

Fondamenti: Alcune Proprietà di Calcolabilità e dei Linguaggi per Funzioni Calcolabili

Sommario: 9 marzo, 2015

- Funzioni Parziali e Esecuzioni Terminanti di Programmi
- Problemi Semidecidibili: Halting, Equivalenza, Ambiguità
- Linguaggi per \mathcal{F} : Lambda Calcolo (Church 1936)
- Logica Combinatoria (Shonfinkel 1924)

Funzioni Parziali e Decidibilità

- **Funzione di Decisione** È una funzione booleana, g , totale, ovvero:
 - $g \in \mathcal{D} \rightarrow \{\text{true}, \text{false}\}$ per due valori $\text{true}, \text{false} \in \mathcal{D}$
 - $\forall x \in \mathcal{D}, g(x) \in \{\text{true}, \text{false}\}$
- **Funzione di semi-Decisione** È una funzione parziale g , ovvero:
 - $g \in \mathcal{D} \rightarrow \{\text{true}\}$ per $\text{true} \in \mathcal{D}$
 - $\forall x \in \mathcal{D}, g(x) = \text{true}$ oppure $g(x) = \uparrow$ ¹
- **Decidibilità** Le funzioni di decisione e semi-decisione, sono utilizzate per descrivere problemi calcolabili con caratteristiche diverse.

¹ \uparrow significa *indefinito*, ovvero ogni programma che esprime la funzione, quando calcolata su tale valore x è non terminante

Decidibilità di alcuni, importanti problemi

- **Ambiguità di un grammatica Libera.** È semi-decidibile se ambigua, ovvero:
 - esiste semi-decisione g , s.t. $g(x)=\text{true}$ sse x è ambigua
- **Equivalenza di due grammatiche Libere.** È semi-decidibile se diverse, ovvero:
 - esiste semi-decisione g , s.t. $g(x,y)=\text{true}$ sse $\mathcal{L}(x) \neq \mathcal{L}(y)$ è ambigua
 - Equivalenza di grammatiche regolari è decidibile
- **Appartenenza a $\mathcal{L}(G)$ per Libera G** È decidibile
 - esiste decisione g_G , s.t. $g_G(x) = \text{true}$ se $x \in \mathcal{L}(G)$; $g_G(x) = \text{false}$ se $x \notin \mathcal{L}(G)$
- **Terminazione Esecuzione Programma p .** È semi-decidibile se termina, ovvero:
 - esiste semi-decisione g , s.t. $g(p,d)=\text{true}$ sse p termina su input d
- **Equivalenza di Programmi.** È semi-decidibile se diversi, ovvero:
 -

Lambda Calcolo

- **Sintassi** (ovviamente, astratta)

$$\Lambda = X \mid \Pi \mid \lambda X . \Lambda \mid \Lambda \Lambda$$

- **Termini:** Λ, g, t, t_i . Insieme (numerabile) dei termini.
 - **Variabili:** X, x, y, z . Insieme (numerabile) di simboli detti variabili.
 - **Costanti:** Π, c, \dots . Insieme (numerabile) di simboli detti costanti (distinguibili: $X \cap \Pi = \{\}$)
 - **op. Astrazione.** $\lambda x . t$ termine ottenuto da t per *astrazione* (generalizzazione) rispetto alla variabile (libera) x . Il termine è anche detto, funzione nella variabile x .
 - **op. Applicazione.** $t_1 t_2$ termine ottenuto per *applicazione* del termine t_1 al termine t_2 . Il termine è anche detto, applicazione di t_1 a t_2 .
- esempi

$\lambda x . \lambda y . x y$

$\lambda x . + x 5$, oppure $\lambda x . +(x)(5)$, o $\lambda x . x + 5$ dipende dalla sintassi concreta

Lambda Calcolo: Convenzioni e Definizioni Sintattiche

$$\Lambda = X \mid \Pi \mid \lambda X. \Lambda \mid \Lambda \Lambda$$

- **Sintassi Concreta e Variabili Libere**

- **op. Applicazione.** è associativa a sinistra:

$x y x$ significa $((x y) x)$

- **op. Applicazione.** ha priorità sull'astrazione:

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

- **Variabile Legata, Libera, Occorrente.** Le variabili che occorrono in un termine t sono raccolte in $\text{Var}(t)$, e si dividono Legate, $\text{BV}(t)$, o Libere, $\text{FV}(t)$, a seconda che siano state astratte o meno.

$$\text{FV}(x) = \{x\}$$

$$\text{FV}(c) = \{\}$$

$$\text{FV}(\lambda x. t) = \text{FV}(t) \setminus \{x\}$$

$$\text{FV}(t_1 t_2) = \text{FV}(t_1) \cup \text{FV}(t_2)$$

$$\text{Var}(x) = \{x\}$$

$$\text{Var}(c) = \{\}$$

$$\text{Var}(\lambda x. t) = \text{Var}(t) \cup \{x\}$$

$$\text{Var}(t_1 t_2) = \text{Var}(t_1) \cup \text{Var}(t_2)$$

$$\text{BV}(t) = \text{Var}(t) \setminus \text{FV}(t).$$

- **esempi**

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

Sia $t \equiv \lambda y. \lambda x. xz$. $\text{Var}(t) = \{x, y, z\}$, $\text{FV}(t) = \{z\}$, $\text{BV}(t) = \{x, y\}$.

Sia $t \equiv \lambda y. \lambda x. (\lambda x. x + 5)(x + y)$

Lambda Calcolo: Semantica

$$\Lambda = X \mid \Pi \mid \lambda X. \Lambda \mid \Lambda \Lambda$$

• Semantica

- Una relazione \rightarrow definita da 3 regole ed estesa in una congruenza su Λ .
- α – **red.**
 $\lambda x. t \rightarrow \lambda y. t[y/x]$ per $y \notin BV(t)$
- β – **red.**
 $(\lambda x. t)t_2 \rightarrow t[t_2/x]$ per $FV(t_2) \cap BV(t) = \{\}$
- η – **red.**
 $\lambda x. (t x) \rightarrow t$ per $x \notin FV(t)$
- Sostituzione. $t_1[t_2/x]$ è l'operazione che rimpiazza, in t_1 , ogni occorrenza libera di x con il termine t_2 .

$$x[t/x] = t$$

$$y[t/x] = y \text{ con } x \neq y; \quad c[t/x] = c$$

$$(\lambda x. t)[t_2/x] = \lambda x. t; \quad (\lambda y. t)[t_2/x] = \lambda y. (t[t_2/x]) \text{ con } x \neq y$$

$$(t_1 t_2)[t_3/x] = (t_1[t_3/x])(t_2[t_3/x])$$

• esempi

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

Lambda Calcolo: Programmare

$$\Lambda = X \mid \Pi \mid \lambda X. \Lambda \mid \Lambda \Lambda$$

- **Programmi** Sono tutti i termini chiusi, i.e.

$$\mathcal{P} = \{t \in \Lambda \mid \text{FV}(t) = \{\}\}$$

- **Aritmetica**

- $[0] \equiv \lambda y. \lambda x. x$
- $[n + 1] \equiv \lambda y. \lambda x. y([n])$
- $\text{succ} \equiv \lambda z. \lambda y. \lambda x. y(zyx)$
- plus, prod, minus, div

- **Conditional e booleani**

- $\text{if} \equiv \lambda b. \lambda x. \lambda y. bxy$
- $\text{true} \equiv \lambda x. \lambda y. x$
- $\text{false} \equiv \lambda x. \lambda y. y$
- and, or, eq, ...

- **Recursion-FixedPoint**

- $\Psi \equiv \lambda g. (\lambda x. g(xx))(\lambda x. g(xx))$

Lambda Calcolo: Programmare

$$\Lambda = X \mid \Pi \mid \lambda X. \Lambda \mid \Lambda \Lambda$$

- **Aritmetica**
- **Programmi**
- **Conditional e costanti booleane**
- **Recursion-FixedPoint**

- $\Psi \equiv \lambda g. (\lambda x. g(xx))(\lambda x. g(xx))$

- $\forall g \in \mathcal{F}, \forall d \in \mathcal{D},$

$$\Psi(g)(d) = g(\Psi(g))(d)$$

- Sia $g = \lambda x. t$ una definizione per una funzione g^2 :

$$g \equiv \Psi(\lambda g. \lambda x. t)$$

- **Esempio**

`fact = $\lambda x. \text{if } (\text{eq } x[0]) [1] (\text{prod } x (\text{fact } (\text{minus } x [1])))$`

è una sintassi concreta per il termine

`$\Psi(\lambda \text{fact}. \lambda x. \text{if } (\text{eq } x[0]) [1] (\text{prod } x (\text{fact } (\text{minus } x [1])))$`

²(possibilmente ricorsiva, i.e. g occorre libera in t)  8/26

Logica Combinatoria

- **Sintassi** (ovviamente, astratta)

$$\mathcal{C} = X \mid \Pi \mid S \mid K \mid I \mid \mathcal{CC}$$

- **Termini:** \mathcal{C}, r, r_i . Insieme (numerabile) dei termini.
 - **Variabili:** X, x, y, z . Insieme (numerabile) di simboli detti variabili libere.
 - **Costanti:** Π, c, \dots . Insieme (numerabile) di simboli detti costanti (distinguibili: $X \cap \Pi = \{\}$)
 - **S, K, I** Sono tre identificatori, i.e. $S, K, I \in \Pi$, detti combinatori fondamentali
 - **op. Applicazione.** $r_1 r_2$ termine ottenuto per *applicazione* del termine r_1 al termine r_2 . Il termine è anche detto, applicazione di r_1 a r_2 .
- **Esempi**
 $S(Kx)I$
 $S(S(KS)(S(KK)I))(KI)$

Logica Combinatoria: Convenzioni e Definizioni Sintattiche

$$\mathcal{C} = X \mid \Pi \mid S \mid K \mid I \mid \mathcal{CC}$$

- **Sintassi Concreta e Variabili Libere**
 - **op. Applicazione.** è associativa a sinistra:
 $x y x$ significa $((x y) x)$
 - **Variabile Libera.** Le variabili che occorrono in un termine sono tutte libere.
- esempi
 $S(Kx)I$
 $S(S(K+)I)(K5)$

Logica Combinatoria: Semantica

$$\mathcal{C} = X \mid \Pi \mid S \mid K \mid I \mid \mathcal{C}\mathcal{C}$$

- **Semantica**

- Una relazione \rightarrow definita da 1 regola per combinatore ed estesa in una congruenza su \mathcal{C} ..

- $S\ r_1\ r_2\ r \rightarrow r_1\ r\ (r_2\ r)$
- $K\ r_1\ r_2 \rightarrow r_1$
- $I\ r \rightarrow r$

- **esempi**

$S(Kx)I\ 3 \rightarrow \dots$

$S(S(K+)I)(K\ 5)7 \rightarrow \dots$

$S(S(KS)(S(KK)I))(KI)3\ 5 \rightarrow \dots$

Logica Combinatoria: Astrazione/1

$$\mathcal{C} = \mathbf{X} \mid \Pi \mid \mathbf{S} \mid \mathbf{K} \mid \mathbf{I} \mid \mathcal{CC}$$

- Sia \mathcal{U} un generico calcolo con termini F per esprimere funzioni e valori, variabili X per e. parametri ³, applicazione di termini per e. l'applicazione di funzione.⁴

Definition (Astratto, Proprietà di Astrazione)

L'astratto di un termine $u \in F$, rispetto ad una variabile $x \in X$, è un termine che indichiamo con $\{x\}u$ avente la seguente proprietà di astrazione:

$$\forall w \in F, (\{x\}u)_w = u[w/x]$$

dove $(\{x\}u)_w$ è l'applicazione di $\{x\}u$ a w , $u[w/x]$ è la sostituzione, in u , di x con w .

- Nel Lambda-Calcolo, $\{x\}u$ è il termine $\lambda x.u$ e fa parte di una classe speciale di termini.
- \mathcal{C} non ha una classe speciale di termini.

Proposition (Astratto in \mathcal{C})

\mathcal{C} esprime l'astratto mediante una composizione di applicazioni dei suoi termini, ovvero: $\forall r \in \mathcal{C}, \forall x \in X, \{x\}r \in \mathcal{C}$

³ si assuma per semplicità, che variabili diverse abbiano sempre nomi diversi in uno stesso termine

⁴ il calcolo può contenere ulteriori operazioni e strutture

Logica Combinatoria: Astrazione/2

$$\mathcal{C} = \mathbf{X} \mid \Pi \mid \mathbf{S} \mid \mathbf{K} \mid \mathbf{I} \mid \mathcal{C}\mathcal{C}$$

Proposition (Astratto in \mathcal{C})

\mathcal{C} esprime l'astratto mediante una composizione di applicazioni dei suoi termini, ovvero: $\forall r \in \mathcal{C}, \forall x \in \mathbf{X}, \{x\}r \in \mathcal{C}$

- **Astrazione.** Dato un qualunque termine r e una qualunque variabile x , mostriamo un metodo ⁵ per ottenere $\{x\}r \in \mathcal{C}$.
 - $\{x\}x = \mathbf{I}$
 - $\{x\}r = \mathbf{K} r$ se $x \notin \text{Var}(r)$
 - $\{x\}(r_1 r_2) = \mathbf{S}(\{x\}r_1)(\{x\}r_2)$
- Esempio
 $\{x\}(\{y\}x) = \dots$

⁵ Si noti che $\{x\}r$ può essere espresso con termini equivalenti ma diversi, pertanto esistono altri metodi per calcolare per l'astratto in \mathcal{C}

Logica Combinatoria: Programmare con liste

$$\mathcal{C} = X \mid \Pi \mid S \mid K \mid I \mid \mathcal{C}\mathcal{C}$$

- **Programmi** Sono tutti i termini chiusi, i.e. senza variabili

$$\mathcal{P} = \{r \in \mathcal{C} \mid \text{Var}(r) = \{\}\}$$

- **Conditional e booleani**

- $\text{if} \equiv \{b\}(\{x\}(\{y\}bxy))$
- $\text{true} \equiv S(KK)I$
- $\text{false} \equiv KI$
- and, or, eq, ...

- **Coppie e liste**

- $\text{pair} \equiv \{x\}(\{y\}(\{s\}sxy))$
- $\text{first } r \equiv r \text{ true}$
- $\text{snd } r \equiv r \text{ false}$

Logica Combinatoria: Programmare con aritmetica

$$\mathcal{C} = \text{X} \mid \Pi \mid \text{S} \mid \text{K} \mid \text{I} \mid \mathcal{CC}$$

- **Programmi** Sono tutti i termini chiusi, i.e. senza variabili

$$\mathcal{P} = \{r \in \mathcal{C} \mid \text{Var}(r) = \{\}\}$$

- **Conditional e booleani**
- **Coppie e liste**
- **Aritmetica**
 - $[0] \equiv \text{pair true false}$
 - $[n + 1] \equiv \text{pair false } [n]$
 - $\text{zerop } r \equiv r \text{ true}$
 - $\text{succ, plus, prod, minus, div}$
- **Recursion-FixedPoint**
 - $\Psi \equiv \dots$

Logica Combinatoria: Conclusioni (per ora)

$$\mathcal{C} = \mathbf{X} \mid \Pi \mid \mathbf{S} \mid \mathbf{K} \mid \mathbf{I} \mid \mathcal{CC}$$

- **Programmi** Sono tutti i termini chiusi, i.e. senza variabili
- **Aritmetica e Liste**
- **Conditional e booleani**
- **Recursion-FixedPoint**
- **Un monoide** con applicazione come unica operazione
- **Il più compatto** formalismo di calcolo
- **Il più compatto** linguaggio di programmazione
 - esecutore semplice da realizzare (non richiede sostituzione)
 - valori e operazioni espressi nel linguaggio (inclusa ricorsione) possono essere realizzati come primitive della sua Macchina Astratta
- **Espressività: Rendere Primitivi i meccanismi di uso comune** (valori, ricorsione,...)

Esercizi.

Esercizio

Si dia una semantica SOS per il Lambda Calcolo

Soluzione

Esercizi.

Esercizio

Si completino i calcoli indicati con '...' delle slides.

Soluzione

Esercizi.

Esercizio

Si mostri la sequenza di α – red applicate per ridurre il termine dato ad un termine contenente sempre, identificatori diversi per variabili diverse: $\lambda x.\lambda y.\lambda x.y(\lambda y.x)x(\lambda y.x(\lambda x.yx)y)$

Soluzione

Esercizi.

Esercizio

Si dimostri che: $\forall F, \Psi F = F(\Psi F)$

Soluzione

Esercizi.

Esercizio

Si scriva, in Lambda-Calcolo, un programma per la funzione prod introdotta nell'aritmetica data per tale linguaggio.

Soluzione

Esercizi.

Esercizio

Si scriva, in Lambda-Calcolo, un programma per la funzione `zerop` che calcola il test su zero nell'aritmetica data per tale linguaggio.

Soluzione

Esercizi.

Esercizio

Si dia una semantica SOS per la Logica Combinatoria

Soluzione

Esercizi.

Esercizio

Si dimostri che $SKK = I$

Soluzione

Esercizi.

Esercizio

Si scriva, in Logica Combinatoria, un programma per la funzione plus introdotta nell'aritmetica data per tale linguaggio.

Soluzione

Esercizi.

Esercizio

Si scriva, in Logica Combinatoria, un programma per la funzione minus introdotta nell'aritmetica data per tale linguaggio.

Soluzione