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

Sommario: 14-15 marzo, 2018

- Funzioni Parziali ed Esecuzioni Terminanti di Programmi
- Problemi Semidecidibili: Halting, Equivalenza, Ambiguità
- Linguaggi per $\mathcal{F}$: Lambda Calcolo (Church 1936)
- Logica Combinatoria (Shonfinkel 1924)
- Macchina a Stati (Minsky 1956, Wang 1957)
- Calcolo di Processi Mobili: π-Calcolo (Milner et al. 1992)

Letture e Approfondimenti:

- [Barendregt90] H.P. Barendregt, Functional Programming and Lambda Calculus, in Handbook of Theoretical Computer Science, vol. B, Chapter 7, pp. 321-363, Elsevier Science Publishers, 1990
- [Minski72] M. Minsky, Computation: Finite and Infinite Machines, Chapter 11, pp. 199-216, Prentice-Hall International, 1972.
- [Milner92] R. Milner, J. Parrow, D. Walker, A Calculus of Mobile Processes, Information and Computations 100, pp. 1-9, 41-49, 1992.

Funzioni Parziali e Decidibilità

- **Funzione di Decisione** È una funzione booleana, g, totale, ovvero:
  - $\mathtt{g} \in \mathcal{D} \to \{\mathtt{true}, \mathtt{false}\}$ per due valori $\mathtt{true}, \mathtt{false} \in \mathcal{D}$
  - $\forall \mathtt{x} \in \mathcal{D}, \mathtt{g}(\mathtt{x}) \in \{\mathtt{true}, \mathtt{false}\}$

- **Funzione di semi-Decisione** È una funzione parziale g, ovvero:
  - $\mathtt{g} \in \mathcal{D} \to \{\mathtt{true}\}$ per $\mathtt{true} \in \mathcal{D}$
  - $\forall \mathtt{x} \in \mathcal{D}, \mathtt{g}(\mathtt{x}) = \mathtt{true}$ oppure $\mathtt{g}(\mathtt{x}) = \uparrow$ [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 $\mathtt{x}$ 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)=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)=true` sse $\mathcal{L}(\mathtt{x}) \neq \mathcal{L}(\mathtt{y})$
  - Equivalenza di grammatiche regolari è decidibile
- **Appartenenza a $\mathcal{L}(\mathtt{G})$ per Libera G** È decidibile
  - esiste decisione $\mathtt{g}_{\mathtt{G}}$, s.t. $\mathtt{g}_{\mathtt{G}}(\mathtt{x}) = \begin{cases} \mathtt{true} & \mathtt{x} \in \mathcal{L}(\mathtt{G}) \\ \mathtt{false} & \mathtt{x} \notin \mathcal{L}(\mathtt{G}) \end{cases}$
- **Terminazione Esecuzione Programma `p`.** È semi-decidibile se termina, ovvero:
  - esiste semi-decisione g, s.t. `g(p,d)=true` sse `p` termina su input `d`
- **Equivalenza di Programmi.** È non decidibile
  - eccetto che per programmi terminanti (vedi esercizio ...)

Calcolo di Funzioni: Lambda Calcolo (Church 1936)

- **Sintassi** (ovviamente, astratta)

$$\Lambda = \mathtt{X} \mid \Pi \mid \lambda\, \mathtt{X}\, .\, \Lambda \mid \Lambda\Lambda$$

  - **Termini:** $\Lambda, \mathtt{g}, \mathtt{t}, \mathtt{t_i}$. Insieme (numerabile) dei termini.
  - **Variabili:** $\mathtt{X}, \mathtt{x}, \mathtt{y}, \mathtt{z}$. Insieme (numerabile) di simboli detti variabili.
  - **Costanti:** $\Pi, \mathtt{c}...$. Insieme (numerabile) di simboli detti costanti (distinguibili: $\mathtt{X} \cap \Pi = \{\}$)
  - **Op. Astrazione:** $\lambda\, \mathtt{x}\, .\mathtt{t}$. Operazione binaria che esprime il valore (funzionale) ottenuto da $\mathtt{t}$ per *astrazione* (generalizzazione) rispetto alla variabile (libera) $\mathtt{x}$. Il termine è anche detto, funzione di $\mathtt{t}$ nella variabile $\mathtt{x}$.
  - **Op. Applicazione:** $\mathtt{t_1 t_2}$. Operazione binaria che esprime il valore ottenuto per *applicazione* del termine $\mathtt{t_1}$ al termine $\mathtt{t_2}$. Il termine è anche detto, applicazione di $\mathtt{t_1}$ a $\mathtt{t_2}$. Ha notazione implicita via giustapposizione degli argomenti.

- Esempi

  $[@ - ([\mathtt{x}], [\mathtt{y}])]$

  $[\lambda - ([\mathtt{x}], [\lambda - ([\mathtt{y}], [@ - ([\mathtt{x}], [\mathtt{y}])])])]$ (grafica migliora la lettura!)

λ-Calcolo: Sintassi Concreta - Convenzioni e Definizioni

$$\Lambda = \mathtt{X} \mid \Pi \mid \lambda\,\mathtt{X}\,.\,\Lambda \mid \Lambda\Lambda$$

- **Sintassi Concreta e Variabili Libere**
  - **Applicazione.** è associativa a sinistra:
    $\mathtt{x\,y\,x}$ significa $\mathtt{((x\,y)x)}$
  - **Applicazione.** ha priorità sull'Astrazione:
    $\lambda\mathtt{y.x\,y}$ significa $\lambda\mathtt{y.(x\,y)}$
  - **Variabile Legata, Libera, Occorrenze.** Le variabili che occorrono in un termine $\mathtt{t}$ sono raccolte in $\mathtt{Var(t)}$, e si dividono Legate, $\mathtt{BV(t)}$, o Libere, $\mathtt{FV(t)}$, a seconda che siano state astratte o meno.

$\mathtt{t} \in \Lambda$	$\mathtt{BV(t)}$	$\mathtt{FV(t)}$
$\mathtt{x} \in \mathtt{X}$	$\{\}$	$\{\mathtt{x}\}$
$\mathtt{c} \in \Pi$	$\{\}$	$\{\}$
$\lambda\,\mathtt{x.t_1}$	$\{\mathtt{x}\} \cup \mathtt{BV(t_1)}$	$\mathtt{FV(t_1)} \setminus \{\mathtt{x}\}$
$\mathtt{t_1 t_2}$	$\mathtt{BV(t_1)} \cup \mathtt{BV(t_2)}$	$\mathtt{FV(t_1)} \cup \mathtt{FV(t_2)}$
$\mathtt{Var(t)} = \mathtt{BV(t)} \cup \mathtt{FV(t)}$		

- Esempi

  Sia $\mathtt{t} \equiv \lambda\mathtt{y}.\lambda\mathtt{x}.\mathtt{y}(\mathtt{y}\,\mathtt{x})$. Possiamo omettere le parentesi?

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

  Sia $\mathtt{t} \equiv \lambda\mathtt{y}.\lambda\mathtt{x}.(\lambda\mathtt{x}.\mathtt{x}+5)(\mathtt{x}+\mathtt{y})$. Abbiamo: ...

Lambda Calcolo: Semantica

$$\Lambda = \mathtt{X} \mid \Pi \mid \lambda\,\mathtt{X}\,.\,\Lambda \mid \Lambda\Lambda$$

- **Semantica**
  - Una relazione $\rightarrow$ definita da 3 regole ed estesa in una congruenza su Λ.
  - $\alpha -$ **red**.
    $$\lambda\,\mathtt{x}\,.\,\mathtt{t} \rightarrow_\alpha \lambda\,\mathtt{y}\,.\,\mathtt{t}[\mathtt{y}/\mathtt{x}] \quad \text{per } \mathtt{y} \notin \mathtt{Var(t)}$$
  - $\beta -$ **red**.
    $$(\lambda\,\mathtt{x}\,.\,\mathtt{t})\mathtt{t}_2 \rightarrow_\beta \mathtt{t}[\mathtt{t}_2/\mathtt{x}] \quad \text{per } \mathtt{FV}(\mathtt{t}_2) \cap \mathtt{BV}(\mathtt{t}) = \{\}$$
  - $\eta -$ **red**.
    $$\lambda\,\mathtt{x}\,.(\mathtt{t}\ \mathtt{x}) \rightarrow_\eta \mathtt{t} \quad \text{per } \mathtt{x} \notin \mathtt{FV}(\mathtt{t})$$
  - Sostituzione. $\mathtt{t}_1[\mathtt{t}_2/\mathtt{x}]$ è l'operazione che rimpiazza, in $\mathtt{t}_1$, ogni occorrenza libera di $\mathtt{x}$ con il termine $\mathtt{t}_2$.
    $\mathtt{x}[\mathtt{t}/\mathtt{x}] = \mathtt{t}$
    $\mathtt{y}[\mathtt{t}/\mathtt{x}] = \mathtt{y}$ con $\mathtt{x} \neq \mathtt{y}$; $\quad \mathtt{c}[\mathtt{t}/\mathtt{x}] = \mathtt{c}$
    $(\lambda\,\mathtt{x}\,.\,\mathtt{t})[\mathtt{t}_2/\mathtt{x}] = \lambda\,\mathtt{x}\,.\,\mathtt{t}$; $\quad (\lambda\,\mathtt{y}\,.\,\mathtt{t})[\mathtt{t}_2/\mathtt{x}] = \lambda\,\mathtt{y}\,.\,(\mathtt{t}[\mathtt{t}_2/\mathtt{x}])$ con $\mathtt{x} \neq \mathtt{y}$
    $(\mathtt{t}_1\mathtt{t}_2)[\mathtt{t}_3/\mathtt{x}] = (\mathtt{t}_1[\mathtt{t}_3/\mathtt{x}])(\mathtt{t}_2[\mathtt{t}_3/\mathtt{x}])$
- Esempi
  Sia $\mathtt{t} \equiv \lambda\mathtt{y}.\lambda\mathtt{x}.\mathtt{y}(\mathtt{y}\,\mathtt{x}))$. Quali regole sono applicabili a t?

Lambda Calcolo: Valutazione, Computazione

$$\Lambda = \mathtt{X} \mid \Pi \mid \lambda\, \mathtt{X}\, .\, \Lambda \mid \Lambda\Lambda$$

- **Valutazione, Computazione** di $\mathtt{t} \in \Lambda$
  - Una sequenza:

    $$\mathtt{t}_0 \to_{\mathtt{r}_1} \mathtt{t}_1 \quad \ldots \quad \to_{\mathtt{r}_n} \mathtt{t}_n$$

    tale che:
    - $(\mathtt{t} \equiv \mathtt{t}_0) \ \wedge \ (\mathtt{n} \geq 0)$
    - annotazioni $\mathtt{r}_i \in \{\alpha, \beta, \eta\}$
    - $\mathtt{t}_n \to_{\mathtt{r}_{n+1}} \mathtt{t}_{n+1}$ solo se $\mathtt{r}_{n+1} \equiv \alpha$

- Esempi

  Sia $\mathtt{t} \equiv \lambda\mathtt{x}.(\lambda\mathtt{y}.\mathtt{x}\,\mathtt{y})(\lambda\mathtt{x}.\mathtt{y}\,\mathtt{x})$. Vediamo una valutazione di $\mathtt{t}$:

  $$\mathtt{t} \to_\eta \lambda\mathtt{x}.\mathtt{x}(\lambda\mathtt{x}.\mathtt{y}\,\mathtt{x}) \to_\eta \lambda\mathtt{x}.\mathtt{x}\mathtt{y}$$

  Non è unica:

  $$\mathtt{t} \to_\beta \lambda\mathtt{x}.\mathtt{x}(\lambda\mathtt{x}.\mathtt{y}\,\mathtt{x}) \to_\eta \lambda\mathtt{x}.\mathtt{x}\mathtt{y}$$

Lambda Calcolo: Programmare

$$\Lambda = \mathtt{X} \mid \Pi \mid \lambda\,\mathtt{X}\,.\,\Lambda \mid \Lambda\Lambda$$

- **Programmi** Sono tutti i termini chiusi, i.e.
$$\mathcal{P} = \{\mathtt{t} \in \Lambda \mid \mathtt{FV(t)} = \{\}\}$$
- **Aritmetica**
  - $[0] \equiv \lambda\,\mathtt{y}.\lambda\,\mathtt{x}.\mathtt{x}$
  - $[\mathtt{n}+1] \equiv \lambda\,\mathtt{y}.\lambda\,\mathtt{x}.\mathtt{y}([\mathtt{n}]\mathtt{yx})$
  - $\mathtt{succ} \equiv \lambda\,\mathtt{z}.\lambda\,\mathtt{y}.\lambda\,\mathtt{x}.\mathtt{y}(\mathtt{zyx})$
  - $\mathtt{plus}, \mathtt{prod}, \mathtt{minus}, \mathtt{div}, \ldots$
- **Conditional e booleani**
  - $\mathtt{if} \equiv \lambda\,\mathtt{b}.\lambda\,\mathtt{x}.\lambda\,\mathtt{y}.\,\mathtt{b}\,\mathtt{x}\,\mathtt{y}$
  - $\mathtt{true} \equiv \lambda\,\mathtt{x}.\lambda\,\mathtt{y}.\mathtt{x}$
  - $\mathtt{false} \equiv \lambda\,\mathtt{x}.\lambda\,\mathtt{y}.\mathtt{y}$
  - $\mathtt{zerop}, \mathtt{and}, \mathtt{or}, \mathtt{eq}, \ldots$
- **Recursion-FixedPoint**
  - $\Psi \equiv \lambda\,\mathtt{g}.(\lambda\,\mathtt{x}.\mathtt{g}(\mathtt{xx}))(\lambda\,\mathtt{x}.\mathtt{g}(\mathtt{xx}))$

Lambda Calcolo: Programmare /2

$$\Lambda = \mathtt{X} \mid \Pi \mid \lambda\,\mathtt{X}\,.\,\Lambda \mid \Lambda\Lambda$$

- **Aritmetica**
- **Programmi**
- **Conditional e costanti booleane**
- **Recursion-FixedPoint**
  - $\Psi \equiv \lambda\,\mathtt{g}.(\lambda\,\mathtt{x}.\mathtt{g}(\mathtt{xx}))(\lambda\,\mathtt{x}.\mathtt{g}(\mathtt{xx}))$
  - $\forall \mathtt{h} \in \mathcal{F}, \forall \mathtt{d} \in \mathcal{D},$
    $$\Psi(\mathtt{h})(\mathtt{d}) = \mathtt{h}(\Psi(\mathtt{h}))(\mathtt{d}) \qquad \text{(vedi esercizio L5.7)}$$
  - Sia $\mathtt{g} = \lambda\,\mathtt{x}.\mathtt{t}$ una definizione per una funzione g [2]:
    $$\mathtt{g} \equiv \Psi(\lambda\,\mathtt{g}.\lambda\,\mathtt{x}.\mathtt{t})$$
- **Esempi**

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

  è una sintassi concreta per il termine

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

[2] (possibilmente ricorsiva, i.e. g occorre libera in t)

Lambda Calcolo: Esercizio

- **Recursion-FixedPoint**
  - $\Psi \equiv \lambda\, \mathtt{g}.(\lambda\, \mathtt{x}.\mathtt{g}(\mathtt{xx}))(\lambda\, \mathtt{x}.\mathtt{g}(\mathtt{xx}))$
  - $\forall \mathtt{h} \in \mathcal{F}, \forall \mathtt{d} \in \mathcal{D}$,
    $$\Psi(\mathtt{h})(\mathtt{d}) = \mathtt{h}(\Psi(\mathtt{h}))(\mathtt{d}) \qquad \text{(vedi esercizio L5.7)}$$
- **Esempi**

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

  è una sintassi concreta per il termine

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

  Calcoliamo: $\mathtt{fact}[0]$ ovvero: $\Psi(\mathtt{h})[0]$, dove $\mathtt{h} \equiv \lambda\, \mathtt{fact}.\lambda\, \mathtt{x}.\mathtt{if}..$

  $\Psi(\mathtt{h})[0] = \mathtt{h}(\Psi(\mathtt{h}))[0] \to_\beta (\lambda\, \mathtt{x}.\mathtt{if}...)[0] \to_\beta$

  $\to_\beta \mathtt{if}\ (\mathtt{eq}\ [0]\ [0])\ [1]\ (\mathtt{prod}\ \mathtt{x}\ ((\Psi(\mathtt{h}))\ (\mathtt{minus}\ \mathtt{x}\ [1])))$

  $\to_\beta ... \to_\beta [1]$ (vedi. esercizio L5.10)

Esercizi

1. (a) Si fornisca una grammatica per la sintassi astratta del $\lambda-$Calcolo in accordo alla notazione utilizzata nell'albero astratto $[\lambda - ([\mathtt{x}], [\lambda - ([\mathtt{y}], [@ - ([\mathtt{x}], [\mathtt{y}])])])]$. (b) Si mostri poi, che tale albero è ottenuto dalla grammatica data.
2. (a) Si fornisca una grammatica per la sintassi concreta del $\lambda-$Calcolo in accordo alle proprietà date per associatività e precedenza tra operatori. (b) Si mostri poi, la sintassi concreta dell'albero astratto $[\lambda - ([\mathtt{x}], [\lambda - ([\mathtt{y}], [@ - ([\mathtt{x}], [\mathtt{y}])])])]$. (c) Si mostri, infine il parse tree del termine ottenuto al punto (b) precedente.
3. Si completino i calcoli indicati con '...' nelle slides precedenti.
4. Si mostri la sequenza di $\alpha-\mathtt{red}$ applicate per ridurre il termine dato ad un termine contenente sempre, identificatori diversi per variabili diverse: $\lambda\,\mathtt{x}.\lambda\,\mathtt{y}.(\lambda\,\mathtt{x}.\mathtt{y}(\lambda\,\mathtt{y}.\mathtt{x})\mathtt{x})(\lambda\,\mathtt{y}.\mathtt{x}(\lambda\,\mathtt{x}.\mathtt{yx})\mathtt{y})$
5. (a) Si mostri la valutazione di $\lambda\,\mathtt{x}.\lambda\,\mathtt{y}.(\lambda\,\mathtt{x}.\mathtt{y}(\lambda\,\mathtt{y}.\mathtt{x})\mathtt{x})(\lambda\,\mathtt{y}.\mathtt{x}(\lambda\,\mathtt{x}.\mathtt{yx})\mathtt{y})$. (b) Nell'ipotesi di aver usato $\alpha-\mathtt{red}$ nella valutazione fornita se ne giustifico l'uso.
6. Si mostri la valutazione di $\lambda\,\mathtt{x}.(\lambda\,\mathtt{y}.\lambda\,\mathtt{x}.\mathtt{x}\,\mathtt{y})(\lambda\,\mathtt{y}.\mathtt{y}\,\mathtt{x}))\mathtt{y}$
7. Si dimostri che: $\forall \mathtt{F},\quad \Psi\,\mathtt{F} = \mathtt{F}(\Psi\,\mathtt{F})$
8. (a) Si scriva, in Lambda-Calcolo, un programma per la funzione `plus` introdotta nell'aritmetica data per tale linguaggio. (b) Si mostri la computazione di `plus`[2][1]
9. Si scriva, in Lambda-Calcolo, un programma per la funzione `zerop` che calcola `true` quando applicata a `[0]`, `false` altrimenti. (b) Si mostri la computazione di `zero`[2].

Altri esercizi in: EserciziL5