6 Strutture applicative e Logica Combinatoria

Nel seguito considereremo espressioni che possono essere indefinite, quindi introduciamo le seguenti convenzioni: $e\downarrow$ significa che l'espressione e e' definita, $e_1 = e_2$ significa che entrambe le espressioni sono definite ed uguali, $e_1 \simeq e_2$ (detta uguaglianza di Kleene) significa che o entrambe le espressioni sono indefinite o sono entrambe definite ed uguali.

Definition 6.1 *Una* **struttura applicativa** *(parziale) e' una coppia $(D, \cdot)$, dove $\cdot$ e' una funziona binaria (parziale) su D. Si dice che $e \in D$* **rappresenta** *la funzione parziale $f: D \rightharpoonup D$ sse $\forall d \in D.f(d) \simeq e \cdot d$.*

L'operazione $x \cdot y$ e' detta applicazione, spesso si omette $\cdot$ e si scrive semplicemente $x\ y$. Per evitare un uso eccessivo delle parentesi si conviene di associare l'applicazione a sinistra, quindi $x_1\ x_2 \ldots x_n$ sta per $(\ (x_1\ x_2) \ldots)$.
L'insieme D e' come un universo non tipato i cui elementi possono essere visti sia come dati che come programmi. Tale ambiguita' e' tipica di molti modelli di calcolo: nella architettura von Neumann il contenuto della memoria puo' essere interpretato sia come un dato che come una istruzione, le macchine di Turing possono essere codificate con stringhe e quindi trattate come dati di input per altre macchine di Turing. Sebbene le strutture applicative parziali siano la classe piu' naturale da un punto di vista computazionale, a livello assiomatico considereremo quasi esclusivamente strutture applicative *totali*, poiche' esse sono di piu' immediata formalizzazione nella logica del primo ordine.

Definition 6.2 *Una* **algebra combinatoria***, o piu' brevemente CA, $(D, \cdot, K, S)$ e' una struttura applicativa in cui $K, S \in D$ soddisfano gli assiomi* (K) $K\ x\ y = x$ (S) $S\ x\ y\ z = x\ z\ (y\ z)$
Una **algebra combinatoria parziale***, o piu' brevemente pCA, $(D, \cdot, K, S)$ e' una struttura applicativa parziale in cui $K, S \in D$ soddisfano gli assiomi* (K) $K\ x\ y = x$ (S.1) $S\ x\ y\downarrow$ (S.2) $S\ x\ y\ z \simeq x\ z\ (y\ z)$

Example 6.3 Un esempio di pCA e' la struttura applicativa di Kleene $(N, \cdot)$ in cui $m \cdot n \simeq U(m, n) \simeq \phi_m(n)$, dove U e' la funzione universale e $\phi_m: N \rightharpoonup N$ e' la funzione ricorsiva parziale rappresentata da m (ovvero la funzione sui naturali calcolata dalla macchina di Turing codificata da m). Per dimostrare che esistono elementi K e S che soddisfano gli assiomi per una pCA si deve sfruttare il teorema s-m-n, che dice che per ogni $m, n \in N$ esiste una funzione ricorsiva totale $s_{m,n}: N^{m+1} \to N$ t.c. $\forall e: N, \overline{x}: N^m, \overline{y}: N^n.\phi_e^{m+n}(\overline{x}, \overline{y}) \simeq \phi^n_{s_{m,n}(e,\overline{x})}(\overline{y})$, dove $\phi_e^m: N^m \rightharpoonup N$ e' la funzione ricorsiva parziale ad m argomenti rappresentata da e.

6.1 Logica Combinatoria

La Logica Combinatoria (CL) e' la teoria equazionale (o se si vuole del primo ordine) delle algebre combinatorie. Il linguaggio di CL e' dato dalla segnatura Σ_{CL} con due simboli di costante, K e S, una simbolo di operazione binaria, $\cdot$ ma spesso lasciato implicito. Tuttavia, e' spesso conveniente estendere il linguaggio con altri simboli di costante. Gli assiomi di CL sono gli assimi equazionali (K) e (S), che nella logica del primo ordine andrebbero scritti piu' propriamente (K) $\forall x, y.K\ x\ y = x$ e (S) $\forall x, y, z.S\ x\ y\ z = x\ z\ (y\ z)$
In CL (eventualmente estesa con altre costanti) possiamo definire una operazione sui termini di CL detta *astrazione*. Tale operazione non fa' parte del linguaggio di CL, bensi' e' definita al meta-livello, come l'operazione di sostituzione $[N/x]M$ o la funzione che restituisce l'insieme delle variabili libere $\mathrm{FV}(M)$ in un termine.

Definition 6.4 *Fissato una variabile x l'***operazione di astrazione** *$\lambda^*x.M$ e' definita per induzione sulla struttura del termine M di CL (anche estesa con costanti)*

- $(\lambda^*x.x) \stackrel{\Delta}{\equiv} I \stackrel{\Delta}{\equiv} SKK$
- $(\lambda^*x.M) \stackrel{\Delta}{\equiv} K\ M$, *se* $x \notin \mathrm{FV}(M)$
- $(\lambda^*x.M_1\ M_2) \stackrel{\Delta}{\equiv} S\ (\lambda^*x.M_1)\ (\lambda^*x.M_2)$, *altrimenti.*

Un'altra operazione di astrazione $\lambda^*_p x.M$, e che al contrario di $\lambda^*x.M$ funziona bene anche per pCA, e'

- $(\lambda^*x.x) \stackrel{\Delta}{\equiv} I \stackrel{\Delta}{\equiv} SKK$
- $(\lambda^*x.M_1\ M_2) \stackrel{\Delta}{\equiv} S\ (\lambda^*x.M_1)\ (\lambda^*x.M_2)$
- $(\lambda^*x.M) \stackrel{\Delta}{\equiv} K\ M$, *altrimenti.*

E' facile dimostrare le seguenti proprieta' dell'astrazione, per induzione sulla struttura dei termini.

Proposition 6.5 *L'operazione di astrazione $\lambda^* x.M$ soddisfa le seguenti proprieta' sintattiche*

- $\mathrm{FV}(\lambda^* x.M) = \mathrm{FV}(M) - \{x\}$, *cioe' la variabile su cui si astrae non e' piu' libera;*
- $(\lambda^* x.M) \equiv (\lambda^* y.[y/x]M)$ *se* $y \notin \mathrm{FV}(M)$, *cioe' la ridenominazione della variabile su cui si astrae e' irrilevante;*
- $[N/y](\lambda^* x.M) \equiv (\lambda^* x.[N/y]M)$ *se* $y \not\equiv x$ *e* $x \notin \mathrm{FV}(N)$, *cioe' astrazione e sostituzione commutato;*
- $CL \vdash (\lambda^* x.M)\ x = M$.

Anche l'astrazione $(\lambda^*_p x.M)$ soddisfa simili proprieta', ad eccezione della commutativita' con la sostituzione, addirittura in CL non si riesce nemmeno a derivare $[N/y](\lambda^*_p x.M) = (\lambda^*_p x.[N/y]M)$. La principale differenza tra le due forme di astrazione e' che nella teoria per le pCA si deriva $(\lambda^*_p x.M)\downarrow$, ma in generale non vale $(\lambda^* x.M)\downarrow$.
Usando l'operazione di astrazione e le sue proprieta' e' facile dimostrare che per ogni termine M con $\mathrm{FV}(M) \subseteq \overline{x}$ esiste un termine chiuso N t.c. $CL \vdash N\ \overline{x} = M$, p.e. $N \equiv \lambda^* \overline{x}.M$. Si osservi che fissato M in generale esistono piu' N che soddisfano $N\ \overline{x} = M$, ma che non sono dimostrabilmente equivalenti. Infatti, in molti casi non e' vero che $CL \vdash (\lambda^* x.M) = (\lambda^*_p x.M)$, p.e. quando $M \equiv y\ z$.

6.2 Nozioni di riduzione e proprieta' sintattiche

Poiche' la logica combinatoria (CL) e' una teoria equazionale, ad essa si possono applicare le tecniche di riscrittura, ovvero si puo' descrivere la relazione di uguaglianza $M = N$ dimostrabile in termini di una nozione piu' semplice $M \to_R N$, che intuitivamente dice che "M si semplifica in N (in un passo)". Ricordiamo i concetti generali [gia' introdotti per le teorie euqazionali] nel contesto di CL.

Definition 6.6 *Sia T l'insieme dei termini di CL (eventualmente con costanti aggiuntive). Una relazione binaria R su T e' detta* **nozione di riduzione**. *La* **chiusura compatibile** $\to_R$ *di R e' la piu' piccola relazione su T t.c.*

- $M \to_R N$, *se* $M\ R\ N$
- $N\ M \to_R N\ M'$ *e* $M\ N \to_R M'\ N$, *se* $M \to_R M'$ *e* $N \in T$

*La R-***riduzione** $\to^*_R$ *e' la chiusura riflessiva e transitiva di $\to_R$, cioe' la piu' piccola relazione su T t.c.*

- $M \to^*_R N$, *se* $M \to_R N$
- $M \to^*_R M$, *se* $M \in T$
- $M \to^*_R P$, *se* $M \to^*_R N$ *e* $N \to^*_R P$

*La R-***equivalenza** $=_R$ *e' la chiusura riflessiva, simmetrica e transitiva di $\to_R$, cioe' la piu' piccola relazione su T t.c.*

- $M =_R N$, *se* $M \to_R N$
- $M =_R M$, *se* $M \in T$
- $N =_R M$, *se* $M =_R N$
- $M =_R P$, *se* $M =_R N$ *e* $N =_R P$

*Un termine $M \in T$ e' detto in R-***forma normale** *(R-nf) sse $M \not\to_R$, cioe' non esiste $N \in T$ t.c. $M \to_R N$. Si dice che N e' una R-nf di M se $M \to^*_R N \not\to_R$.*
La R-riduzione soddisfa la **proprieta' Church-Rosser** *(CR) [ovvero e' confluente] sse per ogni $M_0, M_1, M_2 \in T$ t.c. $M_0 \to^*_R M_1$ e $M_0 \to^*_R M_2$ esiste $M_3 \in T$ t.c. $M_1 \to^*_R M_3$ e $M_2 \to^*_R M_3$.*

Un modo piu' visivo di descrivere la proprieta' CR e' tramite un diagramma commutativo

$$\begin{tikzcd} M_{)} \arrow[r] \arrow[d] & M_1 \arrow[d, dashed] \\ M_2 \arrow[r, dashed] & M_3 \end{tikzcd}$$

dove le linee tratteggiate indica esistenza di una R-riduzione.

Theorem 6.7 *Se la R-riduzione soddisfa la proprieta' CR allora*

- *$M_1 =_R M_2$ sse esiste M t.c. $M_i \to_R^* M$ per $i = 1, 2$*
- *se M_1 e M_2 sono R-nf, allora $M_1 =_R M_2$ sse $M_1 \equiv M_2$*
- *un termine ha al piu' una R-nf.*

Vi sono condizioni sufficienti su R, molto semplici da verificare, che assicurano la proprieta' CR della R-riduzione.

Proposition 6.8 *Se la nozione di riduzione R soddisfa la proprieta' sostitutiva, cioe'*

- *$[N/x]M_1\ R\ [N/x]M_2$, se $M_1\ R\ M_2$ e $N \in T$ e x e' una variabile*

allora anche $\to_R$, $\to_R^*$ e $=_R$ la soddisfano.
Se la nozione di riduzione R non introduce nuove variabili libere, cioe'

- $\mathrm{FV}(M_2) \subseteq \mathrm{FV}(M_1)$, *se $M_1\ R\ M_2$*

allora anche $\to_R$ e $\to_R^*$ non le introducono.

La nozione di riduzione di interesse per studiare CL si ottiene orientanto i due assiomi equazionali (K) e (S) e chiudendo per sostituzione, e quindi la proprieta' sostitutiva vale per definizione.

Definition 6.9 *La* **nozione di riduzione** *CL e' data dalla seguente relazione*

$$\{(K\ M\ N, M) | M, N \in T\} \cup \{(S\ M\ N\ P, M\ P\ (N\ P)) | M, N, P \in T\}$$

Theorem 6.10 *La nozione di riduzione CL soddisfa le seguenti proprieta'*

- *$CL \vdash M = N$ sse $M =_{CL} N$;*
- *la CL-riduzione soddisfa la proprieta' CR;*
- *$(\lambda^* x.M)N \to_{CL}^+ [N/x]M$.*

La prima proprieta' correla la CL-uguaglianza alla teoria equazionale di CL, la seconda ci permette di usare la CL-riduzione per studiare la teoria equazionale di CL, la terza ci fornisce un rafforzamento della proprieta' $CL \vdash (\lambda^* x.M)N = [N/x]M$ dell'astrazione. E' anche facile dimostrare che $(\lambda_p^* x.M)$ e' sempre una CL-nf.
E' abbastanza facile dare una descrizione esplicita dei termini di CL che sono in CL-nf usando la seguente BNF

$$V \in NF ::= x\ \overline{V} \mid K \mid K\ V_1 \mid S \mid S\ V_1 \mid S\ V_1\ V_2$$

Inoltre vi sono termini di CL che non hanno CL-nf, p.e. $\Omega\ \Omega$ dove $\Omega \equiv (\lambda^* x.xx)$. E' immediato vedere che $\Omega\ \Omega \to_{CL}^+ \Omega\ \Omega$, inoltre analizzando in modo esaustivo gli M t.c. $\Omega\ \Omega \to_{CL}^* M$ si vede che $M \to_{CL}^* \Omega\ \Omega$.
La CL-riduzione e' molto conveniente per dimostrare che una certa equazione $M = N$ non e' dimostrabile in CL. Diamo alcuni esempi di tali applicazioni

- $[N/y](\lambda_p^* x.M) = (\lambda_p^* x.[N/y]M)$ non e' derivabile, p.e. si prenda $M \equiv x\ y$ e $N \equiv u\ v$.
- $(\lambda^* x.M) = (\lambda^* x.N)$ non e' derivabile da $M = N$, p.e. si prenda $K\ x\ y = x$.
- $(\lambda^* x.M) = (\lambda_p^* x.M)$ non e' derivabile, p.e. si prenda $M = y\ z$.

6.3 Strutture applicative sui termini

Come ogni teoria equazione, anche per CL si possono costruire dei modelli sintattici, i cui carrier sono classi di equivalenza di termini.

Example 6.11 Fissato un insieme X di variabili la struttura applicativa CL_X e' definita come segue

- $D = \{[M] | \mathrm{FV}(M) \subseteq X\}$, dove $[M]$ e' la classe di equivalenza i cui elementi sono termini N t.c. $CL \vdash M = N$ e $\mathrm{FV}(N) \subseteq X$
- $[M] \cdot [N] = [M\ N]$, e non dipende dalla scelta dei rappresentanti.

Se interpretiamo K e S con le classi di equivalenza $[K]$ e $[S]$, allora CL_X diventa una CA, inoltre $CL_\emptyset$ altro non e' che il modello iniziale di CL. Sfruttando le proprieta' della CL-riduzione e' immediato vedere che CL_X non e' banale, cioe' D ha almeno due elementi, anzi infiniti elementi. Infatti, se V_1 e V_2 sono CL-nf allora $[V_1] = [V_2]$ sse $V_1 \equiv V_2$.

Un'altra modo di costruire strutture applicative e' mediante una *semantica operazione* per i termini chiusi di CL. In tale contesto una semantica operazionale la si puo' identificare con una strategia di riduzione, cioe' una relazione *deterministica* (=grafo di una funzione parziale) $\rightarrow \subset \rightarrow_{CL}$.

Example 6.12 Introduciamo due semantiche operazioni a piccoli passi sui termini chiusi di CL

- la semantica operazionale call-by-name (CBN) $\rightarrow_N$ e' la relazione generata dalle seguenti regole

  $$\frac{}{K\ M_1\ M_2\ \rightarrow_N M_1} \qquad \frac{}{S\ M_1\ M_2\ M_3\ \rightarrow_N M_1\ M_3\ (M_2\ M_3)} \qquad \frac{M_1 \rightarrow_N M_1'}{M_1\ M_2 \rightarrow_N M_1'\ M_2}$$

  dove gli M variano sui termini chiusi.
- la semantica operazionale call-by-value (CBV) $\rightarrow_V$ e' la relazione generata dalle seguenti regole

  $$\frac{}{K\ V_1\ V_2\ \rightarrow_V V_1} \qquad \frac{}{S\ V_1\ V_2\ V_3\ \rightarrow_V V_1\ V_3\ (V_2\ V_3)} \qquad \frac{M_1 \rightarrow_V M_1'}{M_1\ M_2 \rightarrow_V M_1'\ M_2} \qquad \frac{M_2 \rightarrow_V M_2'}{V_1\ M_2 \rightarrow_V V_1\ M_2'}$$

  dove gli M variano sui termini chiusi, i V sulle CL-nf chiuse.

Se $\rightarrow_o$ e' una delle due semantiche operazioni CBN o CBV, allora possiamo definire una struttura applicativa parziale che indicheremo con pCL_o definita come segue

- $D_o = \{M | M \not\rightarrow_o\}$, cioe' M e' un termine chiuso dove la strategia di riduzione $\rightarrow_o$ non e' applicabile
- $M \cdot_o N = P$ sse $M\ N \rightarrow_o^* P \not\rightarrow_o$, poiche' $\rightarrow_o$ e' deterministica se P esiste e' unico.

pCL_V (con l'interpretazione ovvia di K e S) e' una pCA, mentre pCL_N non e' una pCA, poiche' si possono trovare termini $M_1, M_2, M_3 \in D_N$ t.c. $M_1 \cdot_o M_3 = K\ K$ [e quindi $S \cdot_N M_1 \cdot_N M_2 \cdot_N M_3 = K$], mentre $M_2 \cdot_N M_3$ e' indefinito [e quindi $M_1 \cdot_N M_3 \cdot_N (M_2 \cdot_N M_3)$ e' indefinito].
Esempi analoghi di strutture applicative parziali si ottengono a partire dalla semantica operazione CBV e CBN per un linguaggio funzionale non-tipato. Anche in questo caso, solo la struttura applicativa parziale indotta dalla semantica operazionale CBV e' una pCA.

Example 6.13 Si estenda il linguaggio di CL con una costante $\bot$ (che rappresenta la non terminazione), e si aggiunga alla semantica operazionale call-by-name (CBN) $\rightarrow_N$ la regola $\dfrac{}{\bot \rightarrow_N \bot}$.
Possiamo definire una struttura applicativa totale che indicheremo con CL_N definita come segue

- $D_t = \{M | M \not\rightarrow_N\} \cup \{\bot\}$
- $M \cdot_t N = \begin{cases} P & \text{se } M\ N \rightarrow_N^* P \not\rightarrow_N \\ \bot & \text{altrimenti} \end{cases}$

In analogia con quanto visto per la struttura applicativa parziale $(D_N, \cdot_N)$, anche in questo caso non vale l'assioma $S\ x\ y\ z = x\ z\ (y\ z)$.

6.4 Potere espressivo di CL

6.4.1 rappresentazioni di tipi di dato

L'operazione di astrazione ci permette di considerare varie *estensioni definizionali* di CL del tipo nuova costante c e nuovo assioma equazionale $c\ \overline{x} = M$ dove M e' un termine di CL t.c. $\mathrm{FV}(M) \subseteq \overline{x}$. Infatti, possiamo interpretare c con il termine $(\lambda^*\overline{x}.M)$ di CL. Sfruttando questo tipo di estensioni definizionali, e' possibile codificare in modo molto naturale vari tipi di dato caratterizzati da *proprieta' universali.*

- Il tipo di dato *Bool* dei booleani e' specificato da due costanti T, F: *Bool* ed un operatore *polimorfo* $If_X: Bool, X, X \rightarrow X$ t.c. $If(T, x, y) = x$ e $If(F, x, y) = y$.

In effetti, la proprieta' universale di $Bool$ dice che per ogni insieme X ed ogni coppia di elementi $x, y{:}\,X$ esiste ed e' unica $h{:}\,Bool \to X$ t.c. $h\ T = x$ e $h\ F = y$. In modo diagrammatico la proprieta' universale la si puo' esprimere mediante il seguente diagramma commutativo

$$\begin{tikzcd} 1 \arrow[r, "T"] \arrow[dr, "x"'] & Bool \arrow[d, dashed] & 1 \arrow[l, "F"'] \arrow[dl, "y"] \\ & X & \end{tikzcd}$$

dove la linea tratteggiata indica esistenza ed unicita' di una funzione.

L'operatore If permette di dimostrare l'esistenza di h, poiche' $h\ b = If\ b\ x\ y$ soddisfa le proprieta' richieste. Si osservi che per esprimere l'unicita' di h e' necessario aggiungere altri assiomi a quelli di If.

In CL possiamo *rappresentare* T, F e If nel modo seguente $T\ x\ y = x$, $F\ x\ y = y$, $If\ b\ x\ y = b\ x\ y$. E' immediato verificare che le equazioni $If\ T\ x\ y = x$ e $If\ F\ x\ y = y$ sono derivabile in CL a partire dalle equazioni che definiscono T, F e If.

- Il tipo di dato $X_0 \times X_1$ delle coppie specificato dagli operatori $Pair{:}\, X_0, X_1 \to X_0 \times X_1$, $\pi_0{:}\, X_0 \times X_1 \to X_0$ e $\pi_1{:}\, X_0 \times X_1 \to X_1$ t.c. $\pi_i(Pair(x_0, x_1)) = x_i$. La proprieta' universale per il tipo di dato $X_0 \times X_1$ e' data da

$$\begin{tikzcd} X_0 & X_0 \times X_1 \arrow[l, "\pi_0"'] \arrow[r, "\pi_1"] & X_1 \\ & X \arrow[ul, "f"] \arrow[u, dashed] \arrow[ur, "g"'] & \end{tikzcd}$$

  La rappresentazione di tali operatori in CL e' definita nel modo seguente $Pair\ x\ y\ z = z\ x\ y$ e $\pi_i\ p = p\ (\lambda^* x_0, x_1.x_i)$. Anche in questo caso la derivabilita' dell'equazioni $\pi_i\ (Pair\ x_0\ x_1) = x_i$ e' immediata.

- Il tipo di dato Nat dei numeri naturali e' specificato da $0{:}\, Nat$, $Succ{:}\, Nat \to Nat$ e $It_X{:}\, (X \to X), X, Nat \to X$ t.c. $It(f, x, 0) = x$ e $It(f, x, Succ\ n) = f(It(f, x, n))$. Se identifichiamo il naturale n con $Succ^n(0)$, allora abbiamo che $It(f, x, n) = f^n(x)$. La proprieta' universale per il tipo di dato Nat e' data da

$$\begin{tikzcd} 1 \arrow[r, "0"] \arrow[dr, "x"'] & Nat \arrow[r, "Succ"] \arrow[d, dashed] & Nat \arrow[d, dashed] \\ & X \arrow[r, "f"'] & X \end{tikzcd}$$

  La codifica dei numeri naturali in CL proposta da Church e' $\underline{n}\ f\ x = f^n x$ dove $f^0 x \equiv x$ e $f^{n+1} x \equiv f(f^n x)$. Questo suggerisce la seguente rappresentazione degli operatori: $0\ f\ x = x$, $Succ\ n\ f\ x = f\ (n\ f\ x)$, $It\ f\ x\ n = n\ f\ x$. Anche in questo caso e' immediato derivare le equazioni desiderate.

  Usando la codifica delle coppie e' possibile ricavare da It_X un operatore $R_X{:}\, (X, Nat \to X), X, Nat \to X$ di ricorsione primitiva t.c. $R(f, x, 0) = x$ e $R(f, x, succ\ n) = f(R(f, x, n), n)$. R risulta essere piu' conveniente di It per definire funzioni sui numeri naturali.

Come tutte le codifiche, anche queste presentano un certo grado di arbitrarieta'. Data una pCA, e' possibile associargli una categoria simile a quella degli insiemi, ispirata alla nozione di *realizzabilita'* introdotta da Kleene. In tale categoria, si puo' far vedere che differenti codifiche di booleani (coppie o naturali) corrispondono ad oggetti, che soddisfano le stesse proprieta' universali, e quindi isomorfi tra loro.

Definition 6.14 *Data una pCA $(D, \cdot)$, la categoria dei D-set e' definita come segue:*

- *un oggetto e' una coppia $\underline{X} = (X, \Vdash_X)$, dove X e' un insieme e $\Vdash_X \subseteq D \times X$ e' una relazione t.c. $\forall x \in X.\exists d \in D.d \Vdash_X x$ [si dice che d realizza x]*

- *un morfismo $f{:}\, \underline{X} \to \underline{Y}$ e' una funzione $f{:}\, X \to Y$ per cui esiste un $e \in D$ t.c. $\forall x \in X.\forall d \in D.d \Vdash_X x \Longrightarrow e \cdot d \Vdash_Y f(x)$ [si dice che e realizza f]*

- *la definizione di composizione [ed identita'] coincide con quella insiemistica.*

Remark 6.15 Se $(D, \cdot)$ e' solo una struttura applicativa parziale la definizione di D-set ha senso, ma non e' detto che i morfismi includano le identita' e siano chiusi per composizione, quindi non e' detto che si abbia una categoria.

Proposition 6.16 *Data una pCA $(D, \cdot)$, la categoria dei D-set e' una categoria cartesianamente chiusa (CCC) con somme e oggetto dei numeri naturali (NNO). Inoltre la categoria dei D-set ha due* full sub-CCC *[cioe' sono sotto-categorie piene che ereditano la struttura di CCC] i cui oggetti sono rispettivamente*

- *gli $(X, \Vdash_X)$ t.c. $\exists d \in D.\forall x \in X.d \Vdash_X x$, cioe' vi e' un d che realizza tutti gli elementi di X. Tali oggetti $\underline{X}$ sono caratterizzati dalla proprieta' di* non avere contenuto computazionale*, ovvero ogni $f: Y \to X$ e' un morfismo $f: \underline{Y} \to \underline{X}$, per qualunque D-set $\underline{Y}$. Da questa proprieta' segue immediatamente che la categoria degli oggetti senza contenuto computazionale e'* equivalente *alla categoria degli insiemi.*
- *gli $(X, \Vdash_X)$ t.c. $\forall x_1, x_2 \in X.\forall d \in D.d \Vdash_X x_1 \wedge d \Vdash_X x_2 \Longrightarrow x_1 = x_2$, cioe' ogni d realizza al piu' un $x \in X$. Tali oggetti sono detti* **effettivi***.*

Proof Ci limitiamo a descrivere prodotti, somme, esponenziali e NNO nella categoria dei D-set, tralasciando la verifica delle proprieta' universali.

- oggetto terminale $\underline{1} \stackrel{\Delta}{=} (\{*\}, D \times \{*\})$, e' immediato dalla definizione che $\underline{1}$ e' un oggetto effettivo senza contenuto computazionale
- oggetto iniziale $\underline{0} \stackrel{\Delta}{=} (\emptyset, \emptyset)$, anche in questo caso e' e' immediato che $\underline{0}$ e' un oggetto effettivo senza contenuto computazionale
- prodotto $\underline{X} \times \underline{Y} \stackrel{\Delta}{=} (X \times Y, \Vdash)$, dove $d \Vdash (x, y) \stackrel{\Delta}{\Longleftrightarrow} \exists d_0, d_1 \in D.d_0 \Vdash_X x \wedge d_1 \Vdash_Y y \wedge d = Pair\ d_0\ d_1$, nella definizione di $\Vdash$ si usa la codifica delle coppie
- coprodotto/somma $\underline{X} + \underline{Y} \stackrel{\Delta}{=} (X + Y, \Vdash)$, dove $d \Vdash in_0(x) \stackrel{\Delta}{\Longleftrightarrow} \exists d_0 \in D.d_0 \Vdash_X x \wedge d = Pair\ F\ d_0$ e $d \Vdash in_1(y) \stackrel{\Delta}{\Longleftrightarrow} \exists d_1 \in D.d_1 \Vdash_Y y \wedge d = Pair\ T\ d_1$, nella definizione di $\Vdash$ si usa la codifica delle coppie e quella dei booleani. Gli oggetti effettivi sono chiusi rispetto al $=$, cioe' se $\underline{X}$ e $\underline{Y}$ sono effettivi, allora anche $\underline{X} + \underline{Y}$ lo e'. Invece, gli oggetti senza contenuto computazionale non sono chiusi rispetto a $+$, p.e. $\underline{1} + \underline{1}$.
- esponenziale $\underline{Y}^{\underline{X}} \stackrel{\Delta}{=} (\{f | f: \underline{X} \to \underline{Y}\}, \Vdash)$, dove $e \Vdash f \stackrel{\Delta}{\Longleftrightarrow} e$ realizza il morfismo f
- NNO $\underline{N} \stackrel{\Delta}{=} (N, \Vdash)$, dove $d \Vdash n \stackrel{\Delta}{\Longleftrightarrow} d = \underline{n}$, nella definizione di $\Vdash$ si usa la codifica dei numeri naturali. $\underline{N}$ e' un oggetto effettivo.

∎

6.4.2 Definizioni ricorsive e operatori di punto fisso

Le definizioni ricorsive sono un meccanismo molto potente adottato nei linguaggi di programmazione. Mentre una definizione nel senso matematico del termine e' della forma $c = M$, dove in M si usano solo concentti precedentemente definiti (e quindi non si puo' usare la costante c che si sta definendo), una definizione ricorsiva permette di *definire* c anche in termini di se stessa.

Sfruttando l'operazione di astrazione possiamo riscrivere una definizione ricorsiva nella forma $c = f\ c$, in modo che c non venga usata in f. In questo caso si dice che c e' un **punto fisso** di f. Facciamo vedere che in CL esiste un **operatore di punto fisso** Y t.c. $Y\ f = f(Y\ f)$, ovvero Y restituisce un punto fisso per f.

Consideriamo due combinatori di punto fisso definibili in CL:

- il combinatore $Y_{curry} \equiv (\lambda^* f.V\ V)$ dove $V \equiv \lambda^* y.f(y\ y)$, soddisfa la proprieta' $Y\ f = f(Y\ f)$ ma non la proprieta' piu' forte $Y\ f \to^*_{CL} f(Y\ f)$
- il combinatore $Y_{turing} \equiv Z\ Z$ dove $Z \equiv \lambda^* y, f.f(y\ y\ f)$, soddisfa la proprieta' $Y\ f \to^+_{CL} f(Y\ f)$.

In entrambi i casi e' assai facile generalizzare i combinatori di punto fisso per trattare definizioni mutuamente ricorsive, che possono essere sempre messi nella forma $(c_i = f_i\ \overline{c} | i \in n)$. In tal caso avremo bisogno di una famiglia di n combinatori $Y_{n,i}$ con $i \in n$ t.c. $Y_{n,i}\ \overline{f} = f_i(Y_{n,j}\ \overline{f})_{j \in n}$ per ogni $i \in n$. Una possibile definizione di tali combinatori, che generalizza la definizione di Y_{curry}, e' data da $Y_i \equiv (\lambda^* \overline{f}.V_i\ \overline{V})$ dove $V_i \equiv (\lambda^* \overline{y}.f_i(y_j\ \overline{y})_{j \in n})$.

Avendo a disposizione un combinatore di punto fisso e' abbastanza plausibile aspettarsi che tutte le *funzioni calcolabili* sui numeri naturali codificati con i numerali di Church siano *definibili* in CL.

Definition 6.17 *Data una funzione parziale* $f: N^m \rightharpoonup N$ *ed un termine chiuso* M *di* CL*, si dice che* M **rappresenta/definisce** f *sse per ogni* $\overline{n} \in N^m$

- $f(\overline{n})$ *e' definita ed uguale ad* k *e* $CL \vdash M\ \underline{\overline{n}} = \underline{k}$
- *oppure* $f(\overline{n})$ *e' indefinita e il termine* $M\ \underline{\overline{n}}$ *non ha* CL*-nf.*

Si osservi che i numerali di Church sono in CL-nf, e quindi $CL \vdash \underline{m} = \underline{n}$ sse $m = n$.

Theorem 6.18 *Data una funzione* $f: N^m \rightharpoonup N$ *si ha che* f *e' Turing calcolabile sse e' definibile in* CL.

6.5 Estensioni di CL

Consideriamo alcuni assiomi (al primo ordine) per CL e piu' in generale per strutture applicative.

6.5.1 Estensioni inconsistenti

E' facile vedere che $CL + (K = S)$ e' **equazionalmente inconsistente**, ovvero $CL + (K = S) \vdash x = y$. Un risultato leggermente piu' forte (in cui non si richiede che l'applicazione sia una operazione totale) e': se in una pCA D si ha che $K = S$, allora D ha un solo elemento. L'inconsistenza equazionale segue in modo ovvio da $CL + (K = S) \vdash I = x$, quindi vediamo la dimostrazione $I = x$.

- $CL \vdash I\ I\ x = x$ segue facilmente da $I\ M = M$;
- $CL + (K = S) \vdash I\ I\ x = I$ invece usa l'assioma $K = S$ e la definizione di $I \equiv S\ K\ K$

  $I\ I\ x =$ per definizione di I

  $(S\ K\ K)\ I\ x =$ per $K = S$

  $(K\ K\ K)\ I\ x =$ per gli assiomi di CL

  $K\ I\ x = I$.

Vi e' un risultato generale dovuto a Böhm che ci dice quando e' equazionalmente inconsistente aggiungere un assioma $M = N$ con M ed N CL-nf. In particolare, si ha che $CL + (\underline{m} = \underline{n})$ e' equazionalmente inconsistente quando $m \neq n$. Un'altra estensione inconsistente di CL e' l'aggiunta di un costante if_{eq} per effettuare il test di uguaglianza. Gli assiomi (nella logica del primo ordine) per if_{eq} sono
(ifeq.true) $\forall x, y, u, v. x = y \Longrightarrow if_{eq}\ x\ y\ u\ v = u$ (ifeq.false) $\forall x, y, u, v. x \neq y \Longrightarrow if_{eq}\ x\ y\ u\ v = v$
Per dimostrare l'inconsistenza equazionale e' piu' semplice ragionare in termini di modelli, quindi supponiamo che D sia una CA con almeno due elementi $a \neq b$ e che sia anche un modello per if_{eq}, facciamo vedere che abbiamo un assurdo:

- usando l'operazione di astrazione (che vale in CL anche in presenza di costanti addizionali) possiamo trovare un $f \in D$ t.c. $f\ x = if_{eq}\ x\ a\ b\ a$;
- usando un combinatore di punto fisso possiamo provare un punto fisso $p \in D$ t.c. $f\ p = p$;
- facciamo vedere che si ha un assurdo, considerando i due casi possibili $p = a$ e $p \neq a$.

  Se $p = a$ allora $a = p = f\ p = if_{eq}\ a\ a\ b\ a = b$, che contraddice $a \neq b$.

  Se $p \neq a$ allora $p = f\ p = if_{eq}\ a\ p\ b\ a = a$, che contraddice $p \neq a$.

E' tuttavia facile dimostrare che e' possibile avere pCA non banali con un elemento che soddisfa gli assiomi per if_{eq}. Per esempio, la struttura applicativa di Kleene $(N, \cdot)$. Infatti la funzione $f(x, y, u, v) =$ if $x = y$ then u else v e' ovviamente ricorsiva, e $m \in N$ che la rappresenta soddisfa gli assiomi per if_{eq}.

6.5.2 Estensionalita'

Un assioma molto naturale per le strutture applicative (sia totali che parziali) e' l'assioma di estensionalita'
(ext) $\forall x, y. (\forall z. x\ z = y\ z) \Longrightarrow x = y$ o (p.ext) $\forall x, y. (\forall z. x\ z \simeq y\ z) \Longrightarrow x = y$ nel caso parziale.
Il significato intuitivo dell'estensionalita' in una struttura applicativa $(D, \cdot)$ e' "se due programmi x e y rappresentano la stessa funzione $f: D \rightharpoonup D$ allora sono uguali". Quindi in presenza di estensionalita' si ha una corrispondenza biunivoca tra funzioni rappresentabili ed elementi di D. Inoltre le varie operazioni di astrazione definite in CL

diventano dimostrabilmente equivalenti, cioe' $CL + (ext) \vdash \lambda^* x.M = \lambda^*_p x.M$, e si comportano in modo estensionale, cioe' $CL + (ext) \vdash (\forall x.M = N) \Longrightarrow \lambda^* x.M = \lambda^* x.N$.

Una proprieta' leggermente piu' debole della estensionalita' (e che e' usata per dare una assiomatizzazione al primo ordine dei modelli del λ-calcolo non tipato) e' l'esistenza di un combinatore ϵ di scelta di un *rappresentante canonico*:

(ϵ) $\forall x, y. x\ y = \epsilon\ x\ y$ $\quad$ $(\epsilon.\mathrm{ext})$ $\forall x, y. (\forall z. x\ z = y\ z) \Longrightarrow \epsilon\ x = \epsilon\ y$

Il primo assioma dice che x ed il **rappresentante canonico** $\epsilon\ x$ rappresentano la stessa funzione, mentre il secondo assioma dice se due programmi x ed y rappresentano la stessa funzione allora i rappresentanti canonici $\epsilon\ x$ e $\epsilon\ y$ sono uguali. Se una CA soddisfa (ext), allora un combinatore di scelta ϵ esiste ed e' necessariamente I. Data una struttura applicativa $(D, \cdot)$ e' sempre possibile definire una funzione di scelta $f : D \to D$ di un rappresentante canonico, ma in generale tale f non e' rappresentabile. Gli assiomi per ϵ ci assicurano l'esistenza di una funzione di scelta che e' anche rappresentabile.