wedge (B \wedge C))}$$

Example Deduction

$$\lambda a{:}(A \wedge B) \wedge C.(\pi_1\,(\pi_1\,a),(\pi_2\,(\pi_1\,a),\pi_2\,a))$$
$$: ((A \wedge B) \wedge C) \Rightarrow (A \wedge (B \wedge C))$$

Computational Interpretation

- So far we have:
  - Reviewed natural deduction
  - Annotated the trees
- Now we are going to:
  - Execute the annotations!

Computational Interpretation

- $\boldsymbol{\pi}_1\ (p,q) \rightsquigarrow p$
- $\boldsymbol{\pi}_2\ (p,q) \rightsquigarrow q$
- $(\lambda x{:}X.q)\ p \rightsquigarrow q[x \mapsto p]$
- $\textbf{case}_{\vee}\ (\sigma_{1,B}\ r)\ \textbf{of}\ \{\ a \to p;\ b \to q\ \} \rightsquigarrow p[a \mapsto r]$
- $\textbf{case}_{\vee}\ (\sigma_{2,A}\ r)\ \textbf{of}\ \{\ a \to p;\ b \to q\ \} \rightsquigarrow q[b \mapsto r]$
- $\textbf{case}_{\exists}\ (t,p)_B\ \textbf{of}\ \{\ x,b \to q\ \} \rightsquigarrow q[x \mapsto t; b \mapsto p]$
- No rules for $*$ nor abort_A

Computational Interpretation

- Reduction preserves propositions (types)
  - If (π_1 (p,q) : A) then (p : A)
  - If ((λ*x*:X.q) p : B) then (q[*x*↦p] : B)
- Reduction "simplifies" proofs
  - Simpler proofs are not always shorter proofs
- Reduction always terminates
- This relationship between Logic and Functional Programing is called the Curry-Howard isomorphism.

Computational Interpretation

- p : $\exists x$:X.R(x) is a pair containing
  - a witness w:X
  - a proof of R(w)
- f : $\forall x$:X.$\exists y$:Y.R(x,y) is a function
  - Mapping any t:X to
    - a witness w:Y
    - a proof of R(t,w)

Computational Interpretation

- $x \leftrightarrow y$ is the reflexive, symmetric, transitive, congruence closure of $\rightsquigarrow$
- $x \leftrightarrow y$ is decidable
  - Reduce x and y as much as possible.
  - Check if the results are the same.

Inductive Data Types

$$\frac{}{t : \mathbb{B}} \qquad \frac{}{f : \mathbb{B}}$$

$$\frac{b : \mathbb{B} \quad p : C \quad q : C}{\textbf{if } b \textbf{ then } p \textbf{ else } q : C}$$

- **if** t **then** p **else** q $\rightsquigarrow$ p
- **if** f **then** p **else** q $\rightsquigarrow$ q

Inductive Data Types

$$\frac{}{0 : \mathbb{N}} \qquad \frac{n : \mathbb{N}}{(S\ n) : \mathbb{N}} \qquad \frac{n : \mathbb{N} \qquad p : C \qquad \begin{array}{c}[m:\mathbb{N};\ r:C] \\ \vdots \\ q : C\end{array}}{\mathbf{rec}_{\mathbb{N}}\ n\ \mathbf{with}\ \{\ p;\ m,\ r \to q\ \} : C}$$

- **rec** 0 **with** { p; *m*, *r* → q } ⇝ p
- **rec** (S n) **with** { p; *m*, *r* → q } ⇝
  q[*m*↦n;*r*↦**rec** n **with** { p; *m*, *r* → q }]
- Example:
  - x + y := **rec** x **with** { y; *m*, *r* → (S *r*) }

Inductive Data Types

- Structural Recursion Only
  - All functions terminate
  - Language is not Turing complete
  - Ackermann is still trivial to implement
- Next up: Induction

Dependent Types

- $(x =_{\mathbb{B}} y) := \textbf{if } x \textbf{ then } (\textbf{if } y \textbf{ then } \top \textbf{ else } \bot) \textbf{ else } (\textbf{if } y \textbf{ then } \bot \textbf{ else } \top)$
- $\bot : \bigstar$
- $\top : \bigstar$

Dependent Types

$$\frac{}{\bot : \star} \qquad \frac{}{\top : \star}$$

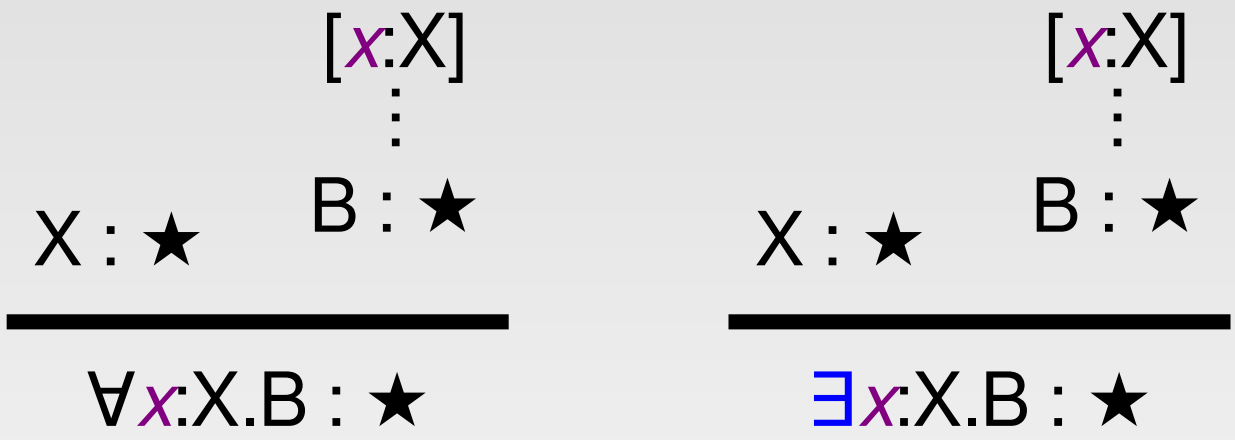
$$\frac{}{\mathbb{B} : \star} \qquad \frac{}{\mathbb{N} : \star}$$

$$\frac{A : \star \qquad B : \star}{A \wedge B : \star}$$

$$\frac{A : \star \qquad B : \star}{A \Rightarrow B : \star}$$

$$\frac{A : \star \qquad B : \star}{A \vee B : \star}$$

Dependent Types



Dependent Types

- $(x =_{\mathbb{B}} y) := \textbf{if } x \textbf{ then } (\textbf{if } y \textbf{ then } \top \textbf{ else } \bot) \textbf{ else } (\textbf{if } y \textbf{ then } \bot \textbf{ else } \top)$
- $\bot : \bigstar$
- $\top : \bigstar$
- We can build formulas such as
  - $(\forall x{:}\mathbb{B}.\exists y{:}\mathbb{B}.x =_{\mathbb{B}} y) : \bigstar$

Dependent Types

- not b := **if** b **then** f **else** t
- $x =_{\mathbb{B}} y$:= **if** x **then** y **else** (not y)
- $n =_{\mathbb{N}} m$:= **rec** n **with** { ... } : $\mathbb{B}$
- $\langle b \rangle$:= **if** b **then** $\top$ **else** $\bot$
- We can build formulas such as
  - $(\forall x{:}\mathbb{B}.\exists y{:}\mathbb{B}.\langle x =_{\mathbb{B}} y \rangle) : \star$
  - $(\forall x{:}\mathbb{N}.\langle x =_{\mathbb{N}} 0 \rangle \vee \exists y{:}\mathbb{N}.\langle x =_{\mathbb{N}} S\ y \rangle) : \star$
  - $(\forall x{:}\mathbb{N}.\forall y{:}\mathbb{N}.\langle x =_{\mathbb{N}} y \rangle) : \star$

Induction

$$\frac{}{0 : \mathbb{N}} \qquad \frac{n : \mathbb{N}}{(S\ n) : \mathbb{N}} \qquad \frac{n : \mathbb{N} \qquad p : C(0) \qquad \begin{array}{c}[m:\mathbb{N};\ r:C(m)] \\ \vdots \\ q : C(S\ m)\end{array}}{\mathbf{rec}_{\mathbb{N}}\ n\ \mathbf{with}\ \{\ p;\ m,\ r \to q\ \} : C(n)}$$

- $\mathbf{rec}_{\mathbb{N}}\ 0\ \mathbf{with}\ \{\ p;\ m,\ r \to q\ \} \rightsquigarrow p$
- $\mathbf{rec}_{\mathbb{N}}\ (S\ n)\ \mathbf{with}\ \{\ p;\ m,\ r \to q\ \} \rightsquigarrow q[m \mapsto n; r \mapsto \mathbf{rec}_{\mathbb{N}}\ n\ \mathbf{with}\ \{\ p;\ m,\ r \to q\ \}]$
- Now we can prove:
  - $\lambda x{:}\mathbb{N}.[\ldots\ \mathrm{rec}_{\mathbb{N}}\ x\ \ldots] : \forall x{:}\mathbb{N}.\forall y{:}\mathbb{N}.\langle x =_{\mathbb{N}} y\rangle \Longrightarrow \langle y =_{\mathbb{N}} x\rangle$

Conversion Rule

$$\frac{p : A \quad A \leftrightarrow B}{p : B}$$

Dependent Types

$$\frac{n : \mathbb{N} \qquad p : C(0) \qquad \begin{array}{c}[m:\mathbb{N};\ r:C(m)]\\ \vdots \\ q : C(S\ m)\end{array}}{\mathbf{rec}_{\mathbb{N}}\ n\ \mathbf{with}\ \{\ p;\ m,\ r \to q\ \} : C(n)}$$

$$\frac{b : \mathbb{B} \qquad p : C(t) \qquad q : C(f)}{\mathbf{if}\ b\ \mathbf{then}\ p\ \mathbf{else}\ q : C(b)}$$

$$\frac{r : A \vee B \qquad \begin{array}{c}[a:A]\\ \vdots \\ p : C(\sigma_{1,B}\ a)\end{array} \qquad \begin{array}{c}[b:B]\\ \vdots \\ q : C(\sigma_{2,A}\ b)\end{array}}{\mathbf{case}_{\vee}\ r\ \mathrm{of}\ \{\ a \to p;\ b \to q\ \} : C(r)}$$

$$\frac{r : \top \qquad p : C(*)}{\mathbf{case}_{*}\ r\ \mathrm{of}\ \{p\} : C(r)}$$

Dependent Types

$$\frac{r : \exists x{:}X.\ B(x) \qquad \begin{array}{c}[x{:}X;\ b{:}B(x)] \\ \vdots \\ q : C((x,b))\end{array}}{\mathbf{case}_{\exists}\ r\ \mathbf{of}\ \{\ x,b \rightarrow p\ \} : C(r)}$$

$$\frac{f : \forall x{:}X.\ B(x) \qquad t : X}{(f\ t) : B(t)}$$

Poincaré Principle

- Proof that $2 + 2 = 4$
  - $*: \langle 2 + 2 =_{\mathbb{N}} 4 \rangle$

Poincaré Principle

- Proof that $2 + 2 = 4$
  - $*: \langle 2 + 2 =_{\mathbb{N}} 4 \rangle$
- How?

Poincaré Principle

- Proof that $2 + 2 = 4$
  - $* : \langle 2 + 2 =_{\mathbb{N}} 4 \rangle$
- How?
  - $* : \top$
  - Check is $\top \leftrightarrow \langle 2 + 2 =_{\mathbb{N}} 4 \rangle$?
  - $\langle 2 + 2 =_{\mathbb{N}} 4 \rangle \rightsquigarrow^* \langle 4 =_{\mathbb{N}} 4 \rangle \rightsquigarrow^* \langle 0 =_{\mathbb{N}} 0 \rangle \rightsquigarrow^* \langle t \rangle \rightsquigarrow^* \top$
  - O.K.

Lemma 1

$$\mathsf{lemma1} : \forall n{:}\mathbb{N}.\ \langle n =_{\mathbb{N}} n + 0 \rangle$$

$$\mathsf{lemma1} := \lambda n{:}\mathbb{N}.\ \mathbf{rec}_{\mathbb{N}}\ n\ \mathbf{with}\ \{\ *;\ m,\ r \longrightarrow r\ \}$$

Lemma 1

$$\mathsf{lemma1} : \forall n{:}\mathbb{N}.\ \langle n =_{\mathbb{N}} n + 0 \rangle$$
$$\mathsf{lemma1}\ 0 \coloneqq *$$
$$\mathsf{lemma1}\ (\mathsf{S}\ m) \coloneqq \mathsf{lemma1}\ m$$

Lemma 1

$$\begin{aligned}
&\mathsf{lemma1}\ 0 : \langle 0 =_{\mathbb{N}} 0 + 0 \rangle \\
&\mathsf{lemma1}\ 0 : \langle 0 =_{\mathbb{N}} 0 \rangle \\
&\mathsf{lemma1}\ 0 : \langle t \rangle \\
&\mathsf{lemma1}\ 0 : \top \\
&\qquad\qquad * : \top
\end{aligned}$$

Lemma 1

$$\mathsf{lemma1} : \forall n{:}\mathbb{N}.\ \langle n =_{\mathbb{N}} n + 0 \rangle$$

$$\mathsf{lemma1}\ (\mathsf{S}\ m) : \langle \mathsf{S}\ m =_{\mathbb{N}} \mathsf{S}\ m + 0 \rangle$$

$$\mathsf{lemma1}\ (\mathsf{S}\ m) : \langle \mathsf{S}\ m =_{\mathbb{N}} \mathsf{S}\ (m + 0) \rangle$$

$$\mathsf{lemma1}\ (\mathsf{S}\ m) : \langle m =_{\mathbb{N}} m + 0 \rangle$$

$$\mathsf{lemma1}\ m : \langle m =_{\mathbb{N}} m + 0 \rangle$$

Lemma 2

$$\begin{aligned}
&\mathsf{lemma2} : \forall n{:}\mathbb{N}.\ \forall m{:}\mathbb{N}.\ \langle n =_{\mathbb{N}} m\rangle \Rightarrow \langle m =_{\mathbb{N}} n\rangle \\
&\mathsf{lemma2}\ 0\ 0\ H &&:= * \\
&\mathsf{lemma2}\ 0\ (\mathsf{S}\ m)\ H &&:= H \\
&\mathsf{lemma2}\ (\mathsf{S}\ n)\ 0\ H &&:= H \\
&\mathsf{lemma2}\ (\mathsf{S}\ n)\ (\mathsf{S}\ m)\ H &&:= \mathsf{lemma2}\ n\ m\ H
\end{aligned}$$

Interactice Proof Assisants

- Coq
  - Used by Gonthier in the Four Colour Theorem
  - Develiped at INRIA
- Agda 2
  - Developed at Chalmers University of Technology
- Epigram
  - Developed at University of Durham

http://coq.inria.fr/

Intepreting Constructive Mathematics

- *Proof* valued, rather that *Truth* valued
  - A proof A $\vee$ B means that either A is provable or B is provable.
- Intepreting Classical Fragment
  - Functional interpretation if classical math is typically a function returning $\bot$ or otherwise trivial object.
    - Recall A $\vee_c$ B :=$\neg(\neg A \wedge \neg B)$
    - A function returning $\bot$ can never be evaluated.
    - The parameters can never simulatneously be satified.

TTFP

- Type Theory and Function Programming
  - By Simon Thompson

http://www.cs.kent.ac.uk/people/staff/sjt/TTFP/