Prove e Tipi: Un'introduzione all'isomorfismo di Curry-Howard

Dipartimento di Matematica ed Informatica, Università di Cagliari

9 maggio 2012

Paolo Di Giamberardino
digiambe@unica.it

Referenze

- J.Y. Girard, Y. Lafont, P. Taylor : *Proofs and Types*. Cambridge University, 1989.
- L. Tortora di Falco: *Sulla struttura logica del calcolo*. Rendiconti di Matematica e delle sue applicazioni (vol. 26, serie VII - pagg. 367-404), 2006.

Le slides sono disponibili sul sito del gruppo di ricerca *Trustworthy Computational Societies*, all'indirizzo `http://tcs.unica.it`

Indice

Indice

Cosa c'entra la logica con l'informatica?

David Hilbert

La questione dei fondamenti

Teoria degli insiemi (Cantor): L'insieme $\mathbb{R}$ dei numeri reali non è numerabile i.e. non può essere messo in corrispondenza biunivoca con l'insieme $\mathbb{N}$ dei numeri naturali: la sua cardinalità è strettamente superiore.

La questione dei fondamenti

Teoria degli insiemi (Cantor): L'insieme $\mathbb{R}$ dei numeri reali non è numerabile i.e. non può essere messo in corrispondenza biunivoca con l'insieme $\mathbb{N}$ dei numeri naturali: la sua cardinalità è strettamente superiore.

Geometrie non euclidee (Riemann): Scoperte nel tentativo di fornire una dimostrazione per assurdo del V postulato di Euclide ("*Per un punto passa una e una sola retta parallela ad una retta data*), sono geometrie costruite negando il V postulato.

La questione dei fondamenti

Teoria degli insiemi (Cantor): L'insieme $\mathbb{R}$ dei numeri reali non è numerabile i.e. non può essere messo in corrispondenza biunivoca con l'insieme $\mathbb{N}$ dei numeri naturali: la sua cardinalità è strettamente superiore.

Geometrie non euclidee (Riemann): Scoperte nel tentativo di fornire una dimostrazione per assurdo del V postulato di Euclide ("*Per un punto passa una e una sola retta parallela ad una retta data*), sono geometrie costruite negando il V postulato.

Secondo problema di Hilbert:

E' possibile dimostrare la consistenza della matematica?.

Entscheidungsproblem

Problema della decisione (Hilbert-Ackermann, 1928):

Dato un enunciato dell' l'aritmetica, esiste una procedura meccanica effettiva capace di stabilire se l'enunciato in questione sia deducibile o meno dagli assiomi?

Entscheidungsproblem

Problema della decisione (Hilbert-Ackermann, 1928):

Dato un enunciato dell' l'aritmetica, esiste una procedura meccanica effettiva capace di stabilire se l'enunciato in questione sia deducibile o meno dagli assiomi?

- Es: "*ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi*" (congettura di Goldbach)

Entscheidungsproblem

Problema della decisione (Hilbert-Ackermann, 1928):

Dato un enunciato dell' l'aritmetica, esiste una procedura meccanica effettiva capace di stabilire se l'enunciato in questione sia deducibile o meno dagli assiomi?

- Es: "*ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi*" (congettura di Goldbach)
- La procedura prende in input l'enunciato e restituisce in output una risposta, "Sì" o "No", in un numero finito di passi.

Entscheidungsproblem

Problema della decisione (Hilbert-Ackermann, 1928):

Dato un enunciato dell' l'aritmetica, esiste **una procedura meccanica effettiva** capace di stabilire se l'enunciato in questione sia deducibile o meno dagli assiomi?

Entscheidungsproblem

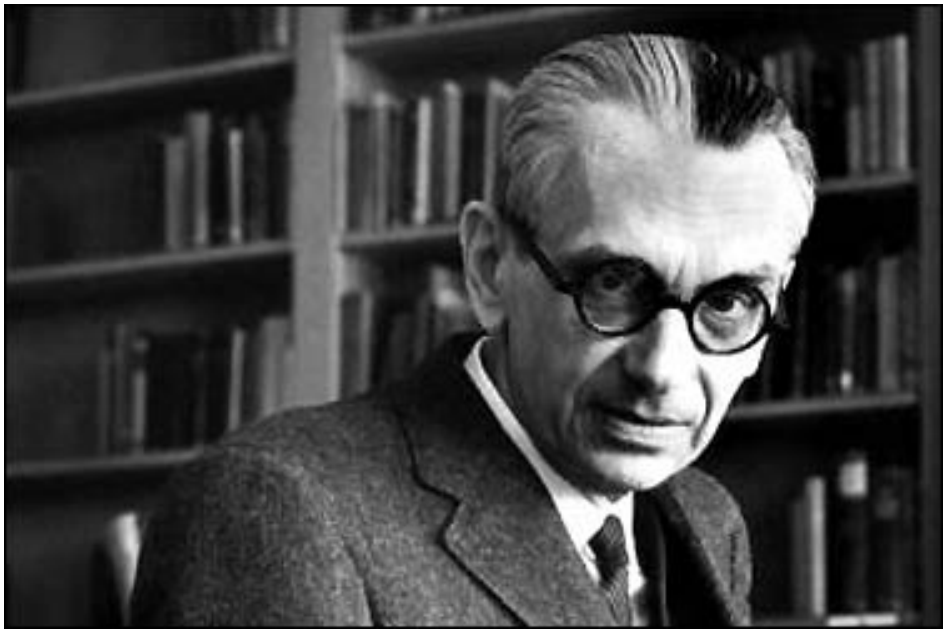
Problema della decisione (Hilbert-Ackermann, 1928):

Dato un enunciato dell' l'aritmetica, esiste **una procedura meccanica effettiva** capace di stabilire se l'enunciato in questione sia deducibile o meno dagli assiomi?

Modello di calcolo:

Definizione matematica rigorosa della nozione intuitiva di procedura meccanica di calcolo.

Modelli di calcolo: le funzioni ricorsive



Kurt Godel

Modelli di calcolo: le funzioni ricorsive

- Modello assiomatico: la classe delle funzioni ricorsive è la piu piccola classe contenente :

Modelli di calcolo: le funzioni ricorsive

- Modello assiomatico: la classe delle funzioni ricorsive è la piu piccola classe contenente :
  - alcune funzioni di base (funzioni zero, successore, proiezioni)

Modelli di calcolo: le funzioni ricorsive

- Modello assiomatico: la classe delle funzioni ricorsive è la piu piccola classe contenente :
  - alcune funzioni di base (funzioni zero, successore, proiezioni)
  - chiusa rispetto a certe operazioni (composizione, ricorsione...)

Modelli di calcolo: le funzioni ricorsive

- Modello assiomatico: la classe delle funzioni ricorsive è la piu piccola classe contenente :
  - alcune funzioni di base (funzioni zero, successore, proiezioni)
  - chiusa rispetto a certe operazioni (composizione, ricorsione...)
- Pro: Definizione matematicamente elegante.

Modelli di calcolo: le funzioni ricorsive

- Modello assiomatico: la classe delle funzioni ricorsive è la piu piccola classe contenente :
  - alcune funzioni di base (funzioni zero, successore, proiezioni)
  - chiusa rispetto a certe operazioni (composizione, ricorsione...)
- Pro: Definizione matematicamente elegante.
- Contro: manca una nozione esplicita di *passo di calcolo*

Modelli di calcolo: le macchine di Turing

Alan Turing

Modelli di calcolo: le macchine di Turing

- Modello "concreto": una macchina di Turing è costituita da :
  - un nastro infinito;

Modelli di calcolo: le macchine di Turing

- Modello "concreto": una macchina di Turing è costituita da :
  - un nastro infinito;
  - una testina in grado di leggere, scrivere e spostarsi sul nastro;

Modelli di calcolo: le macchine di Turing

- Modello "concreto": una macchina di Turing è costituita da :
  - un nastro infinito;
  - una testina in grado di leggere, scrivere e spostarsi sul nastro;
  - un insieme di istruzioni, che descrive come la macchina passa da un determinato stato ad un altro.

Modelli di calcolo: le macchine di Turing

- Modello "concreto": una macchina di Turing è costituita da :
  - un nastro infinito;
  - una testina in grado di leggere, scrivere e spostarsi sul nastro;
  - un insieme di istruzioni, che descrive come la macchina passa da un determinato stato ad un altro.
- Pro: nozione precisa di passo di calcolo, evidentemente meccanizzabile.

Modelli di calcolo: le macchine di Turing

- Modello "concreto": una macchina di Turing è costituita da :
  - un nastro infinito;
  - una testina in grado di leggere, scrivere e spostarsi sul nastro;
  - un insieme di istruzioni, che descrive come la macchina passa da un determinato stato ad un altro.
- Pro: nozione precisa di passo di calcolo, evidentemente meccanizzabile.
- Contro: Difficile dimostrare proprietà astratte in questo modello.

Modelli di calcolo: il λ-calcolo

Alonzo Church

Modelli di calcolo: il λ-calcolo

- Prende il meglio dei due mondi:

Modelli di calcolo: il λ-calcolo

- Prende il meglio dei due mondi:
- Definizione matematicamente elegante: si basa unicamente sulla nozione di funzione e di applicazione di funzioni.

Modelli di calcolo: il λ-calcolo

- Prende il meglio dei due mondi:
- Definizione matematicamente elegante: si basa unicamente sulla nozione di funzione e di applicazione di funzioni.
- Possiede una nozione esplicita di "passo di calcolo" (per quanto non così elementare come quella della macchina di Turing)

Modelli di calcolo: il λ-calcolo

- Prende il meglio dei due mondi:
- Definizione matematicamente elegante: si basa unicamente sulla nozione di funzione e di applicazione di funzioni.
- Possiede una nozione esplicita di "passo di calcolo" (per quanto non così elementare come quella della macchina di Turing)
- E' il nucleo di base di tutti i linguaggi di programmazione funzionali (Lisp, ML, etc)

Equivalenza dei modelli di calcolo

Teorema

Data una qualsiasi funzione f da interi a interi, f è Turing-calcolabile se e solo se è ricorsiva se e solo se è λ-calcolabile.

Equivalenza dei modelli di calcolo

Teorema

Data una qualsiasi funzione f da interi a interi, f è Turing-calcolabile se e solo se è ricorsiva se e solo se è λ-calcolabile.

Tesi di Church-Turing

Ogni funzione per cui esiste una procedura effettiva di calcolo è Turing-calcolabile.

Equivalenza dei modelli di calcolo

Teorema

Data una qualsiasi funzione f da interi a interi, f è Turing-calcolabile se e solo se è ricorsiva se e solo se è λ-calcolabile.

Tesi di Church-Turing

Ogni funzione per cui esiste una procedura effettiva di calcolo è Turing-calcolabile.

- La tesi non è dimostrabile (la nozione di procedura effettiva non è formale);

Equivalenza dei modelli di calcolo

Teorema

Data una qualsiasi funzione f da interi a interi, f è Turing-calcolabile se e solo se è ricorsiva se e solo se è λ-calcolabile.

Tesi di Church-Turing

Ogni funzione per cui esiste una procedura effettiva di calcolo è Turing-calcolabile.

- La tesi non è dimostrabile (la nozione di procedura effettiva non è formale);
- l'evidenza "empirica" conferma la tesi di Church-Turing.

E l'Entscheidungsproblem ?

Teorema di Church

Non esiste una macchina di Turing che dato un enunciato dell'aritmetica sia capace di determinare se esso sia o meno dimostrabile.

E l'Entscheidungsproblem ?

Teorema di Church

Non esiste una macchina di Turing che dato un enunciato dell'aritmetica sia capace di determinare se esso sia o meno dimostrabile.

Problema della fermata (Turing)

Non esiste una macchina di Turing capace di decidere se, a partire da un input arbitrario, un programma termini.

E l'Entscheidungsproblem ?

Teorema di Church

Non esiste una macchina di Turing che dato un enunciato dell'aritmetica sia capace di determinare se esso sia o meno dimostrabile.

Problema della fermata (Turing)

Non esiste una macchina di Turing capace di decidere se, a partire da un input arbitrario, un programma termini.

Secondo teorema di incompletezza (Godel)

Non è possibile dimostrare la coerenza dell'aritmetica con metodi finiti.

Nascita dei moderni calcolatori

John Von Neumann

A volte i problemi sono più interessanti delle soluzioni....

I know that in or about 1943 or '44 von Neumann was well aware of the fundamental importance of Turing's paper of 1936 ... Von Neumann introduced me to that paper and at his urging I studied it with care. Many people have acclaimed von Neumann as the "father of the computer" (in a modern sense of the term) but I am sure that he would never have made that mistake himself. He might well be called the midwife, perhaps, but he firmly emphasized to me, and to others I am sure, that the fundamental conception is owing to Turing.

Stan Frankel, collega di Von Neumann a Los Alamos.

Indice

Cosa è una funzione? Il punto di vista estensionale

- Una funzione è una "legge" che associa ad ogni elemento del dominio di f (l'insieme dei valori di input X) un elemento del codominio di f (l'insieme dei valori di output Y)

Cosa è una funzione? Il punto di vista estensionale

- Una funzione è una "legge" che associa ad ogni elemento del dominio di f (l'insieme dei valori di input X) un elemento del codominio di f (l'insieme dei valori di output Y)
- Piu formalmente, una funzione è un sottoinsieme del prodotto cartesiano $X \times Y$ tale che per ogni $x \in X$ esiste un unico $y \in Y$ tale che $(x, y) \in f$.

Cosa è una funzione? Il punto di vista estensionale

- Una funzione è una "legge" che associa ad ogni elemento del dominio di f (l'insieme dei valori di input X) un elemento del codominio di f (l'insieme dei valori di output Y)
- Piu formalmente, una funzione è un sottoinsieme del prodotto cartesiano $X \times Y$ tale che per ogni $x \in X$ esiste un unico $y \in Y$ tale che $(x, y) \in f$.
- L'esistenza e l'unicità rendono possibile la notazione $f(x) = y$; ogni elemento del dominio di f determina in modo univoco un elemento del codominio di f.

Cosa è una funzione? Il punto di vista estensionale

- Una funzione dal punto di vista estensionale è completamente definita dal suo grafico : $f = \{(x, f(x)) : x \in X\} \subseteq X \times Y$.

Cosa è una funzione? Il punto di vista estensionale

- Una funzione dal punto di vista estensionale è completamente definita dal suo grafico : $f = \{(x, f(x)) : x \in X\} \subseteq X \times Y$.
- Dal punto di vista estensionale, non siamo interessati a sapere in che modo la funzione calcola i suoi valori; ci interessa solo conoscere i valori in entrata, i valori in uscita e la relazione tra di essi (chi corrisponde a chi).

Cosa è una funzione? Il punto di vista estensionale

- Una funzione dal punto di vista estensionale è completamente definita dal suo grafico : $f = \{(x, f(x)) : x \in X\} \subseteq X \times Y$.
- Dal punto di vista estensionale, non siamo interessati a sapere in che modo la funzione calcola i suoi valori; ci interessa solo conoscere i valori in entrata, i valori in uscita e la relazione tra di essi (chi corrisponde a chi).
- Es. Le funzioni $Sum(x, y) = x + y$ e $Mus(x, y) = y + x$ sono estensionalmente equivalenti (anche se calcolano i loro valori in maniera differente).

Cosa è una funzione? Il punto di vista intensionale

- Una funzione f è una procedura effettiva che prende in input un $x \in X$ e produce un output $y \in Y$ in un numero finito di passi elementari.

Cosa è una funzione? Il punto di vista intensionale

- Una funzione f è una procedura effettiva che prende in input un $x \in X$ e produce un output $y \in Y$ in un numero finito di passi elementari.
- Dal punto di vista intensionale, siamo interessati a capire in che modo la funzione f produce il suo risultato: non ci basta conoscere il risultato.

Cosa è una funzione? Il punto di vista intensionale

- Una funzione f è una procedura effettiva che prende in input un $x \in X$ e produce un output $y \in Y$ in un numero finito di passi elementari.
- Dal punto di vista intensionale, siamo interessati a capire in che modo la funzione f produce il suo risultato: non ci basta conoscere il risultato.
- Es. Le funzioni $Sum(x, y) = x + y$ e $Mus(x, y) = y + x$ sono intensionalmente differenti (anche se estensionalmente si comportano allo stesso modo).

La λ-notazione

$$f(x) = t$$

La λ-notazione

$$f(x) = t$$

La funzione f *dove* $f(x) = t$.

La λ-notazione

$$f(x) = x * x$$

La λ-notazione

$$f(x) = x * x$$

La funzione f *dove* $f(x) = x * x$.

La λ-notazione

$$f(x) = x * x$$

La funzione f *dove* $f(x) = x * x$.

$$f(3) = 3 * 3 = 9$$

La λ-notazione

$$f(x) = x * x$$

La funzione f dove $f(x) = x * x$.

$$f(3) = 3 * 3 = 9$$

Calcolare = sostituire (e rendere esplicito)

La λ-notazione

$$f(x) = t$$

La λ-notazione

$$f(x) = t$$

La funzione f *dove* $f(x) = t$.

La λ-notazione

$$f(x) = t$$

La funzione f *dove* $f(x) = t$.

$$\lambda x.t$$

La λ-notazione

$$f(x) = t$$

La funzione f dove $f(x) = t$.

$$\lambda x.t$$

passo di calcolo: $(\lambda x.t)(u) \Rightarrow t[u/x]$

La λ-notazione

$$f(x) = t$$

La funzione f dove $f(x) = t$.

$$\lambda x.t$$

passo di calcolo: $(\lambda x.t)(u) \Rightarrow t[u/x]$

$t[u/x]$ è il termine ottenuto rimpiazzando in t ogni occorrenza libera di x con u.

La λ-notazione

$$f(x) = x * x$$

La λ-notazione

$$f(x) = x * x$$

La funzione f *dove* $f(x) = x * x$.

La λ-notazione

$$f(x) = x * x$$

La funzione f *dove* $f(x) = x * x$.

$$\lambda x.x * x$$

La λ-notazione

$$f(x) = x * x$$

La funzione f *dove* $f(x) = x * x$.

$$\lambda x.x * x$$

passo di calcolo: $(\lambda x.x * x)(3) \Rightarrow x * x[3/x]$

La λ-notazione

$$f(x) = x * x$$

La funzione f *dove* $f(x) = x * x$.

$$\lambda x.x * x$$

passo di calcolo: $(\lambda x.x * x)(3) \Rightarrow x * x[3/x]$

$$x * x[3/x] = 3 * 3 = 9$$

La λ-notazione

$$f(y) = y + 1$$

La λ-notazione

$$f(y) = y + 1$$

La funzione f *dove* $f(y) = y + 1$.

La λ-notazione

$$f(y) = y + 1$$

La funzione f dove $f(y) = y + 1$.

$$\lambda y.y + 1$$

La λ-notazione

$$f(y) = y + 1$$

La funzione f dove $f(y) = y + 1$.

$$\lambda y.y + 1$$

passo di calcolo: $(\lambda y.y + 1)(3) \Rightarrow y + 1[3/y]$

La λ-notazione

$$f(y) = y + 1$$

La funzione f dove $f(y) = y + 1$.

$$\lambda y.y + 1$$

passo di calcolo: $(\lambda y.y + 1)(3) \Rightarrow y + 1[3/y]$

$$y + 1[3/y] = 3 + 1 = 4$$

La λ-notazione

$$f(z) = z + 1$$

La λ-notazione

$$f(z) = z + 1$$

La funzione f *dove* $f(z) = z + 1$.

La λ-notazione

$$f(z) = z + 1$$

La funzione f dove $f(z) = z + 1$.

$$\lambda z.z + 1$$

La λ-notazione

$$f(z) = z + 1$$

La funzione f dove $f(z) = z + 1$.

$$\lambda z.z + 1$$

passo di calcolo: $(\lambda z.z + 1)(3) \Rightarrow z + 1[3/z]$

La λ-notazione

$$f(z) = z + 1$$

La funzione f dove $f(z) = z + 1$.

$$\lambda z.z + 1$$

passo di calcolo: $(\lambda z.z + 1)(3) \Rightarrow z + 1[3/z]$

$$z + 1[3/z] = 3 + 1 = 4$$

La λ-notazione

$$f(z) = z + 1$$

La funzione f dove $f(z) = z + 1$.

$$\lambda z.z + 1$$

passo di calcolo: $(\lambda z.z + 1)(3) \Rightarrow z + 1[3/z]$

$$z + 1[3/z] = 3 + 1 = 4$$

$$\lambda z.z + 1 \equiv \lambda y.y + 1$$

E quando ci sono piu riduzioni?

$$(\lambda x.x * x)((\lambda y.y + 1)(2))$$

E quando ci sono piu riduzioni?

$$(\lambda x.x * x)((\lambda y.y + 1)(2))$$

E quando ci sono piu riduzioni?

$$(\lambda x.x * x)((\lambda y.y + 1)(2))$$

$\swarrow$

$$(\lambda x.x * x)(2 + 1)$$

$\searrow$

$$((\lambda y.y + 1)(2)) * ((\lambda y.y + 1)(2))$$

E quando ci sono piu riduzioni?

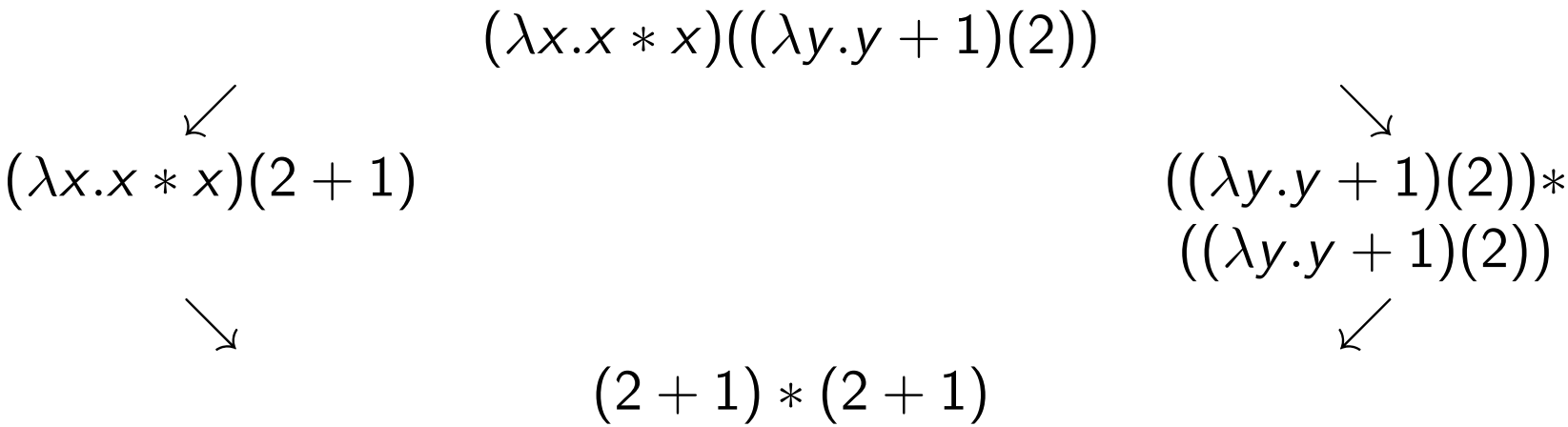
$$(\lambda x.x * x)((\lambda y.y + 1)(2))$$

$$\swarrow \qquad \searrow$$

$$(\lambda x.x * x)(2 + 1) \qquad ((\lambda y.y + 1)(2)) * ((\lambda y.y + 1)(2))$$

$$\searrow \qquad \swarrow$$

E quando ci sono piu riduzioni?



E quando ci sono più argomenti?

$$g(x, y) = x * x + y * y$$

E quando ci sono più argomenti?

$$g(x, y) = x * x + y * y$$

Currificazione: Una funzione a due argomenti può essere rappresentata da una funzione sul primo argomento che restituisce una funzione sul secondo argomento

E quando ci sono più argomenti?

$$g(x, y) = x * x + y * y$$

Currificazione: Una funzione a due argomenti può essere rappresentata da una funzione sul primo argomento che restituisce una funzione sul secondo argomento

$$g(3, 4) = 3 * 3 + 4 * 4 = 25$$

E quando ci sono più argomenti?

$$g(x, y) = x * x + y * y$$

Currificazione: Una funzione a due argomenti può essere rappresentata da una funzione sul primo argomento che restituisce una funzione sul secondo argomento

$$g(3, 4) = 3 * 3 + 4 * 4 = 25$$

$$((\lambda x.\lambda y.x * x + y * y)(3))(4)$$

E quando ci sono più argomenti?

$$g(x, y) = x * x + y * y$$

Currificazione: Una funzione a due argomenti può essere rappresentata da una funzione sul primo argomento che restituisce una funzione sul secondo argomento

$$g(3, 4) = 3 * 3 + 4 * 4 = 25$$

$$\begin{array}{rc} & ((\lambda x.\lambda y.x * x + y * y)(3))(4) \\ \Rightarrow & (\lambda y.3 * 3 + y * y)(4) \end{array}$$

E quando ci sono più argomenti?

$$g(x, y) = x * x + y * y$$

Currificazione: Una funzione a due argomenti può essere rappresentata da una funzione sul primo argomento che restituisce una funzione sul secondo argomento

$$g(3, 4) = 3 * 3 + 4 * 4 = 25$$

$$\begin{aligned} & ((\lambda x.\lambda y.x * x + y * y)(3))(4) \\ \Rightarrow \quad & (\lambda y.3 * 3 + y * y)(4) \\ \Rightarrow \quad & 3 * 3 + 4 * 4 \end{aligned}$$

E quando ci sono più argomenti?

$$g(x, y) = x * x + y * y$$

Currificazione: Una funzione a due argomenti può essere rappresentata da una funzione sul primo argomento che restituisce una funzione sul secondo argomento

$$g(3, 4) = 3 * 3 + 4 * 4 = 25$$

$$\begin{aligned} & ((\lambda x.\lambda y.x * x + y * y)(3))(4) \\ \Rightarrow \quad & (\lambda y.3 * 3 + y * y)(4) \\ \Rightarrow \quad & 3 * 3 + 4 * 4 \\ \Rightarrow \quad & 25 \end{aligned}$$

Composizione di funzioni

$$g(f(z))$$

Composizione di funzioni

$$g(f(z))$$

Composizione: *Una funzione che prende due funzioni f e g come argomenti e restituisce una funzione che prende un argomento z, applica la prima funzione all'argomento, e applica la seconda funzione al risultato*

Composizione di funzioni

$$g(f(z))$$

Composizione: *Una funzione che prende due funzioni f e g come argomenti e restituisce una funzione che prende un argomento z, applica la prima funzione all'argomento, e applica la seconda funzione al risultato*

$$\lambda f.\lambda g.\lambda z.g(f(z))$$

Composizione di funzioni

$$(\lambda f.\lambda g.\lambda z.g(f(z)))(\lambda y.y+1)(\lambda x.x*x)(2)$$

Composizione di funzioni

$$\begin{array}{rc} & (\lambda f.\lambda g.\lambda z.g(f(z)))(\lambda y.y+1)(\lambda x.x*x)(2) \\ \Rightarrow & (\lambda g.\lambda z.g((\lambda y.y+1)(z)))(\lambda x.x*x)(2) \end{array}$$

Composizione di funzioni

$$
\begin{aligned}
& (\lambda f.\lambda g.\lambda z.g(f(z)))(\lambda y.y+1)(\lambda x.x*x)(2) \\
\Rightarrow \quad & (\lambda g.\lambda z.g((\lambda y.y+1)(z)))(\lambda x.x*x)(2) \\
\Rightarrow \quad & (\lambda z.(\lambda x.x*x)((\lambda y.y+1)(z)))(2)
\end{aligned}
$$

Composizione di funzioni

$$
\begin{aligned}
& (\lambda f.\lambda g.\lambda z.g(f(z)))(\lambda y.y+1)(\lambda x.x*x)(2) \\
\Rightarrow\quad & (\lambda g.\lambda z.g((\lambda y.y+1)(z)))(\lambda x.x*x)(2) \\
\Rightarrow\quad & (\lambda z.(\lambda x.x*x)((\lambda y.y+1)(z)))(2) \\
\Rightarrow\quad & (\lambda x.x*x)((\lambda y.y+1)(2))
\end{aligned}
$$

Composizione di funzioni

$$
\begin{aligned}
& (\lambda f.\lambda g.\lambda z.g(f(z)))(\lambda y.y+1)(\lambda x.x*x)(2) \\
\Rightarrow\ & (\lambda g.\lambda z.g((\lambda y.y+1)(z)))(\lambda x.x*x)(2) \\
\Rightarrow\ & (\lambda z.(\lambda x.x*x)((\lambda y.y+1)(z)))(2) \\
\Rightarrow\ & (\lambda x.x*x)((\lambda y.y+1)(2)) \\
\Rightarrow\ & (2+1)*(2+1)
\end{aligned}
$$

Composizione di funzioni

$$
\begin{aligned}
& (\lambda f.\lambda g.\lambda z.g(f(z)))(\lambda y.y+1)(\lambda x.x*x)(2) \\
\Rightarrow\ & (\lambda g.\lambda z.g((\lambda y.y+1)(z)))(\lambda x.x*x)(2) \\
\Rightarrow\ & (\lambda z.(\lambda x.x*x)((\lambda y.y+1)(z)))(2) \\
\Rightarrow\ & (\lambda x.x*x)((\lambda y.y+1)(2)) \\
\Rightarrow\ & (2+1)*(2+1) \\
\Rightarrow\ & 9
\end{aligned}
$$

Composizione di funzioni

$$
\begin{aligned}
& (\lambda f.\lambda g.\lambda z.g(f(z)))(\lambda y.y+1)(\lambda x.x*x)(2) \\
\Rightarrow\ & (\lambda g.\lambda z.g((\lambda y.y+1)(z)))(\lambda x.x*x)(2) \\
\Rightarrow\ & (\lambda z.(\lambda x.x*x)((\lambda y.y+1)(z)))(2) \\
\Rightarrow\ & (\lambda x.x*x)((\lambda y.y+1)(2)) \\
\Rightarrow\ & (2+1)*(2+1) \\
\Rightarrow\ & 9
\end{aligned}
$$

Una funzione può prendere in argomento funzioni e restituire come valore una funzione

λ-calcolo, in due parole

- λ-calcolo: linguaggio di programmazione funzionale;
- termini del λ-calcolo: programmi;
- regola di riduzione: passo di calcolo (esecuzione del programma)

Tutto è una funzione

Sintassi del λ-calcolo

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM$$

Sintassi del λ-calcolo

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM$$

Un λ-termine M è :

Sintassi del λ-calcolo

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM$$

Un λ-termine M è :

- una variabile x presa da un insieme infinito numerabile $\mathcal{V}$ di variabili (denoteremo le variabili con $x, y, z, \ldots$);

Sintassi del λ-calcolo

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM$$

Un λ-termine M è :

- una variabile x presa da un insieme infinito numerabile $\mathcal{V}$ di variabili (denoteremo le variabili con $x, y, z, \ldots$);
- oppure una λ-astrazione $\lambda x.M$;

Sintassi del λ-calcolo

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM$$

Un λ-termine M è :

- una variabile x presa da un insieme infinito numerabile $\mathcal{V}$ di variabili (denoteremo le variabili con $x, y, z, \ldots$);
- oppure una λ-astrazione $\lambda x.M$;
- oppure una applicazione MM di un λ-termine ad un altro.

Sintassi del λ-calcolo

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM$$

Un λ-termine M è :

- una variabile x presa da un insieme infinito numerabile $\mathcal{V}$ di variabili (denoteremo le variabili con $x, y, z, \ldots$);
- oppure una λ-astrazione $\lambda x.M$;
- oppure una applicazione MM di un λ-termine ad un altro.

L'applicazione associa a sinistra: $M_1 M_2 M3 = ((M_1 M_2) M_3)$

α-equivalenza

$$M = (\lambda x.yx)x$$

α-equivalenza

$$M = (\lambda x.yx)x$$

- x è *vincolata* in M; x, y sono *libere* in M.

α-equivalenza

$$M = (\lambda x.yx)x$$

- x è *vincolata* in M; x, y sono *libere* in M.
- *le variabili vincolate sono mute.*

α-equivalenza

$$M = (\lambda x.yx)x$$

- x è *vincolata* in M; x, y sono *libere* in M.
- *le variabili vincolate sono mute.*

$$\lambda x.M \equiv_\alpha \lambda y.(M[y/x])$$

α-equivalenza

$$M = (\lambda x.yx)x$$

- x è *vincolata* in M; x, y sono *libere* in M.
- *le variabili vincolate sono mute.*

$$\lambda x.M \equiv_\alpha \lambda y.(M[y/x])$$

Es. $\lambda x.(xy) \equiv_\alpha \lambda z.(zy)$

α-equivalenza

$$M = (\lambda x.yx)x$$

- x è *vincolata* in M; x, y sono *libere* in M.
- *le variabili vincolate sono mute.*

$$\lambda x.M \equiv_\alpha \lambda y.(M[y/x])$$

Es. $\lambda x.(xy) \equiv_\alpha \lambda z.(zy)$

Due funzioni α-equivalenti rappresentano la stessa funzione.

β-riduzione

$$(\lambda x.M)N \rightarrow_\beta M[N/x]$$

β-riduzione

$$(\lambda x.M)N \to_\beta M[N/x]$$

- la sostituzione è definita modulo α-equivalenza;

β-riduzione

$$(\lambda x.M)N \to_\beta M[N/x]$$

- la sostituzione è definita modulo α-equivalenza;
- $(\lambda x.M)N$ è chiamato *redesso*, $M[N/x]$ è chiamato *ridotto*.

β-riduzione

$$(\lambda x.M)N \to_\beta M[N/x]$$

- la sostituzione è definita modulo α-equivalenza;
- $(\lambda x.M)N$ è chiamato *redesso*, $M[N/x]$ è chiamato *ridotto*.
- Se $M \to_\beta M'$ allora si dice che M *β-riduce in un passo* a M'.

β-riduzione

$$(\lambda x.M)N \to_\beta M[N/x]$$

- la sostituzione è definita modulo α-equivalenza;
- $(\lambda x.M)N$ è chiamato *redesso*, $M[N/x]$ è chiamato *ridotto*.
- Se $M \to_\beta M'$ allora si dice che M *β-riduce in un passo* a M'.
- Es. $(\lambda x.xx)(\lambda y.y) \to_\beta (\lambda y.y)(\lambda y.y)$

β-riduzione

$$(\lambda x.M)N \to_\beta M[N/x]$$

- la sostituzione è definita modulo α-equivalenza;
- $(\lambda x.M)N$ è chiamato *redesso*, $M[N/x]$ è chiamato *ridotto*.
- Se $M \to_\beta M'$ allora si dice che M *β-riduce in un passo* a M'.
- Es. $(\lambda x.xx)(\lambda y.y) \to_\beta (\lambda y.y)(\lambda y.y)$
- Es. $(\lambda x.z)(\lambda y.y) \to_\beta z$

β-riduzione

$$(\lambda x.M)N \to_\beta M[N/x]$$

- la sostituzione è definita modulo α-equivalenza;
- $(\lambda x.M)N$ è chiamato *redesso*, $M[N/x]$ è chiamato *ridotto*.
- Se $M \to_\beta M'$ allora si dice che M *β-riduce in un passo* a M'.
- Es. $(\lambda x.xx)(\lambda y.y) \to_\beta (\lambda y.y)(\lambda y.y)$
- Es. $(\lambda x.z)(\lambda y.y) \to_\beta z$
- nello stesso termine possono comparire piu redessi. Es: $(\lambda x.z)((\lambda y.y)(z))$.

β-riduzione

La *β-riduzione* (denotata $\rightarrow_\beta^*$) è la chiusura riflessiva e transitiva della relazione $\rightarrow_\beta$.

β-riduzione

La *β-riduzione* (denotata $\rightarrow_\beta^*$) è la chiusura riflessiva e transitiva della relazione $\rightarrow_\beta$.

- $M \rightarrow_\beta^* M'$ se e solo se se esiste una successione $M_0, M_1, \ldots, M_{n-1}, M_n$ tale che $M = M_0, M' = M_n$, e $M_i \rightarrow_\beta M_{i+1}$ per $1 \leq i \leq n-1$ (con $n \geq 0$).

β-riduzione

La *β-riduzione* (denotata $\rightarrow_\beta^*$) è la chiusura riflessiva e transitiva della relazione $\rightarrow_\beta$.

- $M \rightarrow_\beta^* M'$ se e solo se se esiste una successione $M_0, M_1, \ldots, M_{n-1}, M_n$ tale che $M = M_0, M' = M_n$, e $M_i \rightarrow_\beta M_{i+1}$ per $1 \leq i \leq n-1$ (con $n \geq 0$).
- un λ-termine M è detto *in forma normale* se e solo se non contiene redessi, i.e. non esiste nessun λ-termine M' tale che $M \rightarrow_\beta^* M'$.

Il λ-calcolo come linguaggio di programmazione: un intuizione

- $(\lambda x_1 \ldots \lambda x_n.M)N_1 \ldots N_n$ corrisponde ad un programma M che attende n argomenti, applicato agli input $N_1, \ldots, N_n$;

Il λ-calcolo come linguaggio di programmazione: un intuizione

- $(\lambda x_1 \ldots \lambda x_n.M)N_1 \ldots N_n$ corrisponde ad un programma M che attende n argomenti, applicato agli input $N_1, \ldots, N_n$;
- La β-riduzione corrisponde all'esecuzione di un programma: se $M \to_\beta^* M'$ allora M' rappresenta uno stadio più avanzato nell'esecuzione del programma M;

Il λ-calcolo come linguaggio di programmazione: un intuizione

- $(\lambda x_1 \ldots \lambda x_n.M)N_1 \ldots N_n$ corrisponde ad un programma M che attende n argomenti, applicato agli input $N_1, \ldots, N_n$;
- La β-riduzione corrisponde all'esecuzione di un programma: se $M \to_\beta^* M'$ allora M' rappresenta uno stadio più avanzato nell'esecuzione del programma M;
- un λ-termine in forma normale corrisponde all'output del programma.

L'η-equivalenza

$\lambda x.(Mx) \equiv_\eta M$, se x non occorre libera in M.

L'η-equivalenza

$\lambda x.(Mx) \equiv_\eta M$, se x non occorre libera in M.

- Due λ-termini M, M' sono η-equivalenti (o *estensionalmente equivalenti*) se si comportano allo stesso modo, una volta applicati ad un argomento.

L'η-equivalenza

$\lambda x.(Mx) \equiv_\eta M$, se x non occorre libera in M.

- Due λ-termini M, M' sono η-equivalenti (o *estensionalmente equivalenti*) se si comportano allo stesso modo, una volta applicati ad un argomento.
- Per ogni N, $(\lambda x.(Mx))N \to_\beta (Mx)[N/x] = MN$

Normalizzazione debole e forte

- un λ-termine è detto *normalizzabile* (o debolmente normalizzabile) se esiste un λ-termine M' normale tale che $M \to_\beta^* M'$; si dice allora che M' è una forma normale di M.

Normalizzazione debole e forte

- un λ-termine è detto *normalizzabile* (o debolmente normalizzabile) se esiste un λ-termine M' normale tale che $M \to_\beta^* M'$; si dice allora che M' è una forma normale di M.
- un λ-termine è detto *fortemente normalizzabile* se non esiste alcuna successione infinita $M_{i \in \mathbb{N}}$ tale che $M = M_0$ e $M_i \to_\beta M_{i+1}$ per ogni $i \in \mathbb{N}$.

Normalizzazione debole e forte

- un λ-termine è detto *normalizzabile* (o debolmente normalizzabile) se esiste un λ-termine M' normale tale che $M \to_\beta^* M'$; si dice allora che M' è una forma normale di M.
- un λ-termine è detto *fortemente normalizzabile* se non esiste alcuna successione infinita $M_{i\in\mathbb{N}}$ tale che $M = M_0$ e $M_i \to_\beta M_{i+1}$ per ogni $i \in \mathbb{N}$.
- un λ-termine non normalizzabile corrisponde ad un programma che cicla all'infinito.

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.x)$ è fortemente normalizzabile: l'unica sequenza di riduzioni possibile termina.

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.x)$ è fortemente normalizzabile: l'unica sequenza di riduzioni possibile termina.

$$(\lambda x.xx)(\lambda x.x)$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.x)$ è fortemente normalizzabile: l'unica sequenza di riduzioni possibile termina.

$$\begin{array}{ll} & (\lambda x.xx)(\lambda x.x) \\ \rightarrow_\beta & x[\lambda x.x/x]x[\lambda x.x/x] \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.x)$ è fortemente normalizzabile: l'unica sequenza di riduzioni possibile termina.

$$\begin{array}{ll} & (\lambda x.xx)(\lambda x.x) \\ \rightarrow_\beta & x[\lambda x.x/x]x[\lambda x.x/x] \\ \rightarrow_\beta & (\lambda x.x)(\lambda x.x) \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.x)$ è fortemente normalizzabile: l'unica sequenza di riduzioni possibile termina.

$$\begin{array}{ll} & (\lambda x.xx)(\lambda x.x) \\ \rightarrow_\beta & x[\lambda x.x/x]x[\lambda x.x/x] \\ \rightarrow_\beta & (\lambda x.x)(\lambda x.x) \\ \rightarrow_\beta & x[\lambda x.x/x] \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.x)$ è fortemente normalizzabile: l'unica sequenza di riduzioni possibile termina.

$$\begin{array}{cc} & (\lambda x.xx)(\lambda x.x) \\ \rightarrow_\beta & x[\lambda x.x/x]x[\lambda x.x/x] \\ \rightarrow_\beta & (\lambda x.x)(\lambda x.x) \\ \rightarrow_\beta & x[\lambda x.x/x] \\ = & \lambda x.x \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.xx)$ non è normalizzabile: l'unica sequenza di riduzioni possibile non termina.

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.xx)$ non è normalizzabile: l'unica sequenza di riduzioni possibile non termina.

$$(\lambda x.xx)(\lambda x.xx)$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.xx)$ non è normalizzabile: l'unica sequenza di riduzioni possibile non termina.

$$\begin{array}{ll} & (\lambda x.xx)(\lambda x.xx) \\ \rightarrow_\beta & x[\lambda x.xx/x]x[\lambda x.xx/x] \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.xx)$ non è normalizzabile: l'unica sequenza di riduzioni possibile non termina.

$$\begin{array}{ll} & (\lambda x.xx)(\lambda x.xx) \\ \rightarrow_\beta & x[\lambda x.xx/x]x[\lambda x.xx/x] \\ \rightarrow_\beta & (\lambda x.xx)(\lambda x.xx) \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda x.xx)(\lambda x.xx)$ non è normalizzabile: l'unica sequenza di riduzioni possibile non termina.

$$\begin{array}{ll} & (\lambda x.xx)(\lambda x.xx) \\ \rightarrow_\beta & x[\lambda x.xx/x]x[\lambda x.xx/x] \\ \rightarrow_\beta & (\lambda x.xx)(\lambda x.xx) \\ \rightarrow_\beta & \vdots \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

$$(\lambda z.y)((\lambda x.xx)\lambda x.xx)$$

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

$$\begin{array}{ll} & (\lambda z.y)((\lambda x.xx)\lambda x.xx) \\ \rightarrow_\beta & y[(\lambda x.xx)\lambda x.xx/z] \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

$$\begin{array}{cc} & (\lambda z.y)((\lambda x.xx)\lambda x.xx) \\ \rightarrow_\beta & y[(\lambda x.xx)\lambda x.xx/z] \\ = & y \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

$$(\lambda z.y)((\lambda x.xx)\lambda x.xx)$$

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

$$\begin{array}{rc} & (\lambda z.y)((\lambda x.xx)\lambda x.xx) \\ \rightarrow_\beta & (\lambda z.y)(x[\lambda x.xx/x]x[\lambda x.xx/x]) \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

$$\begin{array}{ll} & (\lambda z.y)((\lambda x.xx)\lambda x.xx) \\ \rightarrow_\beta & (\lambda z.y)(x[\lambda x.xx/x]x[\lambda x.xx/x]) \\ \rightarrow_\beta & (\lambda z.y)((\lambda x.xx)\lambda x.xx) \end{array}$$

Normalizzazione debole e forte: esempi

- $(\lambda z.y)((\lambda x.xx)\lambda x.xx)$ è normalizzabile, ci sono sequenze di riduzioni che terminano e altre no.

$$
\begin{array}{ll}
 & (\lambda z.y)((\lambda x.xx)\lambda x.xx) \\
\to_\beta & (\lambda z.y)(x[\lambda x.xx/x]x[\lambda x.xx/x]) \\
\to_\beta & (\lambda z.y)((\lambda x.xx)\lambda x.xx) \\
 & \vdots \\
\to_\beta &
\end{array}
$$

Teorema di Church-Rosser

Un λ-termine ha un unica forma normale?

Teorema di Church-Rosser

Un λ-termine ha un unica forma normale?

Church-Rosser

Per ogni λ-termine M, M_1, M_2 tali che $M \to_\beta^* M_1$ e $M \to_\beta^* M_2$ esiste un λ-termine N tale che $M_1 \to_\beta^* N$ ed $M_2 \to_\beta^* N$

Teorema di Church-Rosser

Un λ-termine ha un unica forma normale?

Church-Rosser

Per ogni λ-termine M, M_1, M_2 tali che $M \to_\beta^* M_1$ e $M \to_\beta^* M_2$ esiste un λ-termine N tale che $M_1 \to_\beta^* N$ ed $M_2 \to_\beta^* N$

Corollario

Per ogni λ-termine M, se M_1, M_2 sono due forme normali di M allora $M_1 = M_2$.

Teorema di Church-Rosser

Un λ-termine ha un unica forma normale?

Church-Rosser

Per ogni λ-termine M, M_1, M_2 tali che $M \to_\beta^* M_1$ e $M \to_\beta^* M_2$ esiste un λ-termine N tale che $M_1 \to_\beta^* N$ ed $M_2 \to_\beta^* N$

Corollario

Per ogni λ-termine M, se M_1, M_2 sono due forme normali di M allora $M_1 = M_2$.

Dim. $M \to_\beta^* M_1$ e $M \to_\beta^* M_2$ quindi per la confluenza esiste N tale che $M_1 \to_\beta^* N$ ed $M_2 \to_\beta^* N$; ora poiche M_1, M_2 sono due forme normali, $M_1 = N = M_2$.

Programmare con il λ-calcolo

- Funzione identità: $\lambda x.x$

$$(\lambda x.x)M$$

Programmare con il λ-calcolo

- Funzione identità: $\lambda x.x$

$$\begin{array}{ll} & (\lambda x.x)M \\ \rightarrow_\beta & x[M/x] \end{array}$$

Programmare con il λ-calcolo

- Funzione identità: $\lambda x.x$

$$\begin{array}{cc} & (\lambda x.x)M \\ \rightarrow_\beta & x[M/x] \\ = & M \end{array}$$

Programmare con il λ-calcolo

- Funzione costante: $\lambda x.N$ dove x non è libera in N.

$$(\lambda x.N)M$$

Programmare con il λ-calcolo

- Funzione costante: $\lambda x.N$ dove x non è libera in N.

$$\begin{array}{ll} & (\lambda x.N)M \\ \rightarrow_\beta & N[M/x] \end{array}$$

Programmare con il λ-calcolo

- Funzione costante: $\lambda x.N$ dove x non è libera in N.

$$\begin{array}{cc} & (\lambda x.N)M \\ \rightarrow_\beta & N[M/x] \\ = & N \end{array}$$

Programmare con il λ-calcolo

- Funzione proiezione i-esima: $\lambda x_1 \ldots \lambda x_n.x_i$ (con $1 \leq i \leq n$).

$$(\lambda x_1 \ldots \lambda x_n.x_i) M_1 \ldots M_n$$

Programmare con il λ-calcolo

- Funzione proiezione *i*-esima: $\lambda x_1 \ldots \lambda x_n.x_i$ (con $1 \leq i \leq n$).

$$\begin{array}{rc} & (\lambda x_1 \ldots \lambda x_n.x_i)M_1 \ldots M_n \\ \rightarrow_\beta^* & (\lambda x_{i+1} \ldots \lambda x_n.M_i)M_{i+1} \ldots M_n \end{array}$$

Programmare con il λ-calcolo

- Funzione proiezione *i*-esima: $\lambda x_1 \ldots \lambda x_n.x_i$ (con $1 \leq i \leq n$).

$$\begin{array}{ll} & (\lambda x_1 \ldots \lambda x_n.x_i)M_1 \ldots M_n \\ \rightarrow_\beta^* & (\lambda x_{i+1} \ldots \lambda x_n.M_i)M_{i+1} \ldots M_n \\ \rightarrow_\beta^* & M_i \end{array}$$

Programmare con il λ-calcolo

- `True`: $\lambda x \lambda y.x$

$$(\lambda x \lambda y.x)MN$$

Programmare con il λ-calcolo

- `True`: $\lambda x \lambda y.x$

$$\begin{array}{ll} & (\lambda x \lambda y.x) M N \\ \rightarrow_\beta & (\lambda y.M) N \end{array}$$

Programmare con il λ-calcolo

- `True`: $\lambda x \lambda y.x$

$$\begin{array}{ll} & (\lambda x \lambda y.x) M N \\ \to_\beta & (\lambda y.M) N \\ \to_\beta & M[N/y] \end{array}$$

Programmare con il λ-calcolo

- `True`: $\lambda x \lambda y.x$

$$\begin{array}{cc} & (\lambda x \lambda y.x) M N \\ \rightarrow_\beta & (\lambda y.M) N \\ \rightarrow_\beta & M[N/y] \\ = & M \end{array}$$

Programmare con il λ-calcolo

- `False`: $\lambda x \lambda y.y$

$$(\lambda x \lambda y.y)MN$$

Programmare con il λ-calcolo

- `False`: $\lambda x \lambda y.y$

$$\begin{array}{ll} & (\lambda x \lambda y.y) M N \\ \rightarrow_\beta & (\lambda y.y) N \end{array}$$

Programmare con il λ-calcolo

- `False`: $\lambda x \lambda y.y$

$$\begin{array}{ll} & (\lambda x \lambda y.y)MN \\ \to_\beta & (\lambda y.y)N \\ \to_\beta & y[N/y] \end{array}$$

Programmare con il λ-calcolo

- `False`: $\lambda x \lambda y.y$

$$\begin{array}{cc} & (\lambda x \lambda y.y)MN \\ \to_\beta & (\lambda y.y)N \\ \to_\beta & y[N/y] \\ = & N \end{array}$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y . p\ x\ y$

$$(\lambda p \lambda x \lambda y . p\ x\ y)\texttt{True}\ MN$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y.p\ x\ y$

$$\begin{array}{rl} & (\lambda p \lambda x \lambda y.p\ x\ y)\texttt{True}\ MN \\ \rightarrow_\beta & (\lambda x \lambda y.\texttt{True}\ x\ y)MN \end{array}$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y.p\ x\ y$

$$\begin{array}{rc} & (\lambda p \lambda x \lambda y.p\ x\ y)\texttt{True}\ MN \\ \rightarrow_\beta & (\lambda x \lambda y.\texttt{True}\ x\ y)MN \\ \rightarrow_\beta^* & \texttt{True}\ MN \end{array}$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y.p\ x\ y$

$$\begin{array}{rc} & (\lambda p \lambda x \lambda y.p\ x\ y)\texttt{True}\ MN \\ \to_\beta & (\lambda x \lambda y.\texttt{True}\ x\ y)MN \\ \to_\beta^* & \texttt{True}\ MN \\ \to_\beta^* & M \end{array}$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y . p\ x\ y$

$$(\lambda p \lambda x \lambda y . p\ x\ y)\texttt{False}\ MN$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y.p\ x\ y$

$$\begin{array}{rl} & (\lambda p \lambda x \lambda y.p\ x\ y)\texttt{False}\ MN \\ \rightarrow_\beta & (\lambda x \lambda y.\ \texttt{False}\ x\ y)MN \end{array}$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y.p\ x\ y$

$$\begin{array}{rc} & (\lambda p \lambda x \lambda y.p\ x\ y)\texttt{False}\ MN \\ \to_\beta & (\lambda x \lambda y.\ \texttt{False}\ x\ y)MN \\ \to_\beta^* & \texttt{False}\ MN \end{array}$$

Programmare con il λ-calcolo

- `If Then Else`: $\lambda p \lambda x \lambda y.p\ x\ y$

$$
\begin{array}{rc}
 & (\lambda p \lambda x \lambda y.p\ x\ y)\texttt{False}\ MN \\
\to_\beta & (\lambda x \lambda y.\ \texttt{False}\ x\ y)MN \\
\to_\beta^* & \texttt{False}\ MN \\
\to_\beta^* & N
\end{array}
$$

Programmare con il λ-calcolo

- Gli interi di Church:

$$\underline{n} := \lambda f \lambda x . f^n x = \lambda f \lambda x \overbrace{f(f \cdots (f}^{n \text{ volte}} x) \cdots)$$

Programmare con il λ-calcolo

- Gli interi di Church:

$$\underline{n} := \lambda f \lambda x . f^n x = \lambda f \lambda x \overbrace{f(f \cdots (f}^{n \text{ volte}} x) \cdots)$$

$$\underline{0} := \lambda f \lambda x . x$$

Programmare con il λ-calcolo

- Gli interi di Church:

$$\underline{n} := \lambda f \lambda x.f^n x = \lambda f \lambda x \overbrace{f(f \cdots (f}^{n \text{ volte}} x) \cdots)$$

$$\underline{0} := \lambda f \lambda x.x$$

$$\underline{1} := \lambda f \lambda x.f\ x$$

Programmare con il λ-calcolo

- Gli interi di Church:

$$\underline{n} := \lambda f \lambda x.f^n x = \lambda f \lambda x \overbrace{f(f \cdots (f}^{n \text{ volte}} x) \cdots)$$

$$\underline{0} := \lambda f \lambda x.x$$

$$\underline{1} := \lambda f \lambda x.f\ x$$

$$\underline{2} := \lambda f \lambda x.f(f\ x)$$

Programmare con il λ-calcolo

- In generale, un intero n è interpretato come una funzione che itera n volte la funzione che gli viene applicata:

Programmare con il λ-calcolo

- In generale, un intero n è interpretato come una funzione che itera n volte la funzione che gli viene applicata:

$$(\lambda f \lambda x.f^n x)PQ$$

Programmare con il λ-calcolo

- In generale, un intero n è interpretato come una funzione che itera n volte la funzione che gli viene applicata:

$$\begin{array}{ll} & (\lambda f \lambda x.f^n x)PQ \\ \rightarrow_\beta & (\lambda x.P^n x)Q \end{array}$$

Programmare con il λ-calcolo

- In generale, un intero n è interpretato come una funzione che itera n volte la funzione che gli viene applicata:

$$\begin{array}{ll} & (\lambda f \lambda x.f^n x)PQ \\ \rightarrow_\beta & (\lambda x.P^n x)Q \\ \rightarrow_\beta & P^n Q \end{array}$$

Programmare con il λ-calcolo

- In generale, un intero n è interpretato come una funzione che itera n volte la funzione che gli viene applicata:

$$\begin{array}{ll} & (\lambda f \lambda x.f^n x)PQ \\ \to_\beta & (\lambda x.P^n x)Q \\ \to_\beta & P^n Q \\ = & \overbrace{P(P\cdots(P}^{n \text{ volte}} Q)\cdots) \end{array}$$

Programmare con il λ-calcolo

- La funzione *Succ*: $\lambda n \lambda f \lambda x.f(n\ f\ x)$

$$Succ\ \underline{n}$$

Programmare con il λ-calcolo

- La funzione *Succ*: $\lambda n \lambda f \lambda x.f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Succ}\ \underline{n} \\ = & (\lambda n \lambda f \lambda x.f(n\ f\ x))\underline{n} \end{array}$$

Programmare con il λ-calcolo

- La funzione *Succ*: $\lambda n \lambda f \lambda x.f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Succ}\ \underline{n} \\ = & (\lambda n \lambda f \lambda x.f(n\ f\ x))\underline{n} \\ \rightarrow_\beta & \lambda f \lambda x.f(\underline{n}\ f\ x) \end{array}$$

Programmare con il λ-calcolo

- La funzione *Succ*: $\lambda n \lambda f \lambda x.f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Succ}\ \underline{n} \\ = & (\lambda n \lambda f \lambda x.f(n\ f\ x))\underline{n} \\ \to_\beta & \lambda f \lambda x.f(\underline{n}\ f\ x) \\ \to_\beta & \lambda f \lambda x.f(\ f^n\ x) \end{array}$$

Programmare con il λ-calcolo

- La funzione *Succ*: $\lambda n \lambda f \lambda x.f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Succ}\ \underline{n} \\ = & (\lambda n \lambda f \lambda x.f(n\ f\ x))\underline{n} \\ \rightarrow_\beta & \lambda f \lambda x.f(\underline{n}\ f\ x) \\ \rightarrow_\beta & \lambda f \lambda x.f(\ f^n\ x) \\ = & \lambda f \lambda x.f^{n+1}\ x \end{array}$$

Programmare con il λ-calcolo

- La funzione *Succ*: $\lambda n \lambda f \lambda x.f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Succ}\ \underline{n} \\ = & (\lambda n \lambda f \lambda x.f(n\ f\ x))\underline{n} \\ \to_\beta & \lambda f \lambda x.f(\underline{n}\ f\ x) \\ \to_\beta & \lambda f \lambda x.f(\ f^n\ x) \\ = & \lambda f \lambda x.f^{n+1}\ x \\ = & \underline{n+1} \end{array}$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x . m\ f (n\ f\ x)$

$$Sum\ \underline{m}\ \underline{n}$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x)$

$$\begin{aligned} & \mathit{Sum}\ \underline{m}\ \underline{n} \\ = \quad & (\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x))\underline{m}\ \underline{n} \end{aligned}$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Sum}\ \underline{m}\ \underline{n} \\ = & (\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x))\underline{m}\ \underline{n} \\ \rightarrow_\beta & (\lambda n \lambda f \lambda x.\underline{m}\ f(n\ f\ x))\underline{n} \end{array}$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Sum}\ \underline{m}\ \underline{n} \\ = & (\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x))\underline{m}\ \underline{n} \\ \rightarrow_\beta & (\lambda n \lambda f \lambda x.\underline{m}\ f(n\ f\ x))\underline{n} \\ \rightarrow_\beta & \lambda f \lambda x.\underline{m}\ f(\underline{n}\ f\ x) \end{array}$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Sum}\ \underline{m}\ \underline{n} \\ = & (\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x))\underline{m}\ \underline{n} \\ \rightarrow_\beta & (\lambda n \lambda f \lambda x.\underline{m}\ f(n\ f\ x))\underline{n} \\ \rightarrow_\beta & \lambda f \lambda x.\underline{m}\ f(\underline{n}\ f\ x) \\ = & \lambda f \lambda x.\underline{m}\ f(\ f^n\ x) \end{array}$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x)$

$$\begin{array}{rc} & \mathit{Sum}\ \underline{m}\ \underline{n} \\ = & (\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x))\underline{m}\ \underline{n} \\ \rightarrow_\beta & (\lambda n \lambda f \lambda x.\underline{m}\ f(n\ f\ x))\underline{n} \\ \rightarrow_\beta & \lambda f \lambda x.\underline{m}\ f(\underline{n}\ f\ x) \\ = & \lambda f \lambda x.\underline{m}\ f(\ f^n\ x) \\ = & \lambda f \lambda x.f^m(\ f^n\ x) \end{array}$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x)$

$$
\begin{array}{rc}
 & \mathit{Sum}\ \underline{m}\ \underline{n} \\
= & (\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x))\underline{m}\ \underline{n} \\
\rightarrow_\beta & (\lambda n \lambda f \lambda x.\underline{m}\ f(n\ f\ x))\underline{n} \\
\rightarrow_\beta & \lambda f \lambda x.\underline{m}\ f(\underline{n}\ f\ x) \\
= & \lambda f \lambda x.\underline{m}\ f(\ f^n\ x) \\
= & \lambda f \lambda x.f^m(\ f^n\ x) \\
= & \lambda f \lambda x.f^{m+n}\ x
\end{array}
$$

Programmare con il λ-calcolo

- La funzione *Sum*: $\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x)$

$$
\begin{array}{rc}
 & \mathit{Sum}\ \underline{m}\ \underline{n} \\
= & (\lambda m \lambda n \lambda f \lambda x.m\ f(n\ f\ x))\underline{m}\ \underline{n} \\
\rightarrow_\beta & (\lambda n \lambda f \lambda x.\underline{m}\ f(n\ f\ x))\underline{n} \\
\rightarrow_\beta & \lambda f \lambda x.\underline{m}\ f(\underline{n}\ f\ x) \\
= & \lambda f \lambda x.\underline{m}\ f(\ f^n\ x) \\
= & \lambda f \lambda x.f^m(\ f^n\ x) \\
= & \lambda f \lambda x.f^{m+n}\ x \\
= & \underline{m+n}
\end{array}
$$

Programmare con il λ-calcolo

- Data una funzione f da X a X, un *punto fisso* di f è un $x \in X$ tale che $x = f(x)$.

Programmare con il λ-calcolo

- Data una funzione f da X a X, un *punto fisso* di f è un $x \in X$ tale che $x = f(x)$.
- Un λ-termine Y si dice *combinatore di punto fisso* se per ogni λ-termine F

$$Y(F) \to_\beta^* F(Y(F))$$

Programmare con il λ-calcolo

- Data una funzione f da X a X, un *punto fisso* di f è un $x \in X$ tale che $x = f(x)$.
- Un λ-termine Y si dice *combinatore di punto fisso* se per ogni λ-termine F

$$Y(F) \to_\beta^* F(Y(F))$$

Combinatore di punto fisso di Turing

Se $A = \lambda a \lambda f . f\ (a\ a\ f)$, allora $\Theta := A\ A$ è un combinatore di punto fisso.

Programmare con il λ-calcolo

- Data una funzione f da X a X, un *punto fisso* di f è un $x \in X$ tale che $x = f(x)$.
- Un λ-termine Y si dice *combinatore di punto fisso* se per ogni λ-termine F

$$Y(F) \to_\beta^* F(Y(F))$$

Combinatore di punto fisso di Turing

Se $A = \lambda a \lambda f.f\ (a\ a\ f)$, allora $\Theta := A\ A$ è un combinatore di punto fisso.

Dim. $\Theta = (\lambda a \lambda f.f\ (a\ a\ f))A \to_\beta \lambda f.f\ (A\ A\ f) = \lambda f.f\ (\Theta\ f)$. Ora per ogni F, $\Theta\ F \to_\beta (\lambda f.f\ (\Theta\ f))F \to_\beta F(\Theta\ F)$.

Programmare con il λ-calcolo

- I combinatori di punto fisso nel λ-calcolo sono fondamentali per rappresentare le *funzioni definite per ricorsione*, che a loro volta permettono di definire in maniera funzionale i cicli `while` dei linguaggi di programmazione standard.

Programmare con il λ-calcolo

- I combinatori di punto fisso nel λ-calcolo sono fondamentali per rappresentare le *funzioni definite per ricorsione*, che a loro volta permettono di definire in maniera funzionale i cicli `while` dei linguaggi di programmazione standard.
- Es. il fattoriale $n!$ è definito per ricorsione:

$$n! = \begin{cases} 1, & \text{se } n = 0 \\ n * (n-1)!, & \text{altrimenti} \end{cases}$$

Indice

La logica e l'argomentazione

La logica (dal greco $\lambda o \gamma o \varsigma$, logos, ovvero "parola", "pensiero", "idea", "argomento", "ragione", da cui poi $\lambda o \gamma \iota \kappa \eta$,) è lo studio del ragionamento e dell'argomentazione e, in particolare, dei procedimenti inferenziali, rivolto a chiarire quali procedimenti di pensiero siano validi e quali non validi.

Wikipedia

Il *Modus Ponens*

- Se piove, allora prendo l'ombrello.

Il *Modus Ponens*

- Se piove, allora prendo l'ombrello.
- Piove.

Il *Modus Ponens*

- Se piove, allora prendo l'ombrello.
- Piove.
- Prendo l'ombrello.

Il *Modus Ponens*

- Se n è un numero intero maggiore di 1, allora n può essere espresso in maniera univoca come prodotto di numeri primi.

Il *Modus Ponens*

- Se n è un numero intero maggiore di 1, allora n può essere espresso in maniera univoca come prodotto di numeri primi.
- 6396 è un numero intero maggiore di 1.

Il *Modus Ponens*

- Se n è un numero intero maggiore di 1, allora n può essere espresso in maniera univoca come prodotto di numeri primi.
- 6396 è un numero intero maggiore di 1.
- 6396 può essere espresso in maniera univoca come prodotto di numeri primi.

Il *Modus Ponens*

- Se Roma è la capitale dell'Italia, allora Babbo Natale è verde.

Il *Modus Ponens*

- Se Roma è la capitale dell'Italia, allora Babbo Natale è verde.
- Roma è la capitale dell'Italia.

Il *Modus Ponens*

- Se Roma è la capitale dell'Italia, allora Babbo Natale è verde.
- Roma è la capitale dell'Italia.
- Babbo Natale è verde.

Modus Ponens

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

- Se A allora B.
- A.
- dunque B.

La Logica classica (Sistema di Hilbert)

Tutte e sole le proposizioni classicamente valide sono dimostrabili usando :

La Logica classica (Sistema di Hilbert)

Tutte e sole le proposizioni classicamente valide sono dimostrabili usando :

- i seguenti schemi d'assioma:

  H1 $A \Rightarrow (B \Rightarrow A)$

La Logica classica (Sistema di Hilbert)

Tutte e sole le proposizioni classicamente valide sono dimostrabili usando :

- i seguenti schemi d'assioma:

H1 $A \Rightarrow (B \Rightarrow A)$
H2 $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

La Logica classica (Sistema di Hilbert)

Tutte e sole le proposizioni classicamente valide sono dimostrabili usando :

- i seguenti schemi d'assioma:

H1 $A \Rightarrow (B \Rightarrow A)$
H2 $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
H3 $(\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$

La Logica classica (Sistema di Hilbert)

Tutte e sole le proposizioni classicamente valide sono dimostrabili usando :

- i seguenti schemi d'assioma:

$$\begin{array}{ll} \text{H1} & A \Rightarrow (B \Rightarrow A) \\ \text{H2} & (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)) \\ \text{H3} & (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B) \end{array}$$

- la seguente regola di inferenza :

$$\frac{A \Rightarrow B \qquad A}{B}\ MP$$

La Logica classica (Sistema di Hilbert)

$$A \Rightarrow A$$

La Logica classica (Sistema di Hilbert)

$$A \Rightarrow A$$

- H2) $(A \Rightarrow (A \Rightarrow A) \Rightarrow A) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$

La Logica classica (Sistema di Hilbert)

$$A \Rightarrow A$$

- H2) $(A \Rightarrow (A \Rightarrow A) \Rightarrow A) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$
- H1) $A \Rightarrow (A \Rightarrow A) \Rightarrow A$

La Logica classica (Sistema di Hilbert)

$$A \Rightarrow A$$

- H2) $(A \Rightarrow (A \Rightarrow A) \Rightarrow A) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$
- H1) $A \Rightarrow (A \Rightarrow A) \Rightarrow A$
- MP) $((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$

La Logica classica (Sistema di Hilbert)

$$A \Rightarrow A$$

- H2) $(A \Rightarrow (A \Rightarrow A) \Rightarrow A) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$
- H1) $A \Rightarrow (A \Rightarrow A) \Rightarrow A$
- MP) $((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$
- H1) $A \Rightarrow (A \Rightarrow A)$

La Logica classica (Sistema di Hilbert)

$$A \Rightarrow A$$

- H2) $(A \Rightarrow (A \Rightarrow A) \Rightarrow A) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$
- H1) $A \Rightarrow (A \Rightarrow A) \Rightarrow A$
- MP) $((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$
- H1) $A \Rightarrow (A \Rightarrow A)$
- MP) $A \Rightarrow A$

Osservazioni sul calcolo

- Costruire una dimostrazione nel sistema alla Hilbert è una procedura piuttosto involuta...;

Osservazioni sul calcolo

- Costruire una dimostrazione nel sistema alla Hilbert è una procedura piuttosto involuta...;
- La struttura di una derivazione è estremamente povera, contiene pochissime informazioni;

Osservazioni sul calcolo

- Costruire una dimostrazione nel sistema alla Hilbert è una procedura piuttosto involuta...;
- La struttura di una derivazione è estremamente povera, contiene pochissime informazioni;
- La derivazione non rispecchia il procedere naturale del ragionamento matematico.

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

- A: Siano c e d due angoli opposti al vertice;

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

- A: Siano c e d due angoli opposti al vertice;
- B : c e d sono congruenti.

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

- A: Siano c e d due angoli opposti al vertice;
- B : c e d sono congruenti.
- Dimostriamo $A \Rightarrow B$ in maniera indiretta: assumiamo che c e d siano non siano congruenti ($\neg B$);

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

- A: Siano c e d due angoli opposti al vertice;
- B : c e d sono congruenti.
- Dimostriamo $A \Rightarrow B$ in maniera indiretta: assumiamo che c e d siano non siano congruenti ($\neg B$);
- se da $\neg B$ (c e d non sono congruenti) riesco a dimostrare $\neg A$ (c e d non sono opposti al vertice), cioé se riesco a dimostrare $\neg B \Rightarrow \neg A$ per [H3] e *Modus Ponens* posso concludere $A \Rightarrow B$, cvd.

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

$$\neg\neg A \Rightarrow A$$

- Ragionamento per assurdo: Dalla falsita di $\neg A$ segue la verita di A.

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

$$\neg\neg A \Rightarrow A$$

- Ragionamento per assurdo: Dalla falsita di $\neg A$ segue la verita di A.
- Principio non costruttivo: per dimostrare l'esistenza di un oggetto matematico non è sufficiente dimostrare che ammettere il contrario porta a contraddizione; bisogna esibire l'oggetto in questione.

Osservazioni sulla logica classica

$$[\text{H3}]\ (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

$$\neg\neg A \Rightarrow A$$

- Ragionamento per assurdo: Dalla falsita di $\neg A$ segue la verita di A.
- Principio non costruttivo: per dimostrare l'esistenza di un oggetto matematico non è sufficiente dimostrare che ammettere il contrario porta a contraddizione; bisogna esibire l'oggetto in questione.
- Logica intuizionista Logica che non ammette il ragionamento per assurdo (Brouwer, Prawitz).

La deduzione naturale intuizionista di Gentzen (*ND*)

- Le dimostrazioni in deduzione naturale sono alberi in cui:

La deduzione naturale intuizionista di Gentzen (*ND*)

- Le dimostrazioni in deduzione naturale sono alberi in cui:
  - le formule corrispondono ai nodi;

La deduzione naturale intuizionista di Gentzen (*ND*)

- Le dimostrazioni in deduzione naturale sono alberi in cui:
  - le formule corrispondono ai nodi;
  - le ipotesi corrispondono alle foglie;

La deduzione naturale intuizionista di Gentzen (*ND*)

- Le dimostrazioni in deduzione naturale sono alberi in cui:
  - le formule corrispondono ai nodi;
  - le ipotesi corrispondono alle foglie;
  - la conclusione corrisponde alla radice.

La deduzione naturale intuizionista di Gentzen (*ND*)

- Le dimostrazioni in deduzione naturale sono alberi in cui:
  - le formule corrispondono ai nodi;
  - le ipotesi corrispondono alle foglie;
  - la conclusione corrisponde alla radice.
  - le formule sono collegate assieme da regole di inferenza

La deduzione naturale intuizionista di Gentzen (*ND*)

- Le dimostrazioni in deduzione naturale sono alberi in cui:
  - le formule corrispondono ai nodi;
  - le ipotesi corrispondono alle foglie;
  - la conclusione corrisponde alla radice.
  - le formule sono collegate assieme da regole di inferenza
- le regole di inferenza sono divise in due tipi:
  - le regole di introduzione, che specificano il modo in cui una particolare formula viene dedotta a partire da un certo numero di premesse;

La deduzione naturale intuizionista di Gentzen (*ND*)

- Le dimostrazioni in deduzione naturale sono alberi in cui:
  - le formule corrispondono ai nodi;
  - le ipotesi corrispondono alle foglie;
  - la conclusione corrisponde alla radice.
  - le formule sono collegate assieme da regole di inferenza
- le regole di inferenza sono divise in due tipi:
  - le regole di introduzione, che specificano il modo in cui una particolare formula viene dedotta a partire da un certo numero di premesse;
  - le regole di eliminazione, che specificano il modo in cui una particolare formula può essere utilizzata per derivare altre formule.

Deduzione naturale: l'ipotesi

$$A$$

Deduzione naturale: l'ipotesi

$$A$$

Sotto l'ipotesi A è possibile concludere A.

Deduzione naturale: la regola di $\Rightarrow\mathcal{I}$

$$\begin{array}{c} A \\ \vdots \\ B \end{array}$$

- Supponiamo di avere dimostrato la formula B utilizzando un certo numero di volte l'ipotesi A;

Deduzione naturale: la regola di $\Rightarrow \mathcal{I}$

$$\dfrac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \Rightarrow B} \Rightarrow \mathcal{I}$$

- Supponiamo di avere una dimostrazione della formula B che usa un certo numero di volte l'ipotesi A;
- allora possiamo "consumare" un determinato numero di volte l'ipotesi A per produrre una dimostrazione di $A \Rightarrow B$.

Deduzione naturale: la regola di $\Rightarrow \mathcal{E}$

$$\begin{array}{cc} \vdots & \vdots \\ A \Rightarrow B & A \end{array}$$

- Supponiamo di avere due dimostrazioni indipendenti rispettivamente di $A \Rightarrow B$ e di A;

Deduzione naturale: la regola di $\Rightarrow \mathcal{E}$

$$\frac{\begin{array}{cc}\vdots & \vdots \\ A \Rightarrow B & A\end{array}}{B} \Rightarrow \mathcal{E}$$

- Supponiamo di avere due dimostrazioni indipendenti rispettivamente di $A \Rightarrow B$ e di A;
- allora possiamo comporre le due dimostrazioni per produrre una dimostrazione di B.

Deduzione naturale: la regola di $\wedge\mathcal{I}$

$$\begin{array}{cc} \vdots & \vdots \\ A & B \end{array}$$

- Supponiamo di avere due dimostrazioni indipendenti rispettivamente di A e di B;

Deduzione naturale: la regola di $\wedge\mathcal{I}$

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ A & B\end{array}}{A \wedge B}\ \wedge\mathcal{I}$$

- Supponiamo di avere due dimostrazioni indipendenti rispettivamente di A e di B;
- allora possiamo comporre le due dimostrazioni per produrre una dimostrazione di $A \wedge B$.

Deduzione naturale: la regola di $\wedge\mathcal{E}1$

$$\begin{array}{c} \vdots \\ A \wedge B \end{array}$$

- Supponiamo di avere una dimostrazione della formula $A \wedge B$;

Deduzione naturale: la regola di $\wedge\mathcal{E}1$

$$\frac{\begin{array}{c}\vdots\\ A \wedge B\end{array}}{A}\ \wedge\mathcal{E}1$$

- Supponiamo di avere una dimostrazione della formula $A \wedge B$;
- allora possiamo "selezionare" il primo elemento della congiunzione ed ottenere così una dimostrazione di A.

Deduzione naturale: la regola di $\wedge\mathcal{E}2$

$$\begin{array}{c} \vdots \\ A \wedge B \end{array}$$

- Supponiamo di avere una dimostrazione della formula $A \wedge B$;

Deduzione naturale: la regola di $\wedge\mathcal{E}2$

$$\frac{\begin{array}{c}\vdots\\ A \wedge B\end{array}}{B}\ \wedge\mathcal{E}2$$

- Supponiamo di avere una dimostrazione della formula $A \wedge B$;
- allora possiamo "selezionare" il secondo elemento della congiunzione ed ottenere così una dimostrazione di B.

Esempio

$$A \Rightarrow A$$

Esempio

$$\dfrac{\dfrac{[A]}{A}\Rightarrow\mathcal{I}}{A\Rightarrow A}$$

Esempio

$$\frac{[A]}{A \Rightarrow A} \Rightarrow \mathcal{I}$$

Esempio

$$A \Rightarrow (B \Rightarrow A)$$

Esempio

$$\dfrac{\dfrac{[A]}{B \Rightarrow A}\Rightarrow \mathcal{I}}{A \Rightarrow (B \Rightarrow A)}$$

Esempio

$$\dfrac{\dfrac{[A]}{B \Rightarrow A}\Rightarrow\mathcal{I}}{A \Rightarrow (B \Rightarrow A)}\Rightarrow\mathcal{I}\qquad [B]$$

Esempio

$$\dfrac{\dfrac{[A]}{B \Rightarrow A} \Rightarrow \mathcal{I}}{A \Rightarrow (B \Rightarrow A)} \Rightarrow \mathcal{I}$$

Esempio

$$(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$$

Esempio

$$[A \Rightarrow (B \Rightarrow C)]$$

$$\frac{(A \Rightarrow B) \Rightarrow (A \Rightarrow C)}{(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))} \Rightarrow \mathcal{I}$$

Esempio

$$[A \Rightarrow (B \Rightarrow C)] \qquad\qquad [A \Rightarrow B]$$

$$\dfrac{\dfrac{A \Rightarrow C}{(A \Rightarrow B) \Rightarrow (A \Rightarrow C)} \Rightarrow \mathcal{I}}{(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))} \Rightarrow \mathcal{I}$$

Esempio

$$[A \Rightarrow (B \Rightarrow C)] \qquad [A] \qquad [A \Rightarrow B]$$

$$\dfrac{\dfrac{\dfrac{C}{A \Rightarrow C} \Rightarrow \mathcal{I}}{(A \Rightarrow B) \Rightarrow (A \Rightarrow C)} \Rightarrow \mathcal{I}}{(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))} \Rightarrow \mathcal{I}$$

Esempio

$$\dfrac{\dfrac{[A \Rightarrow (B \Rightarrow C)] \qquad [A]}{B \Rightarrow C} \Rightarrow \mathcal{E} \qquad [A \Rightarrow B]}{\dfrac{\dfrac{C}{A \Rightarrow C} \Rightarrow \mathcal{I}}{(A \Rightarrow B) \Rightarrow (A \Rightarrow C)} \Rightarrow \mathcal{I}}{(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))} \Rightarrow \mathcal{I}$$

Esempio

$$\dfrac{\dfrac{[A \Rightarrow (B \Rightarrow C)] \qquad [A]}{B \Rightarrow C} \Rightarrow \mathcal{E} \qquad [A \Rightarrow B] \qquad [A]}{\dfrac{\dfrac{C}{A \Rightarrow C} \Rightarrow \mathcal{I}}{(A \Rightarrow B) \Rightarrow (A \Rightarrow C)} \Rightarrow \mathcal{I}}{(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))} \Rightarrow \mathcal{I}$$

Esempio

$$\dfrac{\dfrac{[A \Rightarrow (B \Rightarrow C)] \qquad [A]}{B \Rightarrow C} \Rightarrow \mathcal{E} \qquad \dfrac{[A \Rightarrow B] \qquad [A]}{B} \Rightarrow \mathcal{E}}{\dfrac{\dfrac{C}{A \Rightarrow C} \Rightarrow \mathcal{I}}{(A \Rightarrow B) \Rightarrow (A \Rightarrow C)} \Rightarrow \mathcal{I}}{(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))} \Rightarrow \mathcal{I}$$

Esempio

$$\dfrac{\dfrac{\dfrac{\dfrac{[A \Rightarrow (B \Rightarrow C)] \qquad [A]}{B \Rightarrow C} \Rightarrow \mathcal{E} \qquad \dfrac{[A \Rightarrow B] \qquad [A]}{B} \Rightarrow \mathcal{E}}{C} \Rightarrow \mathcal{E}}{A \Rightarrow C} \Rightarrow \mathcal{I}}{(A \Rightarrow B) \Rightarrow (A \Rightarrow C)} \Rightarrow \mathcal{I}}{(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))} \Rightarrow \mathcal{I}$$

Una dimostrazione arzigogolata...

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]}{A}\wedge\mathcal{E}2 \quad \dfrac{[B \wedge A]}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}}{(B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \qquad \dfrac{B \qquad A}{B \wedge A}\wedge\mathcal{I}}{A \wedge B}\Rightarrow\mathcal{E}$$

Una dimostrazione arzigogolata...

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]}{A}\wedge\mathcal{E}2 \quad \dfrac{[B \wedge A]}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}}{(B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \qquad \dfrac{B \qquad A}{B \wedge A}\wedge\mathcal{I}}{A \wedge B}\Rightarrow\mathcal{E}$$

"Dall' ipotesi A e dall'ipotesi B segue $A \wedge B$"

Una dimostrazione arzigogolata...

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]}{A}\wedge\mathcal{E}2 \quad \dfrac{[B \wedge A]}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}}{(B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \qquad \dfrac{B \qquad A}{B \wedge A}\wedge\mathcal{I}}{A \wedge B}\Rightarrow\mathcal{E}$$

"Dall' ipotesi A e dall'ipotesi B segue $A \wedge B$"

$$\{A, B\} \vdash A \wedge B$$

La proprietà della sottoformula

Sia Γ un (multi-)insieme di formule:

La proprietà della sottoformula

Sia Γ un (multi-)insieme di formule:

$$\Gamma \vdash A$$

La proprietà della sottoformula

Sia Γ un (multi-)insieme di formule:

$$\Gamma \vdash A$$

"Dalle ipotesi in Γ *segue* A*"*

La proprietà della sottoformula

Sia Γ un (multi-)insieme di formule:

$$\Gamma \vdash A$$

"Dalle ipotesi in Γ *segue* A*"*

Proprietà della sottoformula

Una prova π di $\Gamma \vdash A$ rispetta la proprietà della sottoformula se le formule che compaiono in π sono tutte e sole e formule che compaiono in Γ, A.

Il taglio

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]}{A}\wedge\mathcal{E}2 \quad \dfrac{[B \wedge A]}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}}{(B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \qquad \dfrac{B \qquad A}{B \wedge A}\wedge\mathcal{I}}{A \wedge B}\Rightarrow\mathcal{E}$$

Il taglio

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]}{A}\wedge\mathcal{E}2 \quad \dfrac{[B \wedge A]}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}}{(B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \qquad \dfrac{B \qquad A}{B \wedge A}\wedge\mathcal{I}}{A \wedge B}\Rightarrow\mathcal{E}$$

- Taglio: L'introduzione di una formula seguita immediatamente dalla sua eliminazione.

Il taglio

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]}{A}\wedge\mathcal{E}2 \quad \dfrac{[B \wedge A]}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}}{(B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \qquad \dfrac{B \qquad A}{B \wedge A}\wedge\mathcal{I}}{A \wedge B}\Rightarrow\mathcal{E}$$

- Taglio: L'introduzione di una formula seguita immediatamente dalla sua eliminazione.
- E' possibile trasformare, riscrivere una prova in maniera da ottenere una prova senza tagli, preservando conclusione ed ipotesi ?

Eliminazione del taglio: il caso $\Rightarrow$

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \Rightarrow B}\Rightarrow\mathcal{I} \qquad \begin{array}{c}\pi\\ \vdots\\ A\end{array}}{B}\Rightarrow\mathcal{E} \quad \rightsquigarrow \quad \begin{array}{c}\pi\\ \vdots\\ A\\ \vdots\\ B\end{array}$$

Eliminazione del taglio: il caso $\wedge$

$$\dfrac{\dfrac{\begin{array}{cc}\pi_1 & \pi_2 \\ \vdots & \vdots \\ A & B\end{array}}{A \wedge B}\,\wedge\mathcal{I}}{A}\,\wedge\mathcal{E}1 \quad \rightsquigarrow \quad \begin{array}{c}\pi_1 \\ \vdots \\ A\end{array}$$

Eliminazione del taglio: il caso $\wedge$

$$\dfrac{\dfrac{\begin{array}{cc}\pi_1 & \pi_2 \\ \vdots & \vdots \\ A & B\end{array}}{A \wedge B}\wedge\mathcal{I}}{B}\wedge\mathcal{E}2 \quad \rightsquigarrow \quad \begin{array}{c}\pi_2 \\ \vdots \\ B\end{array}$$

Riduzione di una prova

$$\dfrac{\dfrac{\dfrac{\dfrac{[B \wedge A]}{A}\wedge\mathcal{E}2 \quad \dfrac{[B \wedge A]}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}}{(B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \qquad \dfrac{B \qquad A}{B \wedge A}\wedge\mathcal{I}}{A \wedge B}\Rightarrow\mathcal{E} \quad \rightsquigarrow$$

Riduzione di una prova

$$\rightsquigarrow \quad \dfrac{\dfrac{\dfrac{B \quad A}{B \wedge A}\wedge\mathcal{I}}{A}\wedge\mathcal{E}2 \qquad \dfrac{\dfrac{B \quad A}{B \wedge A}\wedge\mathcal{I}}{B}\wedge\mathcal{E}1}{A \wedge B}\wedge\mathcal{I}$$

Riduzione di una prova

$$\dfrac{\dfrac{\dfrac{B \quad A}{B \wedge A}\,\wedge\mathcal{I}}{A}\,\wedge\mathcal{E}2 \qquad \dfrac{\dfrac{B \quad A}{B \wedge A}\,\wedge\mathcal{I}}{B}\,\wedge\mathcal{E}1}{A \wedge B}\,\wedge\mathcal{I} \quad \rightsquigarrow$$

Riduzione di una prova

$$\rightsquigarrow \quad \dfrac{\dfrac{\dfrac{B \qquad A}{B \wedge A}\,\wedge\mathcal{I}}{B}\,\wedge\mathcal{E}1 \qquad A}{A \wedge B}\,\wedge\mathcal{I} \quad \rightsquigarrow$$

Riduzione di una prova

$$\rightsquigarrow \quad \dfrac{A \qquad B}{A \wedge B}\,\wedge\mathcal{I}$$

Riduzione di una prova

$$\rightsquigarrow \quad \frac{A \qquad B}{A \wedge B} \wedge\mathcal{I}$$

"Dall' ipotesi A e dall'ipotesi B segue $A \wedge B$"

$$\{A, B\} \vdash A \wedge B$$

Il teorema di eliminazione del taglio

Teorema di eliminazione del taglio, Gentzen

Per ogni prova π di $\Gamma \vdash A$ in deduzione naturale, esiste una prova π' di $\Gamma \vdash A$ senza tagli.

Il teorema di eliminazione del taglio

Teorema di eliminazione del taglio, Gentzen

Per ogni prova π di $\Gamma \vdash A$ in deduzione naturale, esiste una prova π' di $\Gamma \vdash A$ senza tagli.

Corollario

Per ogni prova π di $\Gamma \vdash A$ in deduzione naturale, esiste una prova π' di $\Gamma \vdash A$ in cui occorrono solo sottoformule delle formule in Γ, A.

Il teorema di eliminazione del taglio

Teorema di eliminazione del taglio, Gentzen

Per ogni prova π di $\Gamma \vdash A$ in deduzione naturale, esiste una prova π' di $\Gamma \vdash A$ senza tagli.

Corollario

Per ogni prova π di $\Gamma \vdash A$ in deduzione naturale, esiste una prova π' di $\Gamma \vdash A$ in cui occorrono solo sottoformule delle formule in Γ, A.

La procedura di trasformazione delle dimostrazioni sembra ricordare un processo di calcolo....

λ-calcolo e normalizzazione forte

- Consideriamo il λ-termine considerato il "prototipo" della non-terminazione:

$$\lambda x.xx$$

λ-calcolo e normalizzazione forte

- Consideriamo il λ-termine considerato il "prototipo" della non-terminazione:

$$\lambda x.xx$$

- Cosa fa $\lambda x.xx$? Prende in input un termine e lo applica a se stesso.

λ-calcolo e normalizzazione forte

- Consideriamo il λ-termine considerato il "prototipo" della non-terminazione:

$$\lambda x.xx$$

- Cosa fa $\lambda x.xx$? Prende in input un termine e lo applica a se stesso.
- Ora cosa succede se diamo in input a $\lambda x.xx$ se stesso?

$$(\lambda x.xx)\lambda x.xx \rightarrow_\beta *(\lambda x.xx)\lambda x.xx \rightarrow_\beta * \dots$$

λ-calcolo e normalizzazione forte

- Consideriamo il λ-termine considerato il "prototipo" della non-terminazione:

$$\lambda x.xx$$

- Cosa fa $\lambda x.xx$? Prende in input un termine e lo applica a se stesso.
- Ora cosa succede se diamo in input a $\lambda x.xx$ se stesso?

$$(\lambda x.xx)\lambda x.xx \rightarrow_\beta *(\lambda x.xx)\lambda x.xx \rightarrow_\beta * \dots$$

La radice della non-terminazione è l'autoapplicazione! Come impedirla?

Il λ-calcolo semplicemente tipato

- λ-termini:

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM \quad | \quad \langle MM \rangle \quad | \quad fst\ M \quad | \quad snd\ M$$

Il λ-calcolo semplicemente tipato

- λ-termini:

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MM \quad | \quad \langle MM \rangle \quad | \quad fst\ M \quad | \quad snd\ M$$

  - *"Zucchero sintattico"*:

$$fst\langle M_1 M_2 \rangle \rightarrow_\beta M_1$$

$$snd\langle M_1 M_2 \rangle \rightarrow_\beta M_2$$

I tipi semplici

- Sia data una infinità numerabile di variabili `A, B, C ...` dette *variabili* di tipo:

I tipi semplici

- Sia data una infinità numerabile di variabili `A`, `B`, `C` ... dette *variabili* di tipo:
  1. Una variabile di tipo `A` è un tipo semplice;

I tipi semplici

- Sia data una infinità numerabile di variabili `A`, `B`, `C` ... dette *variabili* di tipo:
  1. Una variabile di tipo `A` è un tipo semplice;
  2. Se `A` è un tipo semplice e `B` è un tipo semplice, allora $\texttt{A} \to \texttt{B}$ è un tipo semplice;

I tipi semplici

- Sia data una infinità numerabile di variabili $\mathtt{A}, \mathtt{B}, \mathtt{C} \ldots$ dette *variabili* di tipo:
  1. Una variabile di tipo $\mathtt{A}$ è un tipo semplice;
  2. Se $\mathtt{A}$ è un tipo semplice e $\mathtt{B}$ è un tipo semplice, allora $\mathtt{A} \to \mathtt{B}$ è un tipo semplice;
  3. Se $\mathtt{A}$ è un tipo semplice e $\mathtt{B}$ è un tipo semplice, allora $\mathtt{A} \times \mathtt{B}$ è un tipo semplice;

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo $\mathtt{A}$:

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo $\mathtt{A}$:
- Se x è una variabile di tipo $\mathtt{A}$ e M è un termine di tipo $\mathtt{B}$ allora $\lambda x.M$ è un termine di tipo $\mathtt{A} \to \mathtt{B}$;

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo $\mathtt{A}$:
- Se x è una variabile di tipo $\mathtt{A}$ e M è un termine di tipo $\mathtt{B}$ allora $\lambda x.M$ è un termine di tipo $\mathtt{A} \to \mathtt{B}$;
- Se M è un termine di tipo $\mathtt{A} \to \mathtt{B}$ e N è un termine di tipo $\mathtt{A}$ allora $M\ N$ è un termine di tipo $\mathtt{B}$.

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo $\mathtt{A}$:
- Se x è una variabile di tipo $\mathtt{A}$ e M è un termine di tipo $\mathtt{B}$ allora $\lambda x.M$ è un termine di tipo $\mathtt{A} \to \mathtt{B}$;
- Se M è un termine di tipo $\mathtt{A} \to \mathtt{B}$ e N è un termine di tipo $\mathtt{A}$ allora $M\ N$ è un termine di tipo $\mathtt{B}$.
- Se M è un termine di tipo $\mathtt{A}$ e N è un termine di tipo $\mathtt{B}$ allora $\langle MN \rangle$ è un termine di tipo $\mathtt{A} \times \mathtt{B}$.

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo $\mathtt{A}$:
- Se x è una variabile di tipo $\mathtt{A}$ e M è un termine di tipo $\mathtt{B}$ allora $\lambda x.M$ è un termine di tipo $\mathtt{A} \to \mathtt{B}$;
- Se M è un termine di tipo $\mathtt{A} \to \mathtt{B}$ e N è un termine di tipo $\mathtt{A}$ allora $M\ N$ è un termine di tipo $\mathtt{B}$.
- Se M è un termine di tipo $\mathtt{A}$ e N è un termine di tipo $\mathtt{B}$ allora $\langle MN \rangle$ è un termine di tipo $\mathtt{A} \times \mathtt{B}$.
- Se M è un termine di tipo $\mathtt{A} \times \mathtt{B}$, allora $fst\ M$ è un termine di tipo $\mathtt{A}$.

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo $\mathtt{A}$:
- Se x è una variabile di tipo $\mathtt{A}$ e M è un termine di tipo $\mathtt{B}$ allora $\lambda x.M$ è un termine di tipo $\mathtt{A} \to \mathtt{B}$;
- Se M è un termine di tipo $\mathtt{A} \to \mathtt{B}$ e N è un termine di tipo $\mathtt{A}$ allora $M\ N$ è un termine di tipo $\mathtt{B}$.
- Se M è un termine di tipo $\mathtt{A}$ e N è un termine di tipo $\mathtt{B}$ allora $\langle MN \rangle$ è un termine di tipo $\mathtt{A} \times \mathtt{B}$.
- Se M è un termine di tipo $\mathtt{A} \times \mathtt{B}$, allora $fst\ M$ è un termine di tipo $\mathtt{A}$.
- Se M è un termine di tipo $\mathtt{A} \times \mathtt{B}$, allora $snd\ M$ è un termine di tipo $\mathtt{B}$.

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo $\mathtt{A}$:
- Se x è una variabile di tipo $\mathtt{A}$ e M è un termine di tipo $\mathtt{B}$ allora $\lambda x.M$ è un termine di tipo $\mathtt{A} \to \mathtt{B}$;
- Se M è un termine di tipo $\mathtt{A} \to \mathtt{B}$ e N è un termine di tipo $\mathtt{A}$ allora $M\ N$ è un termine di tipo $\mathtt{B}$.
- Se M è un termine di tipo $\mathtt{A}$ e N è un termine di tipo $\mathtt{B}$ allora $\langle MN \rangle$ è un termine di tipo $\mathtt{A} \times \mathtt{B}$.
- Se M è un termine di tipo $\mathtt{A} \times \mathtt{B}$, allora *fst* M è un termine di tipo $\mathtt{A}$.
- Se M è un termine di tipo $\mathtt{A} \times \mathtt{B}$, allora *snd* M è un termine di tipo $\mathtt{B}$.

$M^{\mathtt{A}}$ (o anche $M : \mathtt{A}$): *il termine* M *ha tipo* $\mathtt{A}$.

Le regole di riduzione nel λ-calcolo semplicemente tipato

$$((\lambda x^{\mathtt{A}}.M^{\mathtt{B}})^{\mathtt{A}\to\mathtt{B}} N^{\mathtt{A}})^{\mathtt{B}} \to_\beta M^{\mathtt{B}}[N^{\mathtt{A}}/x^{\mathtt{A}}]$$

$$(fst(\langle M_1^{\mathtt{A}} M_2^{\mathtt{B}}\rangle)^{\mathtt{A}\times\mathtt{B}})^{\mathtt{A}} \to_\beta M_1^{\mathtt{A}}$$

$$(snd(\langle M_1^{\mathtt{A}} M_2^{\mathtt{B}}\rangle)^{\mathtt{A}\times\mathtt{B}})^{\mathtt{B}} \to_\beta M_2^{\mathtt{B}}$$

Le regole di riduzione nel λ-calcolo semplicemente tipato

$$((\lambda x^{\mathtt{A}}.M^{\mathtt{B}})^{\mathtt{A}\to\mathtt{B}} N^{\mathtt{A}})^{\mathtt{B}} \to_\beta M^{\mathtt{B}}[N^{\mathtt{A}}/x^{\mathtt{A}}]$$

$$(fst(\langle M_1^{\mathtt{A}} M_2^{\mathtt{B}}\rangle)^{\mathtt{A}\times\mathtt{B}})^{\mathtt{A}} \to_\beta M_1^{\mathtt{A}}$$

$$(snd(\langle M_1^{\mathtt{A}} M_2^{\mathtt{B}}\rangle)^{\mathtt{A}\times\mathtt{B}})^{\mathtt{B}} \to_\beta M_2^{\mathtt{B}}$$

Non è possibile assegnare un tipo al termine $\lambda x.xx$ nel λ-calcolo semplicemente tipato: infatti non si può assegnare ad x allo stesso tempo il tipo $\mathtt{A} \to \mathtt{B}$ e $\mathtt{A}$.

Proprietà fondamentali del λ-calcolo semplicemente tipato

Subject reduction

Se M è un termine di tipo $\mathtt{A}$ del λ-calcolo semplicemente tipato e $M \to_\beta M'$, allora M' è un termine di tipo $\mathtt{A}$.

Proprietà fondamentali del λ-calcolo semplicemente tipato

Subject reduction

Se M è un termine di tipo $\mathtt{A}$ del λ-calcolo semplicemente tipato e $M \to_\beta M'$, allora M' è un termine di tipo $\mathtt{A}$.

Normalizzazione forte

Se M è un termine del λ-calcolo semplicemente tipato, allora M è fortemente normalizzabile.

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Formule / Tipi

Formule	Tipi
A	$\mathtt{A}$

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Formule / Tipi

Formule	Tipi
A	$\mathtt{A}$
$A \Rightarrow B$	$\mathtt{A} \to \mathtt{B}$

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Formule / Tipi

Formule	Tipi
A	$\mathtt{A}$
$A \Rightarrow B$	$\mathtt{A} \to \mathtt{B}$
$A \wedge B$	$\mathtt{A} \times \mathtt{B}$

Regole di deduzione e derivazione di tipo: l'ipotesi

$$x : A$$

Regole di deduzione e derivazione di tipo: l'ipotesi

$$x : A$$

Sotto l'ipotesi che x ha tipo A è possibile concludere che x ha tipo A.

Regole di deduzione e derivazione di tipo: la regola di $\Rightarrow\mathcal{I}$

$$\begin{array}{c} x : A \\ \vdots \\ M : B \end{array}$$

- Supponiamo di avere un λ-termine M di tipo B in cui compare un certo numero di volte la variabile libera x di tipo A;

Regole di deduzione e derivazione di tipo: la regola di $\Rightarrow \mathcal{I}$

$$\frac{\begin{array}{c}[x : A] \\ \vdots \\ M : B\end{array}}{\lambda x.M : A \Rightarrow B} \Rightarrow \mathcal{I}$$

- Supponiamo di avere un λ-termine M di tipo B in cui compare un certo numero di volte la variabile libera x di tipo A;
- allora possiamo "catturare" le occorrenze libere della variabile x di tipo A con una λ-astrazione, ottenendo un termine $\lambda x.M$ di tipo $A \Rightarrow B$.

Regole di deduzione e derivazione di tipo: la regola di $\Rightarrow \mathcal{E}$

$$\begin{array}{cc} \vdots & \vdots \\ M : A \Rightarrow B & N : A \end{array}$$

- Supponiamo di avere un λ-termine M di tipo $A \Rightarrow B$ e un λ termine N di tipo A;

Regole di deduzione e derivazione di tipo: la regola di $\Rightarrow \mathcal{E}$

$$\frac{\begin{array}{cc}\vdots & \vdots \\ M : A \Rightarrow B & N : A\end{array}}{(M\ N) : B} \Rightarrow \mathcal{E}$$

- Supponiamo di avere un λ-termine M di tipo $A \Rightarrow B$ e un λ-termine N di tipo A;
- allora possiamo applicare la funzione M al suo argomento N per produrre un λ-termine di tipo B.

Regole di deduzione e derivazione di tipo: la regola di $\wedge\mathcal{I}$

$$\begin{array}{cc} \vdots & \vdots \\ M : A & N : B \end{array}$$

- Supponiamo di avere due λ-termini M e N rispettivamente di tipo A e B;

Regole di deduzione e derivazione di tipo: la regola di $\wedge\mathcal{I}$

$$\frac{\begin{array}{cc}\vdots & \vdots \\ M : A & N : B\end{array}}{\langle M\ N\rangle : A \wedge B}\ \wedge\mathcal{I}$$

- Supponiamo di avere due λ-termini M e N rispettivamente di tipo A e B;
- allora possiamo comporre i due λ-termini per ottenere la coppia $\langle M\ N\rangle$ di tipo $A \wedge B$.

Regole di deduzione e derivazione di tipo: la regola di $\wedge\mathcal{E}1$

$$\begin{array}{c} \vdots \\ M : A \wedge B \end{array}$$

- Supponiamo di avere un λ-termine di tipo $A \wedge B$;

Regole di deduzione e derivazione di tipo: la regola di $\wedge\mathcal{E}1$

$$\frac{\begin{array}{c}\vdots\\ M : A \wedge B\end{array}}{(fst\ M) : A}\ \wedge\mathcal{E}1$$

- Supponiamo di avere un λ-termine di tipo $A \wedge B$;
- allora possiamo "selezionare" il primo elemento di M ed ottenere così un λ-termine $fst\ M$ di tipo A.

Regole di deduzione e derivazione di tipo: la regola di $\wedge\mathcal{E}2$

$$\begin{array}{c} \vdots \\ M : A \wedge B \end{array}$$

- Supponiamo di avere un λ-termine di tipo $A \wedge B$;

Regole di deduzione e derivazione di tipo: la regola di $\wedge\mathcal{E}2$

$$\frac{\begin{array}{c}\vdots\\ M : A \wedge B\end{array}}{(snd\ M) : B}\ \wedge\mathcal{E}2$$

- Supponiamo di avere un λ-termine di tipo $A \wedge B$;
- allora possiamo "selezionare" il secondo elemento di M ed ottenere così un λ-termine $snd\ M$ di tipo B.

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Regole di deduzione / Costrutti del λ-calcolo

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Regole di deduzione / Costrutti del λ-calcolo

Regole	Costrutti
$\Rightarrow \mathcal{I}$	λ-astrazione

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Regole di deduzione / Costrutti del λ-calcolo

Regole	Costrutti
$\Rightarrow \mathcal{I}$	λ-astrazione
$\Rightarrow \mathcal{E}$	applicazione

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Regole di deduzione / Costrutti del λ-calcolo

Regole	Costrutti
$\Rightarrow \mathcal{I}$	λ-astrazione
$\Rightarrow \mathcal{E}$	applicazione
$\wedge \mathcal{I}$	coppia

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

- Regole di deduzione / Costrutti del λ-calcolo

Regole	Costrutti
$\Rightarrow \mathcal{I}$	λ-astrazione
$\Rightarrow \mathcal{E}$	applicazione
$\wedge \mathcal{I}$	coppia
$\wedge \mathcal{E}1$, $\wedge \mathcal{E}2$	prima e seconda proiezione

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\dfrac{\dfrac{\dfrac{\dfrac{[z : B \wedge A]}{snd\ z : A}\wedge\mathcal{E}2 \quad \dfrac{[z : B \wedge A]}{fst\ z : B}\wedge\mathcal{E}1}{\langle (snd\ z)(fst\ z)\rangle : A \wedge B}\wedge\mathcal{I}}{\lambda z.\langle (snd\ z)(fst\ z)\rangle : (B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \quad \dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{(\lambda z.\langle (snd\ z)(fst\ z)\rangle)\langle y\ x\rangle : A \wedge B}\Rightarrow\mathcal{E}$$

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\dfrac{\dfrac{\dfrac{\dfrac{[z : B \wedge A]}{snd\ z : A}\wedge\mathcal{E}2 \quad \dfrac{[z : B \wedge A]}{fst\ z : B}\wedge\mathcal{E}1}{\langle (snd\ z)(fst\ z)\rangle : A \wedge B}\wedge\mathcal{I}}{\lambda z.\langle (snd\ z)(fst\ z)\rangle : (B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \quad \dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{(\lambda z.\langle (snd\ z)(fst\ z)\rangle)\langle y\ x\rangle : A \wedge B}\Rightarrow\mathcal{E}$$

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\dfrac{\dfrac{\dfrac{\dfrac{[z : B \wedge A]}{snd\ z : A}\wedge\mathcal{E}2 \quad \dfrac{[z : B \wedge A]}{fst\ z : B}\wedge\mathcal{E}1}{\langle (snd\ z)(fst\ z)\rangle : A \wedge B}\wedge\mathcal{I}}{\lambda z.\langle (snd\ z)(fst\ z)\rangle : (B \wedge A) \Rightarrow (A \wedge B)}\Rightarrow\mathcal{I} \quad \dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{(\lambda z.\langle (snd\ z)(fst\ z)\rangle)\langle y\ x\rangle : A \wedge B}\Rightarrow\mathcal{E}$$

I redessi corrispondono ai tagli!

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\dfrac{\dfrac{\dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{snd\ \langle y\ x\rangle : A}\wedge\mathcal{E}2 \qquad \dfrac{\dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{fst\ \langle y\ x\rangle : B}\wedge\mathcal{E}1}{\langle (snd\ \langle y\ x\rangle)(fst\ \langle y\ x\rangle)\rangle : A \wedge B}\wedge\mathcal{I}$$

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\dfrac{\dfrac{\dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{snd\ \langle y\ x\rangle : A}\wedge\mathcal{E}2 \qquad \dfrac{\dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{fst\ \langle y\ x\rangle : B}\wedge\mathcal{E}1}{\langle (snd\ \langle y\ x\rangle)(fst\ \langle y\ x\rangle)\rangle : A \wedge B}\wedge\mathcal{I}$$

Un passo di β-riduzione corrisponde ad un passo di eliminazione del taglio !

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\dfrac{\dfrac{\dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{\mathit{snd}\ \langle y\ x\rangle : A}\wedge\mathcal{E}2 \qquad \dfrac{\dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\wedge\mathcal{I}}{\mathit{fst}\ \langle y\ x\rangle : B}\wedge\mathcal{E}1}{\langle(\mathit{snd}\ \langle y\ x\rangle)(\mathit{fst}\ \langle y\ x\rangle)\rangle : A \wedge B}\wedge\mathcal{I}$$

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\dfrac{x : A \qquad \dfrac{\dfrac{y : B \qquad x : A}{\langle y\ x\rangle : B \wedge A}\,\wedge\mathcal{I}}{\mathit{fst}\ \langle y\ x\rangle : B}\,\wedge\mathcal{E}1}{\langle (x)(\mathit{fst}\ \langle y\ x\rangle)\rangle : A \wedge B}\,\wedge\mathcal{I}$$

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\frac{x : A \qquad y : B}{\langle x\ y \rangle : A \wedge B}\ \wedge\mathcal{I}$$

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\frac{x : A \qquad y : B}{\langle x\ y \rangle : A \wedge B}\ \wedge\mathcal{I}$$

Un λ-termine in forma normale corrisponde ad una dimostrazione senza tagli !

Un λ-termine corrisponde ad una prova in deduzione naturale!

$$\begin{array}{ccc} M & \longrightarrow & M' \\ \downarrow & & \downarrow \\ \pi & \longrightarrow & \pi' \end{array}$$

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

Deduzione Naturale	λ-calcolo
Formule	Tipi

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

Deduzione Naturale	λ-calcolo
Formule	Tipi
Prove	λ-termini

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

Deduzione Naturale	λ-calcolo
Formule	Tipi
Prove	λ-termini
Eliminazione del Taglio	β-riduzione

L'isomorfismo di Curry-Howard

Esiste una corrispondenza stretta tra termini del λ calcolo semplicemente tipato e prove della deduzione naturale intuizionista.

Deduzione Naturale	λ-calcolo
Formule	Tipi
Prove	λ-termini
Eliminazione del Taglio	β-riduzione
Prove senza tagli	λ-termini in forma normale

Costruire un programma da una prova

- Supponiamo di volere calcolare una determinata funzione f sugli interi, definita mediante un sistema di equazioni Eq;

Costruire un programma da una prova

- Supponiamo di volere calcolare una determinata funzione f sugli interi, definita mediante un sistema di equazioni Eq;
- Costruiamo una prova π in deduzione naturale della proposizione " Se x è un intero, allora $f(x)$ è un intero" (cioé dimostriamo che la funzione f è totale);

Costruire un programma da una prova

- Supponiamo di volere calcolare una determinata funzione f sugli interi, definita mediante un sistema di equazioni Eq;
- Costruiamo una prova π in deduzione naturale della proposizione " Se x è un intero, allora $f(x)$ è un intero" (cioé dimostriamo che la funzione f è totale);
- Utilizzando la corrispondenza prove/programmi, associamo alla dimostrazione π il λ termine M_f: in questo modo otteniamo un programma capace di calcolare la funzione f!

Vantaggi del calcolare con le prove

- Qualunque sia la strategia utilizzata per eseguire un programma associato ad una prova l'esecuzione termina (teorema di normalizzazione forte).

Vantaggi del calcolare con le prove

- Qualunque sia la strategia utilizzata per eseguire un programma associato ad una prova l'esecuzione termina (teorema di normalizzazione forte).
- Due strategie diverse di esecuzione di un programma associato ad una prova conducono allo stesso risultato (teorema di confluenza).

Vantaggi del calcolare con le prove

- Qualunque sia la strategia utilizzata per eseguire un programma associato ad una prova l'esecuzione termina (teorema di normalizzazione forte).
- Due strategie diverse di esecuzione di un programma associato ad una prova conducono allo stesso risultato (teorema di confluenza).
- Un programma associato ad una prova è *sempre corretto*, nel senso che il programma M_f estratto dalla prova π calcola effettivamente i valori della funzione definita da Eq (teorema di correttezza).

Oltre i tipi semplici

Il λ-calcolo semplicemente tipato ha un espressività limitata...

Oltre i tipi semplici

Il λ-calcolo semplicemente tipato ha un espressività limitata...

- Estensione dei tipi: il λ-calcolo con tipi polimorfi corrisponde alla logica intuizionista del secondo ordine.

Oltre i tipi semplici

Il λ-calcolo semplicemente tipato ha un espressività limitata...

- Estensione dei tipi: il λ-calcolo con tipi polimorfi corrisponde alla logica intuizionista del secondo ordine.
- Estensione del linguaggio : il λ-calcolo con operatori di controllo corrisponde alla logica classica.

Oltre i tipi semplici

Il λ-calcolo semplicemente tipato ha un espressività limitata...

- Estensione dei tipi: il λ-calcolo con tipi polimorfi corrisponde alla logica intuizionista del secondo ordine.
- Estensione del linguaggio : il λ-calcolo con operatori di controllo corrisponde alla logica classica.

Indice

Sul potere espressivo dei tipi semplici

Possiamo individuare due limiti fondamentali al sistema dei tipi semplici:

Sul potere espressivo dei tipi semplici

Possiamo individuare due limiti fondamentali al sistema dei tipi semplici:

- Non è possibile fare astrazione sui tipi (es. ogni tipo A ha la "sua" funzione identità $\lambda x.x \ : A \Rightarrow A$);

Sul potere espressivo dei tipi semplici

Possiamo individuare due limiti fondamentali al sistema dei tipi semplici:

- Non è possibile fare astrazione sui tipi (es. ogni tipo A ha la "sua" funzione identità $\lambda x.x \ : A \Rightarrow A$);
- non è possibile applicare un termine a se stesso ($x\ x$ non è tipabile, perché x non puó avere al tempo stesso il tipo $A \Rightarrow B$ e il tipo A)

Logica del secondo ordine

Sia data una proposizione P ; allora vale che $P \Rightarrow P$.
Come esprimere questa proprietà nella logica?

Logica del secondo ordine

Sia data una proposizione P ; allora vale che $P \Rightarrow P$.
Come esprimere questa proprietà nella logica?

- Sia data un'infinità numerabile di variabili proposizionali $X, Y, Z, \ldots$;

Logica del secondo ordine

Sia data una proposizione P ; allora vale che $P \Rightarrow P$.
Come esprimere questa proprietà nella logica?

- Sia data un'infinità numerabile di variabili proposizionali $X, Y, Z, \ldots$;
- per ogni X, $X \Rightarrow X$;

Logica del secondo ordine

Sia data una proposizione P ; allora vale che $P \Rightarrow P$.
Come esprimere questa proprietà nella logica?

- Sia data un'infinità numerabile di variabili proposizionali $X, Y, Z, \ldots$;
- per ogni X, $X \Rightarrow X$;
- $\forall X\ X \Rightarrow X$;

Deduzione naturale: la regola di $\forall^2\mathcal{I}$

$$\dfrac{\begin{array}{c}\vdots\\ A\end{array}}{\forall X\ A}\ \forall^2\mathcal{I}$$

- Supponiamo di avere una dimostrazione della formula A, dove la variabile proposizionale X non occora libera nelle ipotesi (cioè sto usando la variabile X in modo *generico*);

Deduzione naturale: la regola di $\forall^2\mathcal{I}$

$$\dfrac{\begin{array}{c}\vdots\\ A\end{array}}{\forall X\ A}\ \forall^2\mathcal{I}$$

- Supponiamo di avere una dimostrazione della formula A, dove la variabile proposizionale X non occora libera nelle ipotesi (cioè sto usando la variabile X in modo *generico*);
- allora posso "astrarre" rispetto ad X e concludere $\forall X\ A$.

Deduzione naturale: la regola di $\forall^2\mathcal{E}$

$$\frac{\begin{array}{c}\vdots\\ \forall X\ A\end{array}}{A[B/X]}\ \forall^2\mathcal{E}$$

- Supponiamo di avere una dimostrazione della formula $\forall X\ A$;

Deduzione naturale: la regola di $\forall^2\mathcal{E}$

$$\dfrac{\begin{array}{c}\vdots\\ \forall X\ A\end{array}}{A[B/X]}\ \forall^2\mathcal{E}$$

- Supponiamo di avere una dimostrazione della formula $\forall X\ A$;
- allora posso "usare" questa dimostrazione sostituendo in A ogni occorrenza di X con una qualsiasi formula B, poiché X viene usata in A in modo generico.

Eliminazione del taglio: il caso $\forall^2$

$$\dfrac{\dfrac{\begin{array}{c}\pi_1\\ \vdots\\ A\end{array}}{\forall X\ A}\,\forall^2\mathcal{I}}{A[B/X]}\,\forall^2\mathcal{E} \quad \rightsquigarrow \quad \begin{array}{c}\pi_1\\ \vdots\\ A[B/X]\end{array}$$

Esempio

$$X$$

Esempio

$$\frac{[X]}{X \Rightarrow X} \Rightarrow \mathcal{I}$$

Esempio

$$\dfrac{\dfrac{[X]}{X \Rightarrow X}\Rightarrow\mathcal{I}}{\forall X\ X \Rightarrow X}\forall^2\mathcal{I}$$

Esempio

$$\dfrac{\dfrac{\dfrac{[X]}{X \Rightarrow X}\Rightarrow\mathcal{I}}{\forall X\ X \Rightarrow X}\forall^2\mathcal{I}}{B \Rightarrow B}\forall^2\mathcal{E}$$

Il Sistema F: tipi

- Sia data una infinità numerabile di variabili $X, Y, Z \ldots$ dette *variabili* di tipo:

Il Sistema F: tipi

- Sia data una infinità numerabile di variabili $X, Y, Z \ldots$ dette *variabili* di tipo:
  1. Una variabile di tipo X è un tipo;

Il Sistema F: tipi

- Sia data una infinità numerabile di variabili $X, Y, Z \ldots$ dette *variabili* di tipo:
  1. Una variabile di tipo X è un tipo;
  2. Se A è un tipo e B è un tipo, allora $A \Rightarrow B$ è un tipo;

Il Sistema F: tipi

- Sia data una infinità numerabile di variabili $X, Y, Z \ldots$ dette *variabili* di tipo:
  1. Una variabile di tipo X è un tipo;
  2. Se A è un tipo e B è un tipo, allora $A \Rightarrow B$ è un tipo;
  3. Se A è un tipo e X è una variabile di tipo, allora $\forall X\ A$ è un tipo.

Il Sistema F: termini

- Tipi:

$$U \quad ::= \quad X \quad | \quad U \Rightarrow U \quad | \quad \forall X \; U$$

- λ-termini:

$$M \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad M \; M \quad | \quad \Lambda X.M \quad | \quad M \; U$$

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo A:

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo A:
- Se x è una variabile con tipo A e M è un termine di tipo B allora $\lambda x.M$ è un termine di tipo $A \Rightarrow B$;

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo A:
- Se x è una variabile con tipo A e M è un termine di tipo B allora $\lambda x.M$ è un termine di tipo $A \Rightarrow B$;
- Se M è un termine di tipo $A \Rightarrow B$ e N è un termine di tipo A allora $M\ N$ è un termine di tipo B.

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo A:
- Se x è una variabile con tipo A e M è un termine di tipo B allora $\lambda x.M$ è un termine di tipo $A \Rightarrow B$;
- Se M è un termine di tipo $A \Rightarrow B$ e N è un termine di tipo A allora $M\ N$ è un termine di tipo B.
- Se M è un termine di tipo A, e se la variabile di tipo X non occorre libera nel tipo di una variabile libera di M, allora $\Lambda X.M$ è un termine di tipo $\forall XA$;

Assegnazione di un tipo ad un termine

- Ogni variabile x ha assegnato un tipo A:
- Se x è una variabile con tipo A e M è un termine di tipo B allora $\lambda x.M$ è un termine di tipo $A \Rightarrow B$;
- Se M è un termine di tipo $A \Rightarrow B$ e N è un termine di tipo A allora $M\ N$ è un termine di tipo B.
- Se M è un termine di tipo A, e se la variabile di tipo X non occorre libera nel tipo di una variabile libera di M, allora $\Lambda X.M$ è un termine di tipo $\forall XA$;
- Se M è un termine di tipo $\forall XA$ e se B è un tipo, allora $M\ B$ è un termine di tipo $A[B/X]$.

Le regole di riduzione nel Sistema F

$$((\lambda x^A.M^B)^{A\Rightarrow B}N^A)^B \rightarrow_\beta M^B[N^A/x^A]$$

Le regole di riduzione nel Sistema F

$$((\lambda x^A.M^B)^{A\Rightarrow B}N^A)^B \rightarrow_\beta M^B[N^A/x^A]$$

$$((\Lambda X.M^A)^{\forall X A}B)^{A[B/X]} \rightarrow_\beta M^{A[B/X]}$$

Isomorfismo di Curry-Howard: la regola di $\forall^2\mathcal{I}$

$$\begin{array}{c} \vdots \\ M : A \end{array}$$

- Supponiamo di avere un λ-termine M di tipo A, e che nel tipo delle variabili libere di M (i.e. le ipotesi) non occorra la variabile di tipo X;

Isomorfismo di Curry-Howard: la regola di $\forall^2\mathcal{I}$

$$\dfrac{\begin{array}{c}\vdots\\ M : A\end{array}}{\Lambda X.M : \forall XA}\ \wedge\mathcal{I}$$

- Supponiamo di avere un λ-termine M di tipo A, e che nel tipo delle variabili libere di M (i.e. le ipotesi) non occorra la variabile di tipo X;
- allora possiamo astrarre rispetto alla variabile di tipo X ed ottenere un termine $\Lambda X.M$ di tipo $\forall XA$.

Isomorfismo di Curry-Howard: la regola di $\forall^2\mathcal{E}$

$$\begin{array}{c} \vdots \\ M : \forall XA \end{array}$$

- Supponiamo di avere un λ-termine M di tipo $\forall XA$;

Isomorfismo di Curry-Howard: la regola di $\forall^2\mathcal{E}$

$$\dfrac{\begin{array}{c}\vdots\\ M : \forall X A\end{array}}{M\ B : A[B/X]}\ \forall^2\mathcal{E}$$

- Supponiamo di avere un λ-termine M di tipo $\forall X A$;
- allora possiamo instanziare le occorrenze della variabile di tipo X con un tipo B a nostra scelta.

Eliminazione del taglio e riduzione

$$\dfrac{\dfrac{\begin{array}{c}\pi_1\\ \vdots\\ M : A\end{array}}{\Lambda X.M : \forall X\ A}\,\forall^2\mathcal{I}}{\Lambda X.M\ B : A[B/X]}\,\forall^2\mathcal{E} \quad \rightsquigarrow \quad \begin{array}{c}\pi_1\\ \vdots\\ M : A[B/X]\end{array}$$

Polimorfismo

$$x : X$$

Polimorfismo

$$\frac{[x : X]}{\lambda x.x : X \Rightarrow X} \Rightarrow \mathcal{I}$$

Polimorfismo

$$\dfrac{\dfrac{[x : X]}{\lambda x.x : X \Rightarrow X}\Rightarrow\mathcal{I}}{\Lambda X.\lambda x.x : \forall X\ X \Rightarrow X}\forall^2\mathcal{I}$$

Polimorfismo

$$\dfrac{\dfrac{\dfrac{[x : X]}{\lambda x.x : X \Rightarrow X}\Rightarrow\mathcal{I}}{\Lambda X.\lambda x.x : \forall X\ X \Rightarrow X}\forall^2\mathcal{I}}{\Lambda X.\lambda x.x\ B : B \Rightarrow B}\forall^2\mathcal{E}$$

Polimorfismo

$$\frac{[x : B]}{\lambda x.x : B \Rightarrow B} \Rightarrow \mathcal{I}$$

Auto applicazione

$$\dfrac{\dfrac{\dfrac{[x : \forall X(X \Rightarrow X)]}{x\ \forall X(X \Rightarrow X) : (\forall X(X \Rightarrow X)) \Rightarrow (\forall X(X \Rightarrow X))}\ \forall^2\mathcal{E} \qquad [x : \forall X(X \Rightarrow X)]}{(x\ \forall X(X \Rightarrow X))\ x : \forall X(X \Rightarrow X)}\ \Rightarrow \mathcal{E}}{\lambda x.((x\ \forall X(X \Rightarrow X))\ x) : (\forall X(X \Rightarrow X)) \Rightarrow (\forall X(X \Rightarrow X))}\ \Rightarrow \mathcal{I}$$

$$\vdots$$

Auto applicazione

$$\dfrac{\dfrac{\vdots}{\lambda x.((x\ \forall X(X \Rightarrow X))\ x) : (\forall X(X \Rightarrow X)) \Rightarrow (\forall X(X \Rightarrow X))} \qquad \dfrac{\dfrac{\dfrac{[y : X]}{\lambda y.y : X \Rightarrow X} \Rightarrow \mathcal{I}}{\Lambda X.\lambda y.y : \forall X\ X \Rightarrow X} \forall^2\mathcal{I}}{(\lambda x.((x\ \forall X(X \Rightarrow X))\ x))\ \Lambda X.\lambda y.y : \forall X\ X \Rightarrow X} \Rightarrow \mathcal{E}$$

Auto applicazione

$$(\lambda x.((x\ \forall X(X \Rightarrow X))\ x))\ \Lambda X.\lambda y.y$$

Auto applicazione

$$\begin{array}{ll} & (\lambda x.((x\ \forall X(X \Rightarrow X))\ x))\ \Lambda X.\lambda y.y \\ \rightarrow_\beta & (\Lambda X.\lambda y.y\ \forall X(X \Rightarrow X))\ \Lambda X.\lambda y.y \end{array}$$

Auto applicazione

$$
\begin{array}{ll}
 & (\lambda x.((x\ \forall X(X \Rightarrow X))\ x))\ \Lambda X.\lambda y.y \\
\rightarrow_\beta & (\Lambda X.\lambda y.y\ \forall X(X \Rightarrow X))\ \Lambda X.\lambda y.y \\
\rightarrow_\beta & (\lambda y.y)\ \Lambda X.\lambda y.y
\end{array}
$$

Auto applicazione

$$
\begin{array}{ll}
 & (\lambda x.((x\ \forall X(X \Rightarrow X))\ x))\ \Lambda X.\lambda y.y \\
\rightarrow_\beta & (\Lambda X.\lambda y.y\ \forall X(X \Rightarrow X))\ \Lambda X.\lambda y.y \\
\rightarrow_\beta & (\lambda y.y)\ \Lambda X.\lambda y.y \\
\rightarrow_\beta & \Lambda X.\lambda y.y
\end{array}
$$

Il funtore "smemorato"

$$\begin{array}{ccc} (\lambda x.((x\ \forall X(X \Rightarrow X))\ x))\ \Lambda X.\lambda y.y & \longrightarrow & \Lambda X.\lambda y.y \\ \downarrow & & \downarrow \\ (\lambda x.((x)\ x))\ \lambda y.y & \longrightarrow & \lambda y.y \end{array}$$

Tipi di dati: `Bool`

$$\texttt{Bool} = \forall X(X \Rightarrow (X \Rightarrow X))$$

Tipi di dati: Bool

$$\texttt{Bool} = \forall X(X \Rightarrow (X \Rightarrow X))$$

- Costruttori:

$$\texttt{True} = \Lambda X.\lambda x^X.\lambda y^X.x : \texttt{Bool}$$

Tipi di dati: Bool

$$\texttt{Bool} = \forall X(X \Rightarrow (X \Rightarrow X))$$

- Costruttori:

$$\texttt{True} = \Lambda X.\lambda x^X.\lambda y^X.x : \texttt{Bool}$$

$$\texttt{False} = \Lambda X.\lambda x^X.\lambda y^X.y : \texttt{Bool}$$

- Distruttore :
  siano M, N, T di tipo rispettivamente U, U e Bool;

$$\texttt{If } T \texttt{ then } M \texttt{ else } N = T\ U\ M\ N : U$$

Tipi di dati: `Bool`

$$\texttt{If True then } M \texttt{ else } N \quad = \quad (\Lambda X.\lambda x^X.\lambda y^X.x)\ U\ M\ N$$

Tipi di dati: `Bool`

$$\begin{array}{lcl} \texttt{If True then } M \texttt{ else } N & = & (\Lambda X.\lambda x^X.\lambda y^X.x)\ U\ M\ N \\ & \to_\beta & (\lambda x^U.\lambda y^U.x)\ M\ N \end{array}$$

Tipi di dati: `Bool`

$$\begin{array}{lcl} \texttt{If True then } M \texttt{ else } N & = & (\Lambda X.\lambda x^X.\lambda y^X.x)\ U\ M\ N \\ & \to_\beta & (\lambda x^U.\lambda y^U.x)\ M\ N \\ & \to_\beta & (\lambda y^U.M)\ N \end{array}$$

Tipi di dati: Bool

$$
\begin{array}{lcc}
\texttt{If True then } M \texttt{ else } N & = & (\Lambda X.\lambda x^X.\lambda y^X.x)\ U\ M\ N \\
& \to_\beta & (\lambda x^U.\lambda y^U.x)\ M\ N \\
& \to_\beta & (\lambda y^U.M)\ N \\
& \to_\beta & M
\end{array}
$$

Tipi di dati: `Bool`

$$\texttt{If False then } M \texttt{ else } N \quad = \quad (\Lambda X.\lambda x^X.\lambda y^X.y)\ U\ M\ N$$

Tipi di dati: `Bool`

$$\begin{array}{lcl} \texttt{If False then } M \texttt{ else } N & = & (\Lambda X.\lambda x^X.\lambda y^X.y)\ U\ M\ N \\ & \to_\beta & (\lambda x^U.\lambda y^U.y)\ M\ N \end{array}$$

Tipi di dati: `Bool`

$$\begin{array}{lcl} \texttt{If False then } M \texttt{ else } N & = & (\Lambda X.\lambda x^X.\lambda y^X.y)\ U\ M\ N \\ & \to_\beta & (\lambda x^U.\lambda y^U.y)\ M\ N \\ & \to_\beta & (\lambda y^U.y)\ N \end{array}$$

Tipi di dati: `Bool`

$$
\begin{array}{lcc}
\texttt{If False then } M \texttt{ else } N & = & (\Lambda X.\lambda x^X.\lambda y^X.y)\ U\ M\ N \\
& \to_\beta & (\lambda x^U.\lambda y^U.y)\ M\ N \\
& \to_\beta & (\lambda y^U.y)\ N \\
& \to_\beta & N
\end{array}
$$

Tipi di dati: il tipo coppia

$$U \times V = \forall X(U \Rightarrow V \Rightarrow X) \Rightarrow X$$

Tipi di dati: il tipo coppia

$$U \times V = \forall X(U \Rightarrow V \Rightarrow X) \Rightarrow X$$

- Costruttore:

$$\langle M\ N \rangle = \Lambda X.\lambda x^{U \Rightarrow V \Rightarrow X}.x\ M\ N : U \times V$$

Tipi di dati: il tipo coppia

$$U \times V = \forall X(U \Rightarrow V \Rightarrow X) \Rightarrow X$$

- Costruttore:

$$\langle M\ N\rangle = \Lambda X.\lambda x^{U \Rightarrow V \Rightarrow X}.x\ M\ N : U \times V$$

- Distruttori :
  Sia P di tipo $U \times V$:

$$\pi^1 P = P\ U(\lambda x^U.\lambda y^V.x) : U$$

Tipi di dati: il tipo coppia

$$U \times V = \forall X(U \Rightarrow V \Rightarrow X) \Rightarrow X$$

- Costruttore:

$$\langle M\ N\rangle = \Lambda X.\lambda x^{U \Rightarrow V \Rightarrow X}.x\ M\ N : U \times V$$

- Distruttori :
  Sia P di tipo $U \times V$:

$$\pi^1 P = P\ U(\lambda x^U.\lambda y^V.x) : U$$

$$\pi^2 P = P\ V(\lambda x^U.\lambda y^V.y) : V$$

Tipi di dati: il tipo coppia

$$\pi^1\langle M\ N\rangle \quad = \quad (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.x)$$

Tipi di dati: il tipo coppia

$$\begin{array}{rcl} \pi^1\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.x) \\ & \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^U.\lambda y^V.x) \end{array}$$

Tipi di dati: il tipo coppia

$$\begin{array}{rcl} \pi^1\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.x) \\ & \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^U.\lambda y^V.x) \\ & \to_\beta & (\lambda x^U.\lambda y^V.x)\ M\ N \end{array}$$

Tipi di dati: il tipo coppia

$$
\begin{array}{lcl}
\pi^1\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.x) \\
& \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^U.\lambda y^V.x) \\
& \to_\beta & (\lambda x^U.\lambda y^V.x)\ M\ N \\
& \to_\beta & (\lambda y^V.M)\ N
\end{array}
$$

Tipi di dati: il tipo coppia

$$
\begin{array}{lcl}
\pi^1\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.x) \\
& \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^U.\lambda y^V.x) \\
& \to_\beta & (\lambda x^U.\lambda y^V.x)\ M\ N \\
& \to_\beta & (\lambda y^V.M)\ N \\
& \to_\beta & M
\end{array}
$$

Tipi di dati: il tipo coppia

$$\pi^2\langle M\ N\rangle \quad = \quad (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.y)$$

Tipi di dati: il tipo coppia

$$\begin{array}{rcl} \pi^2\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.y) \\ & \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^U.\lambda y^V.y) \end{array}$$

Tipi di dati: il tipo coppia

$$\begin{array}{lcl} \pi^2\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.y) \\ & \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^U.\lambda y^V.y) \\ & \to_\beta & (\lambda x^U.\lambda y^V.y)\ M\ N \end{array}$$

Tipi di dati: il tipo coppia

$$\begin{array}{lcl} \pi^2\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^U.\lambda y^V.y) \\ & \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^U.\lambda y^V.y) \\ & \to_\beta & (\lambda x^U.\lambda y^V.y)\ M\ N \\ & \to_\beta & (\lambda y^V.y)\ N \end{array}$$

Tipi di dati: il tipo coppia

$$
\begin{array}{lcc}
\pi^2\langle M\ N\rangle & = & (\Lambda X.\lambda x^{U\Rightarrow V\Rightarrow X}.x\ M\ N)\ U(\lambda x^{U}.\lambda y^{V}.y) \\
& \to_\beta & (\lambda x^{U\Rightarrow V\Rightarrow U}.x\ M\ N)(\lambda x^{U}.\lambda y^{V}.y) \\
& \to_\beta & (\lambda x^{U}.\lambda y^{V}.y)\ M\ N \\
& \to_\beta & (\lambda y^{V}.y)\ N \\
& \to_\beta & N
\end{array}
$$

Tipi di dati: il tipo $\bot$

$$\bot = \forall X\ X$$

- Il tipo $\bot$ non è abitato.

Tipi di dati: il tipo Nat

$$\mathtt{Nat} = \forall X(X \Rightarrow ((X \Rightarrow X) \Rightarrow X))$$

- Costruttori:

$$0 = \Lambda X.\lambda x^X.\lambda y^{X \Rightarrow X}.x : \mathtt{Nat}$$

Tipi di dati: il tipo Nat

$$\mathtt{Nat} = \forall X(X \Rightarrow ((X \Rightarrow X) \Rightarrow X))$$

- Costruttori:

$$0 = \Lambda X.\lambda x^X.\lambda y^{X \Rightarrow X}.x : \mathtt{Nat}$$

Dato N di tipo Nat allora

$$\mathtt{Succ}\ N = \Lambda X.\lambda x.^X.\lambda y^{X \Rightarrow X}.y(N\ X\ x\ y) : \mathtt{Nat}$$

Tipi di dati: il tipo Nat

$$\mathtt{Nat} = \forall X(X \Rightarrow ((X \Rightarrow X) \Rightarrow X))$$

- Costruttori:

$$0 = \Lambda X.\lambda x^X.\lambda y^{X \Rightarrow X}.x : \mathtt{Nat}$$

Dato N di tipo Nat allora

$$\mathtt{Succ}\ N = \Lambda X.\lambda x.^X.\lambda y^{X \Rightarrow X}.y(N\ X\ x\ y) : \mathtt{Nat}$$

- Distruttori:
Sia P di tipo U, Q di tipo $U \Rightarrow U$ ed N di tipo Nat:

$$\mathtt{It}\ P\ Q\ N = N\ U\ P\ Q : U$$

Tipi di dati: il tipo Nat

$$\underline{n} := \Lambda X.\lambda x^X \lambda y^{X \Rightarrow X}.\overbrace{y(y \cdots (y}^{n \text{ volte}} x) \cdots)$$

Tipi di dati: il tipo Nat

$$\underline{n} := \Lambda X.\lambda x^X \lambda y^{X \Rightarrow X}.\overbrace{y(y \cdots (y}^{n \text{ volte}} x) \cdots)$$

$$\underline{0} := 0$$

$$\underline{n+1} := \mathtt{Succ}\ \underline{n}$$

Tipi di dati: il tipo Nat

$$\texttt{It}\ P\ Q\ 0 = \quad = \quad (\Lambda X.\lambda x^X.\lambda y^{X \Rightarrow X}.x)\ U\ P\ Q$$

Tipi di dati: il tipo Nat

$$\begin{array}{lcl} \texttt{It}\ P\ Q\ 0 = & = & (\Lambda X.\lambda x^X.\lambda y^{X\Rightarrow X}.x)\ U\ P\ Q \\ & \to_\beta & (\lambda x^U.\lambda y^{U\Rightarrow U}.x)\ P\ Q \end{array}$$

Tipi di dati: il tipo Nat

$$\begin{array}{rcl} \texttt{It}\ P\ Q\ 0 = & = & (\Lambda X.\lambda x^X.\lambda y^{X\Rightarrow X}.x)\ U\ P\ Q \\ & \to_\beta & (\lambda x^U.\lambda y^{U\Rightarrow U}.x)\ P\ Q \\ & \to_\beta & (\lambda y^{U\Rightarrow U}.P)\ Q \end{array}$$

Tipi di dati: il tipo Nat

$$\begin{array}{lcl} \texttt{It}\ P\ Q\ 0 = & = & (\Lambda X.\lambda x^X.\lambda y^{X\Rightarrow X}.x)\ U\ P\ Q \\ & \to_\beta & (\lambda x^U.\lambda y^{U\Rightarrow U}.x)\ P\ Q \\ & \to_\beta & (\lambda y^{U\Rightarrow U}.P)\ Q \\ & \to_\beta & P \end{array}$$

Tipi di dati: il tipo Nat

$$\texttt{It}\ P\ Q\ (\texttt{Succ}\ N) \quad = \quad (\Lambda X.\lambda x.^{X}.\lambda y^{X \Rightarrow X}.y(N\ X\ x\ y))\ U\ P\ Q$$

Tipi di dati: il tipo Nat

$$\begin{array}{lcl} \mathtt{It}\ P\ Q\ (\mathtt{Succ}\ N) & = & (\Lambda X.\lambda x.^{X}.\lambda y^{X\Rightarrow X}.y(N\ X\ x\ y))\ U\ P\ Q \\ & \to_\beta & (\lambda x.^{U}.\lambda y^{U\Rightarrow U}.y(N\ U\ x\ y))\ P\ Q \end{array}$$

Tipi di dati: il tipo Nat

$$\begin{array}{lcl} \texttt{It}\ P\ Q\ (\texttt{Succ}\ N) & = & (\Lambda X.\lambda x.^{X}.\lambda y^{X\Rightarrow X}.y(N\ X\ x\ y))\ U\ P\ Q \\ & \to_\beta & (\lambda x.^{U}.\lambda y^{U\Rightarrow U}.y(N\ U\ x\ y))\ P\ Q \\ & \to_\beta & (\lambda y^{U\Rightarrow U}.y(N\ U\ P\ y))\ Q \end{array}$$

Tipi di dati: il tipo Nat

$$
\begin{array}{lcl}
\texttt{It}\ P\ Q\ (\texttt{Succ}\ N) & = & (\Lambda X.\lambda x.^{X}.\lambda y^{X \Rightarrow X}.y(N\ X\ x\ y))\ U\ P\ Q \\
& \to_\beta & (\lambda x.^{U}.\lambda y^{U \Rightarrow U}.y(N\ U\ x\ y))\ P\ Q \\
& \to_\beta & (\lambda y^{U \Rightarrow U}.y(N\ U\ P\ y))\ Q \\
& \to_\beta & Q(N\ U\ P\ Q)
\end{array}
$$

Tipi di dati: il tipo Nat

$$
\begin{array}{lcl}
\texttt{It}\ P\ Q\ (\texttt{Succ}\ N) & = & (\Lambda X.\lambda x.^{X}.\lambda y^{X\Rightarrow X}.y(N\ X\ x\ y))\ U\ P\ Q \\
& \to_\beta & (\lambda x.^{U}.\lambda y^{U\Rightarrow U}.y(N\ U\ x\ y))\ P\ Q \\
& \to_\beta & (\lambda y^{U\Rightarrow U}.y(N\ U\ P\ y))\ Q \\
& \to_\beta & Q(N\ U\ P\ Q) \\
& = & Q(\texttt{It}\ P\ Q\ \ N)
\end{array}
$$

Tipi di dati: Liste

$$\texttt{List}\ U = \forall X\ X \Rightarrow (U \Rightarrow X \Rightarrow X) \Rightarrow X$$

Tipi di dati: Liste

$$\texttt{List}\ U = \forall X\ X \Rightarrow (U \Rightarrow X \Rightarrow X) \Rightarrow X$$

- Costruttori:

$$\texttt{nil} = \Lambda X.\lambda^{X}.\lambda y^{U \Rightarrow X \Rightarrow X}.x : \texttt{List}\ U$$

Tipi di dati: Liste

$$\texttt{List}\ U = \forall X\ X \Rightarrow (U \Rightarrow X \Rightarrow X) \Rightarrow X$$

- Costruttori:

$$\texttt{nil} = \Lambda X.\lambda^{X}.\lambda y^{U \Rightarrow X \Rightarrow X}.x : \texttt{List}\ U$$

Sia M di tipo U e N di tipo $\texttt{List}\ U$. Allora

$$\texttt{cons}\ M\ N = \Lambda X.\lambda x^{X}.\lambda y^{U \Rightarrow X \Rightarrow X}.y\ M\ (N\ X\ x\ y) : \texttt{List}\ U$$

Tipi di dati: Liste

$$\texttt{List}\ U = \forall X\ X \Rightarrow (U \Rightarrow X \Rightarrow X) \Rightarrow X$$

- Costruttori:

$$\texttt{nil} = \Lambda X.\lambda^{X}.\lambda y^{U \Rightarrow X \Rightarrow X}.x : \texttt{List}\ U$$

Sia M di tipo U e N di tipo $\texttt{List}\ U$. Allora

$$\texttt{cons}\ M\ N = \Lambda X.\lambda x^{X}.\lambda y^{U \Rightarrow X \Rightarrow X}.y\ M\ (N\ X\ x\ y) : \texttt{List}\ U$$

- Distruttori:
Sia W un tipo, P un termine di tipo W, F di tipo $U \Rightarrow W \Rightarrow W$, ed N di tipo $\texttt{List}\ U$:

$$\texttt{It}\ P\ F\ N = N\ W\ P\ F : W$$

Tipi di dati: Liste

$$[u_1, \ldots u_n]$$

Tipi di dati: Liste

$$[u_1, \ldots u_n]$$

$$\Lambda X.\lambda y^{U \Rightarrow X \Rightarrow X}.y\ u_1(y\ u_2 \ldots (y\ u_n\ x)\ldots)$$

Tipi di dati: Liste

$$[u_1, \ldots u_n]$$

$$\Lambda X.\lambda y^{U \Rightarrow X \Rightarrow X}.y\ u_1(y\ u_2 \ldots (y\ u_n\ x)\ldots)$$

$$\texttt{cons}\ u_1(\texttt{cons}\ u_2 \ldots (\texttt{cons}\ u_n\ \texttt{nil})\ldots)$$

Tipi di dati : Liste

$$\texttt{It}\ P\ F\ \texttt{nil} \qquad \rightarrow_\beta \qquad P$$

Tipi di dati : Liste

$$\texttt{It}\ P\ F\ \texttt{nil} \quad \to_\beta \quad P$$

$$\texttt{It}\ P\ F\ \texttt{cons}\ M\ N \quad \to_\beta \quad F\ M\ (\texttt{It}\ P\ F\ N)$$

Normalizzazione forte nel Sistema F

- Modulo l'isomorfismo di Curry-Howard, la normalizzazione forte di F implica la proprietà di eliminazione del taglio per la logica intuzionista del secondo ordine;

Normalizzazione forte nel Sistema F

- Modulo l'isomorfismo di Curry-Howard, la normalizzazione forte di F implica la proprietà di eliminazione del taglio per la logica intuzionista del secondo ordine;
- L'eliminazione del taglio della logica intuizionista del secondo ordine implica la coerenza della logica intuizionista del secondo ordine;

Normalizzazione forte nel Sistema F

- Modulo l'isomorfismo di Curry-Howard, la normalizzazione forte di F implica la proprietà di eliminazione del taglio per la logica intuzionista del secondo ordine;
- L'eliminazione del taglio della logica intuizionista del secondo ordine implica la coerenza della logica intuizionista del secondo ordine;
- Il sistema F (i.e. la logica intuizionista del secondo ordine) è un sistema formale sufficientemente espressivo da poter rappresentare l'aritmetica di Peano;

Normalizzazione forte nel Sistema F

- Modulo l'isomorfismo di Curry-Howard, la normalizzazione forte di F implica la proprietà di eliminazione del taglio per la logica intuzionista del secondo ordine;
- L'eliminazione del taglio della logica intuizionista del secondo ordine implica la coerenza della logica intuizionista del secondo ordine;
- Il sistema F (i.e. la logica intuizionista del secondo ordine) è un sistema formale sufficientemente espressivo da poter rappresentare l'aritmetica di Peano;
- per il secondo teorema di Godel, il Sistema F non può dimostrare la sua propria coerenza con metodi finiti.

Normalizzazione forte nel Sistema F

- Modulo l'isomorfismo di Curry-Howard, la normalizzazione forte di F implica la proprietà di eliminazione del taglio per la logica intuzionista del secondo ordine;
- L'eliminazione del taglio della logica intuizionista del secondo ordine implica la coerenza della logica intuizionista del secondo ordine;
- Il sistema F (i.e. la logica intuizionista del secondo ordine) è un sistema formale sufficientemente espressivo da poter rappresentare l'aritmetica di Peano;
- per il secondo teorema di Godel, il Sistema F non può dimostrare la sua propria coerenza con metodi finiti.

Dimostrare la forte normalizzazione del Sistema F non può essere banale...

Normalizzazione forte per i tipi semplici

Per ogni tipo A definiamo la nozione di termine *riducibile* di tipo A:

Normalizzazione forte per i tipi semplici

Per ogni tipo A definiamo la nozione di termine *riducibile* di tipo A:

- Se A è un tipo atomico, un termine t di tipo A è *riducibile* sse t è fortemente normalizzante;

Normalizzazione forte per i tipi semplici

Per ogni tipo A definiamo la nozione di termine *riducibile* di tipo A:

- Se A è un tipo atomico, un termine t di tipo A è *riducibile* sse t è fortemente normalizzante;
- Un termine t di tipo $A \Rightarrow B$ è *riducibile* sse per tutti i termini reducibili u di tipo A, $t\ u$ è riducibile di tipo B.

Normalizzazione forte per i tipi semplici

Per ogni tipo A definiamo la nozione di termine *riducibile* di tipo A:

- Se A è un tipo atomico, un termine t di tipo A è *riducibile* sse t è fortemente normalizzante;
- Un termine t di tipo $A \Rightarrow B$ è *riducibile* sse per tutti i termini reducibili u di tipo A, $t\ u$ è riducibile di tipo B.

Un termine è *semplice* quando non comincia per λ.

Normalizzazione forte per i tipi semplici

Proprietà della riducibilità

R1 Ogni termine riducibile è fortemente normalizzante;

Normalizzazione forte per i tipi semplici

Proprietà della riducibilità

R1 Ogni termine riducibile è fortemente normalizzante;

R2 Se t è riducibile e t si riduce a t' allora t' è riducibile,

Normalizzazione forte per i tipi semplici

Proprietà della riducibilità

- R1 Ogni termine riducibile è fortemente normalizzante;
- R2 Se t è riducibile e t si riduce a t' allora t' è riducibile,
- R3 Se t è semplice e tutti i ridotti immediati di t sono riducibili allora t è riducibile.

Normalizzazione forte per i tipi semplici

Proprietà della riducibilità

- R1 Ogni termine riducibile è fortemente normalizzante;
- R2 Se t è riducibile e t si riduce a t' allora t' è riducibile,
- R3 Se t è semplice e tutti i ridotti immediati di t sono riducibili allora t è riducibile.

Proposizione 1

Se t è un termine di tipo B e se per tutti i termine riducibili u di tipo A, $t[u/x^A]$ è riducibile, allora $\lambda x^A.t$ è riducibile.

Normalizzazione forte per i tipi semplici

Teorema 1 (W.W. Tait)

Tutti i λ-termini semplicemente tipati sono riducibili

Normalizzazione forte per i tipi semplici

Teorema 1 (W.W. Tait)

Tutti i λ-termini semplicemente tipati sono riducibili

Si dimostra, per induzione sul termine t che se si sostituiscono le variabili libere di t con termini riducibili di tipo opportuno, si ottiene un termine riducibile. Nel caso t sia una variabile è immediato; se t è una λ-astrazione, la proprietà vale per la Proposizione 1, mentre nel caso t sia un'applicazione, per definizione di reducibilità.
Dato che le variabili sono riducibili per $R3$, per sostituzione identica il teorema segue.

Normalizzazione forte per i tipi semplici

Corollario 1

Tutti i λ-termini semplicemente tipati sono fortemente normalizzabili.

Normalizzazione forte per i tipi semplici

Corollario 1

Tutti i λ-termini semplicemente tipati sono fortemente normalizzabili.

Per il Teorema 1 e per $R1$.

Una generalizzazione sbagliata

Proviamo ad estendere la nozione di riducibilità ai tipi universali:

- Un termine t di tipo $\forall XA$ è *riducibile* sse per tutti i tipi B, $t\ B$ è riducibile.

Una generalizzazione sbagliata

Proviamo ad estendere la nozione di riducibilità ai tipi universali:

- Un termine t di tipo $\forall XA$ è *riducibile* sse per tutti i tipi B, $t\ B$ è riducibile.

Sia t di tipo $\forall X(X \Rightarrow X)$; allora $t\ B$ è di tipo $B \Rightarrow B$.

Una generalizzazione sbagliata

Proviamo ad estendere la nozione di riducibilità ai tipi universali:

- Un termine t di tipo $\forall XA$ è *riducibile* sse per tutti i tipi B, $t\ B$ è riducibile.

Sia t di tipo $\forall X(X \Rightarrow X)$; allora $t\ B$ è di tipo $B \Rightarrow B$.

t è riducibile sse per ogni tipo B e per ogni termine riducibile u di tipo B, $(t\ B)\ u$ di tipo B è riducibile.

Una generalizzazione sbagliata

Proviamo ad estendere la nozione di riducibilità ai tipi universali:

- Un termine t di tipo $\forall XA$ è *riducibile* sse per tutti i tipi B, $t\ B$ è riducibile.

Sia t di tipo $\forall X(X \Rightarrow X)$; allora $t\ B$ è di tipo $B \Rightarrow B$.

t è riducibile sse per ogni tipo B e per ogni termine riducibile u di tipo B, $(t\ B)\ u$ di tipo B è riducibile.

Definizione circolare: la riducibilità del singolo universale presuppone la riducibilità di tutti i tipi (dunque anche dell'universale stesso!)

I candidati di riducibilità

Candidati di riducibilità (J.Y. Girard)

Sia A un tipo: un candidato di riducibilità di tipo A è un insieme $\mathcal{C}$ di termini di A, che gode delle seguenti proprie tà:

CR1 se t appartiene a $\mathcal{C}$ è fortemente normalizzante;

I candidati di riducibilità

Candidati di riducibilità (J.Y. Girard)

Sia A un tipo: un candidato di riducibilità di tipo A è un insieme $\mathcal{C}$ di termini di A, che gode delle seguenti proprie tà:

- CR1 se t appartiene a $\mathcal{C}$ è fortemente normalizzante;
- CR2 Se t appartiene a $\mathcal{C}$ e t si riduce a t' allora t' appartiene a $\mathcal{C}$,

I candidati di riducibilità

Candidati di riducibilità (J.Y. Girard)

Sia A un tipo: un candidato di riducibilità di tipo A è un insieme $\mathcal{C}$ di termini di A, che gode delle seguenti proprie tà:

- CR1 se t appartiene a $\mathcal{C}$ è fortemente normalizzante;
- CR2 Se t appartiene a $\mathcal{C}$ e t si riduce a t' allora t' appartiene a $\mathcal{C}$,
- CR3 Se t è semplice e tutti i ridotti immediati di t appartengono a $\mathcal{C}$ allora t appartiene a $\mathcal{C}$.

I candidati di riducibilità

Sia t di tipo $\forall X(X \Rightarrow X)$; allora $t\ B$ è di tipo $B \Rightarrow B$.

I candidati di riducibilità

Sia t di tipo $\forall X(X \Rightarrow X)$; allora $t\ B$ è di tipo $B \Rightarrow B$.

t è riducibile sse per ogni tipo B e per ogni candidato di reducibilità $\mathcal{B}$ di tipo B, se u appartiene a $\mathcal{B}$, allora $(t\ B)\ u$ appartiene a $\mathcal{B}$.

I candidati di riducibilità

Sia t di tipo $\forall X(X \Rightarrow X)$; allora $t\ B$ è di tipo $B \Rightarrow B$.

t è riducibile sse per ogni tipo B e per ogni candidato di reducibilità $\mathcal{B}$ di tipo B, se u appartiene a $\mathcal{B}$, allora $(t\ B)\ u$ appartiene a $\mathcal{B}$.

Definizione non circolare: appartenere ad un candidato di reducibilità non coincide a priori con l'essere riducibile.

I termini del Sistema F sono fortemente normalizzanti

Una volta definita la nozione di reducibilità usando i candidati di riducibilità, la prova segue (a meno di dettagli tecnici) lo stesso pattern della prova di Tait.

I termini del Sistema F sono fortemente normalizzanti

Una volta definita la nozione di reducibilità usando i candidati di riducibilità, la prova segue (a meno di dettagli tecnici) lo stesso pattern della prova di Tait.

Teorema 2

Tutti i termini del sistema F sono riducibili.

I termini del Sistema F sono fortemente normalizzanti

Una volta definita la nozione di reducibilità usando i candidati di riducibilità, la prova segue (a meno di dettagli tecnici) lo stesso pattern della prova di Tait.

Teorema 2

Tutti i termini del sistema F sono riducibili.

Corollario 2

Tutti i termini del sistema F sono fortemente normalizzanti.