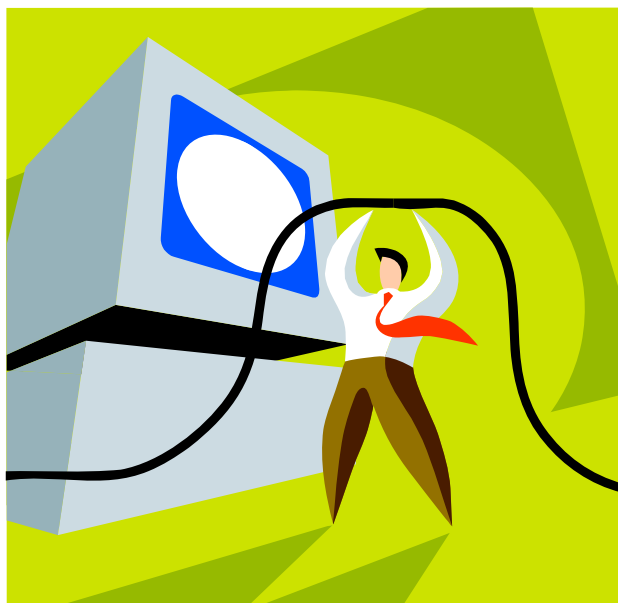




**Università degli Studi di Roma “La Sapienza”
Corso di Laurea in Informatica**

INSEGNAMENTO DI SEMANTICA DEI LINGUAGGI DI PROGRAMMAZIONE

**Anno 2004-2005
Prof. Anna LABELLA**



A cura di Stefano Melmeluzzi

0. Introduzione

Il corso si propone di esaminare, da un punto di vista algebrico, alcuni concetti che sono alla base dell'informatica.

L'informatica, infatti, in un primo tempo si è sviluppata con l'analisi numerica, che ha fornito dei metodi di computazione e, successivamente, con lo sviluppo dei *linguaggi artificiali*, si è evoluta verso un livello crescente di astrazione, avvicinandosi sempre di più ad una descrizione algebrica.

L'algebra si propone di studiare la “teoria delle operazioni”, ma noi vorremo considerarla insieme ad una “teoria dell'ordine”.

Tale classificazione delle teorie matematiche è stata sviluppata dalla prestigiosa scuola matematica Nicolas Bourbaki (metà XX secolo), nella quale, dopo attente revisioni, si è arrivati a proporre la suddivisione della matematica in tre filoni principali:

Teoria delle operazioni

Teoria dell'ordine

Topologia

La prima si occupa di teoria degli insiemi e di strutture algebriche (gruppi, anelli, campi, etc.). La seconda tratta gli insiemi ordinati o parzialmente ordinati. La terza presenta dei forti legami con le prime due: ad esempio basti pensare che gli aperti di uno spazio topologico sono dei particolari sottoinsiemi dello spazio e quindi si comportano come le parti di un insieme sui quali possiamo definire delle operazioni che inducono topologie e viceversa.

Tuttavia la topologia presenta delle peculiarità che la differenziano dalle due precedenti teorie.

Il tentativo di classificare gli aspetti fondamentali dell'algebra in alcune teorie fondamentali risponde all'esigenza di un crescente livello di astrazione, ma spesso questo porta a dimenticare l'aspetto “empirico” della matematica, necessario specialmente nella fase di apprendimento.

Un esempio di come la matematica risulti essere una “traduzione” ad un livello più astratto di concetti reali, concreti, può essere fornito dalla nozione di *numero*.

Il numero nasce dalla necessità di avere un controllo sull'ordine e sulla quantità degli oggetti, basti pensare, per esempio, alle prime forme di commercio e quindi al problema di quantificare il valore degli oggetti da barattare.

Il concetto di numero nasce da una esperienza concreta e si sviluppa in astratto seguendo l'esigenza di eseguire dei calcoli. Il concetto di calcolo deve poi essere inteso in astratto e quindi separato dal particolare procedimento risolutivo, *algoritmo*, che si adopera per eseguirlo.

Esempio:

Se vogliamo eseguire la somma $3 + 5$ basta contare scorrendo in sequenza 3 dita di una mano e 5 dell'altra, ma per sommare $19 + 114$ abbiamo bisogno di un algoritmo: eseguiamo la somma delle unità e del risultato ottenuto scriviamo l'unità memorizzando la eventuale decina come riporto da sommare alle altre decine e così via.

In generale possiamo procedere scomponendo il problema in sotto-problemi più semplici (ad es.: $7253 + 3128 = 7253 + 8 + 20 + 100 + 3000$). Tuttavia sono molte le operazioni che non siamo in grado di compiere facilmente, pur avendo a disposizione un algoritmo (ad esempio l'estrazione della radice quadrata di un numero).

Come l'uso delle cifre indiane ha permesso di abbandonare l'abaco, ma ha richiesto lo sviluppo di

nuovi algoritmi, così il moderno *calcolatore o computer*, che ci dà un valido aiuto nel calcolo, ci forza ad escogitare nuovi algoritmi particolarmente adatti alla sua struttura.

Un calcolatore opera in maniera sequenziale, ha poca capacità di sintesi ma ha una caratteristica fondamentale che gli permette di immagazzinare una grande quantità di dati: la *memoria*.

Al contrario, l'uomo fa uso della propria capacità *sincronica*, che gli permette, una volta acquisita una visione globale dei dati, di farne una sintesi.

Il calcolatore opera in base 2, ovvero con sequenze di 0 e di 1, dunque abbiamo bisogno di opportuni algoritmi che, utilizzando soltanto questa numerazione, portino la macchina a simulare il nostro modo di pensare. Data però la nostra difficoltà nel trattare il linguaggio binario, utilizziamo vari *metalinguaggi* per far arrivare la macchina ad un livello di linguaggio più elevato e quindi più simile al nostro. Si è così sviluppata la *teoria dei linguaggi artificiali*.

1. I numeri naturali

La prima assiomatizzazione è dovuta a Campano da Novara (metà del XIII secolo) il quale affermava che: "Si dice serie naturale dei numeri quella per la quale il calcolo degli stessi avviene aggiungendo un'unità". A questa definizione aggiungeva i seguenti assiomi (*petitiones*):

- Di ogni numero si possono prendere quante copie si vuole o quanti multipli si vuole.

Infatti, dato che il numero è trattato come un termine del linguaggio, un numero può essere ripetuto quante volte si vuole.

- La serie dei numeri può procedere all'infinito.
- Nessun numero può essere diminuito all'infinito.

In questo modo egli ha fornito una prima forma del Principio di Induzione. Questa assiomatizzazione, pur non caratterizzando completamente i numeri naturali, ne offre un procedimento costruttivo di generazione.

1.1 Principio di induzione

Il Principio di induzione può essere enunciato come segue:

Proposizione 1:

Sia P una certa proprietà.

Se $P(0)$ è vera (ovvero P è vera per il primo numero).

se P essendo vera per n , è vera per il successore di n , $s(n)$, allora P è vera per tutti i numeri naturali.

Dimostrazione: (Spiegazione usando l'intuizione di Campano da Novara)

Supponiamo, per assurdo, che esista $m \in \mathbb{N}$ tale che $P(m)$ non sia vera. Allora, indicato con $p(m)$ il predecessore di m , avremmo che $P(p(m))$ non è vera (altrimenti per l'ipotesi induttiva anche $P(m) = P(s(p(m)))$ sarebbe vera).

Allora anche $P(p(p(m)))$ non è vera, e così via. Questo modo di procedere non può continuare all'infinito per la quarta delle precedenti affermazioni. Ne segue che, ammessa l'esistenza di un primo numero 0, si arriverebbe all'assurdo che $P(0)$ sarebbe falsa contro le ipotesi. ■

Il principio di induzione può essere equivalentemente espresso nel seguente modo:

Proposizione 2:

Sia A un sottoinsieme di \mathbb{N} tale che:

- $0 \in A$
- $\forall n \in A$ si ha $s(n) \in A$ (cioè A è chiuso rispetto alla funzione successore), allora $A = \mathbb{N}$

Si può definire sui numeri naturali \mathbb{N} la seguente terna di funzioni ($s, 0, +$) ovvero:

- 1) una funzione iniettiva (che prende il nome di successore):

$$s : \mathbb{N} \rightarrow \mathbb{N}$$

$$\eta \mapsto s(\eta)$$

2) una costante $0 \in \mathbb{N}$ e $0 \notin s(\mathbb{N})$ (cioè 0 non è successore di alcun numero);

3) ed una funzione *somma*

$$\begin{aligned} + : \mathbb{N} \times \mathbb{N} &\rightarrow \mathbb{N} \\ (m, n) &\mapsto m + n \end{aligned}$$

Avendo 1) e 2) possiamo definire la somma per induzione.

Il principio di induzione ci permette di dimostrare una proposizione o definire una proprietà per un insieme infinito di elementi limitandoci ad un numero finito di casi. Ciò si traduce ad esempio nel definire le funzioni per ricorsività.

La somma è definita come una funzione di due variabili, m ed n . Dal momento che il prodotto cartesiano ci permette di far variare indipendentemente gli oggetti che lo compongono, possiamo pensare $+$ come una funzione che varia prima su n , visto come una variabile e con m , visto come un parametro e poi viceversa. Dunque fissato il primo termine m , definiamo la seguente funzione per ricorrenza

$$\begin{aligned} m + _ : \mathbb{N} &\rightarrow \mathbb{N} \\ 0 &\mapsto m + 0 = m \\ s(h) &\mapsto m + s(h) = s(m + h) \end{aligned}$$

Una volta che si è definita, mediante il principio di induzione, la funzione sulla variabile n , pensiamo questa ultima come un parametro e ripetiamo lo stesso ragionamento facendo variare m .

Definita la somma, possiamo dimostrare che il successore di m è proprio $m+1$:

$$\begin{aligned} 1+0 &= 1 \\ m + 1 &= m + s(0) = s(m + 0) = s(m) \end{aligned}$$

Esercizio: Dimostrare che la somma verifica le proprietà: associativa. commutativa. esistenza dell'elemento neutro.

In \mathbb{N} possiamo introdurre un *ordinamento* nel seguente modo:

$$n \leq M \stackrel{def}{\Leftrightarrow} \exists h \in \mathbb{N} \text{ tale che } n+h=m$$

Esercizio:

Verificare che:

- \leq è una relazione d'ordine, ovvero che è riflessiva, antisimmetrica e transitiva;
- $\forall n \in \mathbb{N}^+ = \mathbb{N} - \{0\}$ risulta $0 < n$;
- $\forall n \in \mathbb{N}$ risulta $n < s(n)$.

In conclusione, tutte le proprietà dei numeri naturali si possono ritrovare partendo dai 5 assiomi definiti da Giuseppe Peano (1858-1932).

Ad esempio, si può far vedere che la somma è monotona rispetto all'ordine, ovvero:

$$\forall n, n' \in \mathbb{N} \text{ tali che } n \leq n' \text{ e } \forall m \in \mathbb{N} \Rightarrow n + m \leq n' + m$$

Riassumeremo con la scrittura $(\mathbb{N}, +, 0, \leq)$ la struttura considerata.

Osservazione:

Nella definizione di somma abbiamo usato il principio di induzione su $\mathbb{N} \times \mathbb{N}$, dotato dell'ordinamento *lessicografico* così definito:

$$\forall a, b, c, d \in \mathbb{N} \quad (a, b) \leq (c, d) \stackrel{\text{def}}{\Leftrightarrow} a < c \text{ oppure } a = c \text{ e } b \leq d$$

Questo risulta essere un ordinamento totale.

Un esempio di tale ordinamento è quello del vocabolario.

In $\mathbb{N} \times \mathbb{N}$ possiamo anche definire un altro ordinamento, chiamato *ordinamento prodotto*, tale che:

$$\forall n, m, n', m' \in \mathbb{N} \quad (n, m) \leq (n', m') \stackrel{\text{def}}{\Leftrightarrow} n \leq n' \text{ e } m \leq m'$$

Tale ordinamento, pur essendo parziale, può risultare molto utile, come vedremo in seguito, in quanto opera indipendentemente sulle componenti.

2. Semantica dei linguaggi di Programmazione

È noto che un linguaggio di programmazione serve ad esprimere certi processi di calcolo che un elaboratore dovrà svolgere.

Il problema di dare una semantica a un linguaggio di programmazione consiste nel trovare il modo di specificare senza ambiguità qual è il significato di un programma scritto nel linguaggio stesso.

I pregi di una definizione formale della semantica di un linguaggio di programmazione sono molteplici e li elenchiamo qui di seguito.

1. Innanzitutto ne beneficia l'utente del linguaggio, e cioè il programmatore, che viene messo in grado di comprendere con precisione il suo strumento di lavoro.
2. Ne beneficia chi dovrà implementare il linguaggio su uno specifico calcolatore. Questo perché un linguaggio di programmazione ad alto livello, tramite procedimenti di traduzione e compilazione possa essere utilizzato su elaboratori diversi. Inoltre il realizzatore di un linguaggio, che ne costruisce il traduttore, trova ovvi benefici dalla definizione formale della sua semantica, in quanto essa rappresenta uno standard realizzativo.
3. Consente l'analisi di programmi, cioè la verifica formale di proprietà; ad esempio per applicazioni critiche.
4. Infine permette di stabilire la correttezza di compilatori, sia in fase di traduzione che di ottimizzazione : studio di equivalenza di programmi. Ad esempio si può dire che due programmi sono equivalenti se producono lo stesso risultato, oppure quando è possibile sostituire l'uno con l'altro in ogni contesto senza produrre effetti "visibili".

2.1 Semantica statica

I programmi vengono trattati come oggetti matematici, e poiché gli oggetti matematici hanno un tipo, una semantica formale deve dunque assegnare un tipo ai termini del linguaggio.

Il sistema dei tipi di un linguaggio di programmazione ovvero l'insieme delle regole che consentono di dare un tipo ad espressioni, comandi ed altri costrutti di programmazione rappresenta la semantica statica. Lo scopo principale di un sistema di tipi è quello di prevenire il verificarsi di errori durante l'esecuzione di un programma. Ad esempio l'applicazione di funzioni non propri, riferimenti illegali della memoria e tentativi di dividere un numero per zero.

2.2 Semantica dinamica

Il significato di un programma consiste nel suo comportamento durante l'esecuzione. Può essere informale come nei manuali di programmazione, o formale, cioè applicando nozioni e tecniche della matematica o della logica. Vi sono tre approcci : quello *operazionale*, quello *denotazionale* ed infine quello *assiomatico*.

2.2.1 Semantica operazionale

Il significato di un programma viene descritto da una relazione matematica tra i termini del linguaggio che rappresenta una nozione di valutazione.

Una forma "primitiva" di semantica operazionale, consisteva nel definire una macchina in grado di eseguire il linguaggio (per esempio una macchina di Turing). Il suo punto debole consisteva nel fatto che il significato di un programma dipendesse da qualcosa di "esterno" al linguaggio, cioè il funzionamento di una specifica macchina.

In base alla struttura sintattica dei programmi si definisce la relazione di valutazione : ad esempio, sia $Env \times Term$ l'insieme delle “configurazioni”, dove $Env = Var \xrightarrow{fin} Const$ è l'insieme degli ambienti che associano variabili a costanti, e $Term$ l'insieme dei termini del linguaggio; una relazione \longrightarrow tra configurazioni può essere definita da regole del tipo :

$$\text{Se } E_1, e_1 \longrightarrow E_2$$

Cosicché il significato della espressione:

$$(4 + x) - 3 \text{ nell'ambiente } \{(x, 2)\}$$

è la computazione:

$$\{(x, 2)\}, (4 + x) - 3 \rightarrow \{(x, 2)\}, (4 + 2) - 3 \rightarrow \{(x, 2)\}, (6) - 3 \rightarrow \{(x, 2)\}, 3$$

Questo tipo di valutazione viene chiamata a piccoli passi, perché rappresenta passi elementari di computazione, mentre quella a grandi passi, la relazione di valutazione associa termini a risultati.

2.2.2 Semantica denotazionale

Il significato di un programma si ottiene interpretando i termini del linguaggio in oggetti matematici: numeri, elementi di strutture algebriche e così via.

Ad esempio se $\rho = (x, 2)$ è l'ambiente semantico che associa alla variabile x il valore 2 (nota la differenza con la costante 2 della semantica operativa), allora:

$$\llbracket (4 + x) - 3 \rrbracket_\rho = \llbracket 4 + x \rrbracket_\rho - \llbracket 3 \rrbracket_\rho = 3$$

dove con $\llbracket t \rrbracket_\rho$ si intende il significato di t nell'ambiente ρ .

Si evidenzia una caratteristica fondamentale della semantica denotazionale, cioè la composizionalità. Dove il significato di una espressione complessa come $(4 + x) - 3$ è una funzione delle sue componenti più semplici :

$$(4+x) \text{ e } 3$$

La semantica denotazionale risulta più astratta di quella operativa, inoltre fornisce un ordine di valutazione degli operandi.

2.2.3 Semantica assiomatica

Il significato di un programma è determinato nell'ambito di una teoria assiomatica, dall'insieme delle proposizioni vere per il programma stesso. Quindi scrivere $\{A\} p \{B\}$ viene letta : “Se il programma p , eseguito in uno stato che soddisfa la proprietà A , termina, allora lo stato che raggiunge soddisfa la proprietà B ”.

Questo corso si occuperà di alcune tecniche della Semantica Denotazionale.

Una delle proprietà della Semantica Denotazionale è specificare i costrutti dei linguaggi di programmazione in maniera astratta ed indipendente dall'implementazione. In questo modo si possono comprendere i concetti fondamentali dei linguaggi di programmazione, delle loro relazioni.

In ogni modo è importante verificare che le specifiche denotazionali dei linguaggi siano implementabili, confrontando la semantica denotazionale con la semantica operativa.

2.3 Caratteristiche della Semantica Denotazionale

Ogni frase o parte di programma P , è costituito da una denotazione $\llbracket P \rrbracket$, un oggetto matematico che rappresenta il contributo di P di alcune occorrenze di parti di programma.

La denotazione di una frase è determinata dalla denotazione delle sue sottofrasi, da qui proviene che la semantica è composizionale.

Esempio:

Data la funzione parziale:

$$\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$$

e la funzione:

$$\llbracket B \rrbracket : State \rightarrow \{true, false\}$$

possiamo definire:

$$\llbracket if\ B\ then\ C\ else\ C' \rrbracket = \lambda s \in State. if\ (\llbracket B \rrbracket(s), \llbracket C \rrbracket(s), \llbracket C' \rrbracket(s))$$

dove:

$$if(b, x, x') = \begin{cases} x & \text{if } b = true \\ x' & \text{if } b = false \end{cases}$$

Esempio:

La denotazione di una sequenza di composizioni $C; C'$ di due comandi:

$$\llbracket C; C' \rrbracket = \llbracket C' \rrbracket \circ \llbracket C \rrbracket = \lambda s \in State. \llbracket C' \rrbracket(\llbracket C \rrbracket(s))$$

È dato dalla composizione di funzioni parziali da stato a stato $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$ i quali rappresentano le denotazioni dei comandi.

Le semantiche operative di sequenze di composizioni si rappresentano così:

$$\frac{C, s \Downarrow s' \quad C', s' \Downarrow s''}{C; C', s \Downarrow s''}$$

Definiamo ora alcune strutture algebriche che sono alla base della teoria dei linguaggi e che giustificheranno il nostro interesse per i numeri naturali in questo contesto.

3. Strutture Algebriche e d'Ordine

3.1 Monoidi

Definizione 1:

Un monoide M è un semigrupp (= insieme in cui è definita una operazione binaria associativa) che sia dotato di elemento neutro, ovvero, indicato il monoide con $(M, +, u)$, deve risultare:

$$M1) (x + y) + z = x + (y + z) \quad \forall x, y, z \in M$$

$$M2) x + u = u + z = x \quad \forall x \in M$$

Esempio: $(\mathbb{N}, +, 0, \leq)$ è un monoide. Inoltre è ordinato.

Esempio: $(\mathbb{Z}_n, +, [0])$ l'insieme delle classi resto modulo n è un esempio di monoide non ordinato.

Infatti abbiamo la relazione (dove la somma si estende a n classi):

$$[1] + [1] + \dots + [1] = [0]$$

Questa relazione esprime una “condizione”, che, in particolare, impedisce l'ordinamento. Noi siamo invece interessati a trattare *strutture libere*.

Definizione 2:

Sia $G \subseteq M$. Diremo che G è un insieme di generatori se ogni elemento di M si può scrivere come combinazione finita di elementi di G .

Osservazione:

Ogni monoide ammette sempre un insieme di generatori, ad esempio quello banale, cioè $G = M$; ovviamente noi saremo interessati all'insieme di generatori minimo.

Esempio:

$(\mathbb{N}, +, 0, \leq)$ ammette come insieme di generatori $\{1\}$. Infatti ogni elemento $n \in \mathbb{N}$ si può esprimere come $\sum_i 1$ (l'elemento 0 si ottiene prendendo l'indice $i = 0$ nella somma).

Definizione 3:

Un monoide L , generato da G si dice libero su G e si indica con $L(G)$ o G^* , se per ogni monoide M e per ogni $\bar{f} : G \rightarrow M$, esiste un unico morfismo $f : L \rightarrow M$ tale che $f|_G = \bar{f}$.

Questa definizione si può esprimere dicendo che esiste un unico morfismo tra L ed M che rende commutativo il seguente diagramma:

$$\begin{array}{ccc} G & \xrightarrow{i} & L \\ \searrow \bar{f} & & \swarrow f \\ & M & \end{array}$$

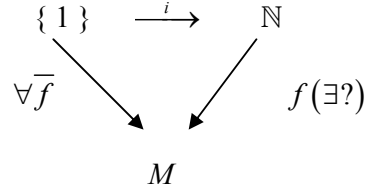
ovvero che $\overline{f}(g) = f(i(g))$, $\forall g \in G$, dove $i : G \subseteq L \rightarrow L$ è l'inclusione.

Esempio: Si noti l'analogia con la proprietà che uno spazio vettoriale è libero sulla sua base.

Esempio: \mathbb{N} è il monoide libero generato da $\{1\}$.

Infatti abbiamo già osservato che \mathbb{N} è un monoide ed è generato da $\{1\}$. Vediamo ora che è libero:

\forall monoide (M, \bullet, u) e $\forall \overline{f} : \{1\} \rightarrow M$, vogliamo trovare un morfismo $f : \mathbb{N} \rightarrow M$ tale che $f|_{\{1\}} = \overline{f}$.



Affinché $f : \mathbb{N} \rightarrow M$ sia un morfismo, l'unica scrittura possibile è:

$$\begin{aligned}
 f(n) &= f(1 + \dots + 1) = f(1) \bullet \dots \bullet f(1) \\
 f(1) &= u
 \end{aligned}$$

infatti deve essere $f(1) = \overline{f}(1)$ affinché il diagramma sia commutativo.

Proposizione 3:

\mathbb{N} è, a meno di isomorfismi, l'unico monoide libero generato da un elemento.

Dimostrazione:

Sia $(\mathbb{N}', +, 1')$ un altro monoide libero generato dall'elemento $\{1'\}$.

Poiché \mathbb{N} è libero, preso $M = \mathbb{N}'$, abbiamo visto che esiste un solo omomorfismo $\alpha :$

$$\begin{aligned}
 \alpha : \mathbb{N} &\xrightarrow{!} \mathbb{N}' \\
 1 &\mapsto 1'
 \end{aligned}$$

Analogamente, essendo \mathbb{N}' libero, esiste un solo omomorfismo $\alpha' :$

$$\begin{aligned}
 \alpha' : \mathbb{N}' &\xrightarrow{!} \mathbb{N} \\
 1' &\mapsto 1
 \end{aligned}$$

La composizione di due omomorfismi è ancora un omomorfismo ed è l'unico.

$$\begin{aligned}
 \alpha\alpha' : \mathbb{N} &\rightarrow \mathbb{N}' \rightarrow \mathbb{N} & \alpha'\alpha : \mathbb{N}' &\rightarrow \mathbb{N} \rightarrow \mathbb{N}' \\
 1 &\mapsto 1' \mapsto 1 & 1' &\mapsto 1 \mapsto 1'
 \end{aligned}$$

e risulta:

$$\alpha\alpha' = id_{\mathbb{N}}$$

$$\alpha'\alpha = id_{\mathbb{N}}$$

dunque α è un isomorfismo ed \mathbb{N} è l'unico monoide libero generato da un elemento a meno di isomorfismi: $\mathbb{N} \simeq \mathbb{N}'$. ■

Vogliamo ora descrivere un monoide libero generato da un insieme finito (o infinito) di generatori.

Proposizione 4:

Sia A un insieme finito di generatori. Allora il monoide libero generato da A è:

$$L(A) = A^* = (\{a_1 a_2 \dots a_n \mid a_i \in A, \forall i = 1, \dots, n\}, \bullet, \varepsilon)$$

dove un elemento del tipo $a_1 a_2 \dots a_n$, detto “parola”, si ottiene componendo gli elementi di A con l'operazione associativa \bullet , omettendo le parentesi.

La parola priva di lettere ε prende il nome di parola vuota.

Dimostrazione:

$L(A)$ è generato da A perché ogni elemento di $L(A)$ si può scrivere come combinazione finita di elementi di A .

$L(A)$ è un monoide. Infatti possiamo definire una operazione tra queste parole (sequenze finite di elementi di A) mettendole in sequenza:

$$\forall a_1 \dots a_n, b_1 \dots b_m \in A, \quad (a_1 \dots a_n) \bullet (b_1 \dots b_m) \stackrel{def}{=} a_1 \dots a_n b_1 \dots b_m$$

tale operazione è associativa ed ε è l'elemento neutro.

$L(A)$ è libero. Infatti $A \subseteq L(A)$ perché ogni elemento di A è una parola di lunghezza 1, cioè formata da un solo elemento, dunque possiamo considerare l'inclusione $i: A \subseteq L(A) \rightarrow L(A)$. Vogliamo che per ogni monoide (M, \bullet, u) e per ogni $\bar{f}: A \rightarrow M$, sia commutativo il seguente diagramma:

$$\begin{array}{ccc} A & \xrightarrow{i} & L(A) \\ \searrow \bar{f} & & \swarrow !f \\ & M & \end{array}$$

Affinché f sia un omomorfismo, deve essere:

$$\begin{aligned} f(a_1 \dots a_n) &= f(a_1) \bullet \dots \bullet f(a_n) \\ f(\varepsilon) &= u \end{aligned}$$

ed affinché il diagramma sia commutativo deve essere:

$$f(a_1 \dots a_n) = f(a_1) \bullet \dots \bullet f(a_n) = \bar{f}(a_1) \bullet \dots \bullet \bar{f}(a_n)$$

dunque rimane univocamente determinato l'omomorfismo f . ■

Esempio:

Sia A^* il monoide libero generato da A . Possiamo considerare una funzione:

$$\begin{aligned}\bar{l}: A &\rightarrow \mathbb{N} \\ a_i &\mapsto 1\end{aligned}$$

Sappiamo che questa individua un unico omomorfismo l :

$$\begin{aligned}l: L(A) &\xrightarrow{!} \mathbb{N} \\ a_1 \dots a_n &\mapsto l(a_1) + \dots + l(a_n) = n\end{aligned}$$

l prende il nome di *lunghezza* in quanto esprime la "lunghezza" delle parole.

Osservazione 1:

Se si considera un monoide non libero, in generale la lunghezza non è ben definita.

Esempio:

Un monoide non libero è un monoide in cui sono definite delle equazioni.

Se consideriamo una equazione del tipo $a_1 a_2 = a_3$, non siamo in grado di definire univocamente la lunghezza (perché non sappiamo se la parola $a_1 a_2$ ha lunghezza 2 oppure 1).

Osservazione 2:

In generale i monoidi liberi non sono commutativi. L'unico monoide libero commutativo è \mathbb{N} in quanto è generato da un solo elemento.

Esempio:

Consideriamo il monoide A^* generato da $A = \{a_1, a_2\}$. La commutatività dovrebbe essere espressa da una condizione del tipo $a_1 a_2 = a_2 a_1$ e quindi il monoide non sarebbe più libero.

In generale una equazione induce una relazione di equivalenza, dunque un monoide commutativo è ottenibile come quoziente di un monoide libero.

3.2 Insiemi Ordinati

Adesso vediamo una serie di definizioni su insiemi, in particolare su insiemi ordinati e parzialmente ordinati, su insiemi parzialmente ordinati continui, sui reticoli, reticoli distributivi e reticoli completi.

Definizione 4:

Un insieme X si dice *parzialmente ordinato* se è definita in esso una relazione binaria $\mathbb{R} \subseteq X \times X$ che soddisfa la proprietà riflessiva, antisimmetrica e transitiva. Indicheremo l'insieme ordinato (parzialmente) con (X, \leq) .

Definizione 5:

Preso una coppia di elementi (a, b) in un insieme parzialmente ordinato (X, \leq) , che chiameremo *massimo comune minorante*, in simboli $a \cap b$, l'elemento tale che:

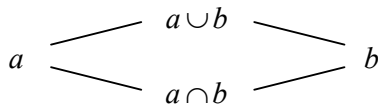
- $a \cap b \leq a, a \cap b \leq b$;
- per ogni $c \in X$ tale che $c \leq a, c \leq b$ risulta $c \leq a \cap b$.

Definizione 6:

Nelle medesime condizioni è possibile definire dualmente il *minimo comune maggiorante* come quell'elemento, in simboli $a \cup b$, tale che:

- $a \cup b \geq a, a \cup b \geq b$;
- per ogni $c \in X$ tale che $c \geq a, c \geq b$ risulta $c \geq a \cup b$.

Attraverso l'utilizzo dei diagrammi di Hasse le ultime due definizioni possono essere rappresentate nel seguente modo:



Esempio:

Consideriamo l'insieme dei \mathbb{N} con la relazione di divisibilità che risulta d'ordine parziale: il massimo comune minorante coincide con il Massimo Comune Divisore ed il minimo comune maggiorante con il minimo comune multiplo.

Definizione 7:

Un *reticolo* è un insieme parzialmente ordinato (X, \leq) nel quale, per ogni coppia di elementi (a, b) esiste un massimo comune minorante $a \cap b$ ed un minimo comune maggiorante $a \cup b$.

Esempio:

Considerato un insieme X , l'insieme delle parti di X , $P(X)$, risulta essere un reticolo.

Proposizione 5:

Dato un reticolo (X, \leq, \cap, \cup) , \cap e \cup sono due operazioni che godono delle seguenti proprietà:

- 1) $a \cap a = a;$ $a \cup a = a;$

- | | |
|---|--|
| 2) $a \cap b = b \cap a;$ | $a \cup b = b \cup a;$ |
| 3) $(a \cap b) \cap c = a \cap (b \cap c);$ | $(a \cup b) \cup c = a \cup (b \cup c);$ |
| 4) $a \cap (a \cup b) = a;$ | $a \cup (a \cap b) = a;$ |

Si può osservare che se (X, \cap, \cup) è un insieme con due operazioni che godono delle proprietà precedenti, definendo su X la relazione d'ordine:

$$a \leq b \text{ se e soltanto se } a \cap b = a \text{ e } a \cup b = b$$

allora \cap rappresenta il massimo comune minorante e \cup il minimo comune maggiorante e conseguentemente si ottiene un reticolo.

Definizione 8:

Affermeremo che un reticolo X è distributivo se $\forall a, b \in X$ sono verificate le seguenti uguaglianze:

- 1) $a \cup (b \cap c) = (a \cup b) \cap (a \cup c)$
- 2) $a \cap (b \cup c) = (a \cap b) \cup (a \cap c).$

Definizione 9:

Dato un reticolo con massimo 1 e minimo 0, chiameremo complemento di a e lo indicheremo con a' l'elemento che verifica le seguenti proprietà:

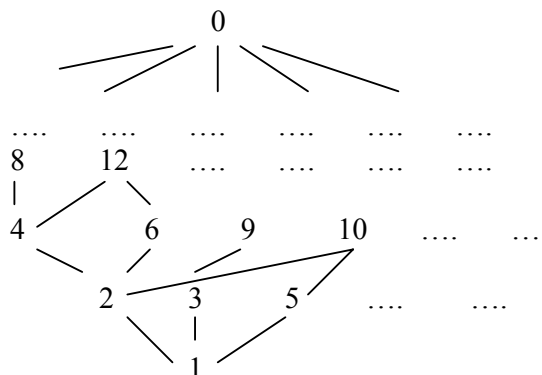
- 1) $a \cap a' = 0$
- 2) $a \cup a' = 1.$

Definizione 10:

Un reticolo si dice **COMPLETO** se contiene il minimo comune maggiorante per ogni sottoinsieme scelto.

Esempio:

L'insieme dei numeri Naturali \mathbb{N} , oppure l'insieme dei numeri Reali positivi \mathbb{R}^+ . Vediamo invece un reticolo così costruito:



Questo reticolo è un caso particolare in cui lo 0 è in cima all'albero in quanto è divisibile per tutti gli elementi sotto a lui. Ma questo reticolo NON E' COMPLETO, in quanto per la definizione si può trovare un sottoinsieme che non contiene il minimo comune maggiorante.

Ma a noi non serve obbligatoriamente che un reticolo sia completo, basta che esista un minimo comune maggiorante per ogni sottoinsieme totalmente ordinato, cioè per ogni CATENA, all'interno di un sottoinsieme parzialmente ordinato cioè :

\exists un insieme parzialmente ordinato continuo (c.p.o.) se \exists un minimo comune maggiorante per ogni catena.

3.3 Insiemi Parzialmente Ordinati

Definizione 11:

Un insieme parzialmente ordinato (D, \leq) è un insieme D su cui è definita una relazione binaria \leq , detta relazione d'ordine, tale che $\forall x, y, z \in D$ si ha:

- 1) $x \leq x$ (proprietà riflessiva)
- 2) Se $x \leq y$ e $y \leq z \Rightarrow x \leq z$ (proprietà transitiva)
- 3) Se $x \leq y$ e $y \leq x \Rightarrow x = y$ (proprietà antisimmetrica)

Quindi la coppia (D, \sqsubseteq) è chiamata insieme parzialmente ordinato (poset).

Definizione 12:

Supponiamo D un insieme parzialmente ordinato ed S un sottoinsieme di D . Un elemento $d \in S$ è il più piccolo elemento di S se esso soddisfa che :

$$\forall x \in S . d \sqsubseteq x$$

Definizione 13:

Un insieme parzialmente ordinato (D, \leq) è un insieme parzialmente ordinato continuo (c.p.o.) se :

- 1) Ha un minimo che indichiamo con il simbolo \perp
- 2) Ha un minimo confine superiore per tutte le catene (sottoinsiemi totalmente ordinati) che indicheremo con il simbolo \vee (a volte chiamato sup).

Esempio:

Nell'insieme $P(A^*)$ queste condizioni sono verificate, infatti:

- 1) \emptyset rappresenta il minimo
- 2) Presa una catena di sottoinsiemi $A_1 \subseteq A_2 \subseteq A_3 \subseteq \dots$, ha un minimo confine superiore, rappresentato dall'unione di questi insiemi, che indicheremo con $\bigvee_{i \in \mathbb{N}} A_i$

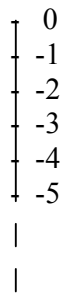
Se la catena dovesse essere vuota avremo che :

$$C = \emptyset$$

$$\begin{array}{lll} \text{Dato } x & \forall y \in C & x \geq y \\ \forall z \text{ tale che} & \forall y \in C & z \geq y \Rightarrow x \leq z \end{array}$$

Esempio:

L'insieme dei numeri minori di zero:



Qui non esiste il minimo comune maggiorante.

Definizione 14:

Siano (A, \leq) e (B, \leq') due insiemi parzialmente ordinati. Una funzione $f : (A, \leq) \Rightarrow (B, \leq')$ è monotona se conserva l'ordine, in pratica se $x \leq y \Rightarrow f(x) \leq' f(y)$.

Definizione 15:

Siano (D, \leq) e (D', \leq') due insiemi parzialmente ordinati continui (c.p.o). Una funzione $f : D \rightarrow D'$ è continua se per ogni catena $a_1 \leq a_2 \leq a_3 \leq \dots$ in D , la successione $\{f(a_n) | n \geq 0\}$ in D' ha un minimo confine superiore che rispetta f :

$$\bigvee_{i \in I} f(a_n) = f\left(\bigvee_{i \in I} a_n\right)$$

Definizione 16: Omomorfismo

Siano (D, \leq) e (D', \leq') due insiemi parzialmente ordinati continui (c.p.o). Una funzione $f : D \rightarrow D'$ è un omomorfismo se:

- 1) È monotona
- 2) È continua.

Lemma 1:

Ogni applicazione continua è monotona.

Dimostrazione:

Data la funzione continua $f : D \rightarrow D'$ e $a \leq b$ in D noi dobbiamo provare che $f(a) \leq' f(b)$ in D' . Consideriamo allora la catena ascendente $a \leq b \leq b \leq b \leq \dots$, essa ha chiaramente come minimo confine superiore b , ossia $\vee \{a, b\} = b$.

Dunque, per definizione di continuità $\vee \{f(a), f(b)\} = f(\vee \{a, b\}) = f(b)$ in D' , cioè $f(a) \leq' f(b)$. ■

Lemma 2: Intreccio di catene

Sia (D, \leq) un c.p.o. e siano $x_1 \leq x_2 \leq x_3 \leq \dots$ e $y_1 \leq y_2 \leq y_3 \leq \dots$ due catene di D tali che :

1. $\forall x_i \quad \exists y_j \mid x_i \leq y_j$
2. $\forall y_i \quad \exists x_j \mid y_i \leq x_j$

Allora:

$$\vee x_i = \vee y_j.$$

Dimostrazione:

Proveremo che $\vee y_j \leq \vee x_i$ e che $\vee x_i \leq \vee y_j$ e conseguentemente saranno uguali.

Preso un arbitrario y_j , dalle ipotesi segue che esiste un indice k tale che $y_j \leq x_k$; poiché $\vee x_i$ è il minimo confine superiore degli x_i , allora $y_j \leq x_k \leq \vee x_i$. Essendo la precedente disuguaglianza vera per ogni indice j , segue che $\vee y_j \leq \vee x_i$. Allo stesso modo si può provare che $\vee x_i \leq \vee y_j$, e quindi $\vee x_i = \vee y_j$. ■

Definizione 17: Punto Fisso di Kleene

Sia (D, \leq) un insieme parzialmente ordinato continuo (c.p.o) e sia f una funzione continua tale che $f : D \rightarrow D$. Allora esiste il (unico) minimo punto fisso, che è dato da $\bigvee_{k \in \mathbb{N}} f^k(\perp)$, dove il simbolo \perp identifica il minimo elemento di (D, \leq) e f^k indica la composizione di f con se stesso k volte. ■

Riassumendo abbiamo che:

Definizione di Punto Fisso: $\exists \bar{x}$ tale che $f(\bar{x}) = \bar{x}$

Definizione di Minimo Punto Fisso: $\forall y$ tale che $f(y) = y \Rightarrow \bar{x} \leq y$

Esempio:

Nell'insieme dei numeri naturali \mathbb{N} con la funzione successivo, non esiste il punto fisso perché c'è sempre un successore.

Dimostrazione costruttiva (del Punto Fisso di Kleene) :

Partiamo prendendo il minimo rappresentato dal simbolo \perp e costruiamo $f(\perp)$. Tra \perp e $f(\perp)$, dal momento che \perp è il minimo, esiste la seguente relazione:

$$\perp \leq f(\perp).$$

Se $\perp = f(\perp)$, ci fermiamo poiché abbiamo trovato un punto fisso.

Se $\perp \neq f(\perp)$, riproviamo costruendo f^2 , e applicando f alla disuguaglianza $\perp \leq f(\perp)$ otteniamo:

$$f(\perp) \leq f^2(\perp).$$

Notiamo che il passaggio precedente è lecito, poiché f è continua e monotona e che $\perp \leq f(\perp)$.

Se $f(\perp) = f^2(\perp)$, ci fermiamo, altrimenti iterando il procedimento abbiamo:

$$f^2(\perp) \leq f^3(\perp)$$

Ottenendo così la catena di disuguaglianze:

$$\perp \leq f(\perp) \leq f^2(\perp) \leq f^3(\perp) \leq \dots \quad (1)$$

La (1) potrebbe essere una catena infinita, oppure in un certo punto potrebbe arrestarsi, avendo così individuato un punto fisso.

Nel caso di una catena infinita, essendo in ogni caso totalmente ordinata in un c.p.o., ha un minimo confine superiore (sup).

Diciamo che il minimo confine superiore di (1), ossia $\bigvee_{k \in \mathbb{N}} f^k(\perp)$, è il punto fisso cercato, anzi è proprio il minimo punto fisso.

Proviamo prima di tutto che è un punto fisso.

Applicando la f a $\bigvee_{k \in \mathbb{N}} f^k(\perp)$ otteniamo:

$$f\left(\bigvee_{k \in \mathbb{N}} f^k(\perp)\right) = \bigvee_{k \in \mathbb{N}} f^{k+1}(\perp).$$

Vediamo ora di quale catena $\bigvee_{k \in \mathbb{N}} f^{k+1}(\perp)$ è il minimo confine superiore.

Per $k=0$, $\bigvee_{k \in \mathbb{N}} f^{k+1}(\perp) = f(\perp)$, per $k=1$, $\bigvee_{k \in \mathbb{N}} f^{k+1}(\perp) = f^2(\perp)$, ed iterando il procedimento otteniamo la seguente catena:

$$f(\perp) \leq f^2(\perp) \leq f^3(\perp) \leq f^4(\perp) \leq \dots \quad (2)$$

Possiamo notare che la catena (2) risulta sfasata di uno rispetto alla catena (1), quindi differiscono l'una dall'altra a causa del primo elemento. Tuttavia il minimo confine superiore di (2) coincide con il minimo confine superiore di (1) dal momento che vale il lemma 2 dell'intreccio di catene.

Infatti per ogni elemento di (1) c'è un elemento di (2) che lo supera e viceversa.

Quindi abbiamo dimostrato che:

$$\bigvee_{k \in \mathbb{N}} f^k(\perp) = \bigvee_{k \in \mathbb{N}} f^{k+1}(\perp) = f\left(\bigvee_{k \in \mathbb{N}} f^k(\perp)\right).$$

è un punto fisso.

Proviamo ora che è il minimo punto fisso, quindi dato comunque un altro punto fisso, questo risulterà maggiore o uguale.

Supponiamo che esista un altro punto fisso \bar{x} e consideriamo la catena:

$$\bar{x} \leq f(\bar{x}) \leq f^2(\bar{x}) \leq f^3(\bar{x}) \leq \dots \quad (3)$$

questa risulta essere banale, dal momento che \bar{x} è un punto fisso e perciò $\bar{x} = f(\bar{x})$, $\bar{x} = f^2(\bar{x})$, $\bar{x} = f^3(\bar{x})$,

Vediamo ora in quale relazione sono gli elementi della catena (3) con gli elementi della catena (1):

- $\perp \leq \bar{x}$ poiché \perp è il minimo
- $f(\perp) \leq f(\bar{x})$ poiché abbiamo applicato la f monotona
- $f^2(\perp) \leq f^2(\bar{x})$ e così via

La catena (3) è una maggiorante della catena (1), quindi ogni elemento di (3) è maggiore di ogni elemento di (1), e così anche il minimo confine superiore di (3) sarà maggiore o uguale al minimo confine superiore di (1).

Notiamo che il minimo confine superiore di (3) è \bar{x} , dal momento che la catena (3) è costante, che sarà maggiore o uguale di ogni elemento di (1) e quindi anche del suo minimo confine superiore, che è il punto fisso di (1), cioè:

$$\bar{x} \geq \bigvee_{k \in \mathbb{N}} f^k[\perp].$$

Da tutto ciò discende che $\bigvee_{k \in \mathbb{N}} f^k(\perp)$ è il minimo punto fisso. ■

3.4 Algebra dei linguaggi

Da ora in poi studieremo il monoide libero A^* generato da un insieme finito di generatori A . Questa struttura è alla base della teoria dei linguaggi, che, nata inizialmente dallo studio dei linguaggi naturali, si è sviluppata sui linguaggi artificiali e sulla definizione delle regole che li caratterizzano. Infatti non saremo interessati a tutte le parole, ma solo ad alcune successioni.

Le successioni per noi “buone” potrebbero essere scelte tra quelle

Sintatticamente corrette

Un altro criterio di selezione potrebbe essere invece la richiesta che le parole siano

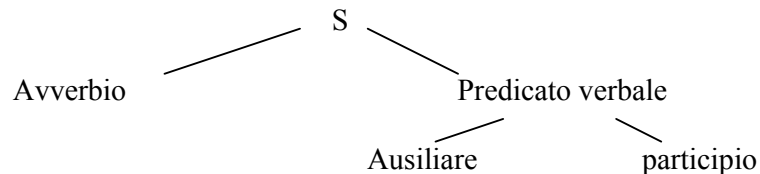
Semanticamente significanti

Esempio:

Consideriamo la frase:

“Domani sono stato piovuto”

Indicando tale affermazione con la lettera S (Sentence), possiamo schematizzarla nel seguente modo:



Nonostante sia strutturalmente corretta, possiamo evidenziare degli errori nella sintassi: l'ausiliare è un participio passato del verbo essere ed è passivo, mentre piovere è un verbo intransitivo e richiede una forma attiva, l'avverbio domani è un futuro che richiede quindi la forma futura dell'ausiliare.

Invece l'affermazione:

“Ieri ho mangiato”

nonostante abbia la stessa struttura della frase precedente, è sintatticamente corretta.

Esempio:

Se consideriamo la frase:

“Per sarà il certamente”

non riusciamo a trovare una struttura sintattica lecita in italiano.

La frase:

“il *certamente* è un avverbio”

è invece sintatticamente corretta perchè abbiamo sostantivato l'avverbio, che gioca quindi un ruolo diverso da quello abituale.

In generale è piuttosto difficile stabilire se una frase è semanticamente corretta, in quanto il significato delle parole può cambiare a seconda del ruolo che si attribuisce loro. Basti pensare ai giochi linguistici in cui si avventurano i poeti.

Il nostro interesse è pertanto rivolto alla sintassi, ovvero alla struttura delle frasi.

Il punto di partenza sarà la ricerca di linguaggi descritti da una grammatica che fornisce le regole per costruirli.

Definizione 18:

Sia A un insieme non vuoto ma finito di simboli, che chiamiamo “alfabeto”, con A^* indichiamo l'insieme di tutte le stringhe di lunghezza finita su A . In questo insieme infinito di stringhe è contenuta anche la stringa vuota che indichiamo con ε .

Esempio:

Sia $A = \{(,)\}$. Allora A^* è l'insieme di tutte le stringhe finite di parentesi.

Definizione 19:

Un “linguaggio” L su A è un sottoinsieme di A^* , ossia $L \in P(A^*)$.

Ricordiamo che in $P(A^*)$ sono definite le seguenti operazioni : Intersezione \cap , unione \cup e complemento $(\cdot)'$. Inoltre $(P(A^*), \subseteq)$ è un insieme parzialmente ordinato in cui 0 è il minimo e A^* è il massimo. Ricordiamo cos'è il complemento: $L^c \cup L = A^*$, $L^c \cap L = \emptyset$. Quindi osserviamo che $(P(A^*), \cap, \cup, (\cdot)')$ è un'algebra di Boole, ossia è un reticolo distributivo complementato.

Possiamo introdurre il “Prodotto di Frobenius” definendolo nel seguente modo:

$$\forall L, M \in P(A^*) \quad L \cdot M \stackrel{\text{def}}{=} \{ww' \mid \text{tali che } w \in L, w' \in M\}$$

Tale prodotto gode delle seguenti proprietà:

- Associatività: infatti fare il prodotto $L \cdot M \cdot N$ significa unire terne di parole;
- Elemento neutro $\{\varepsilon\}$ tale che $\{\varepsilon\} \cdot L = L \cdot \{\varepsilon\} = L$;
- 0 è annullatore, ovvero $0 \cdot L = 0$ e $L \cdot 0 = 0 \quad \forall L \in P(A^*)$;
- Monotonia: $L \subseteq L'$ e $M \subseteq M' \Rightarrow L \cdot M \subseteq L' \cdot M'$.

Osserviamo che tale prodotto non è commutativo.

Possiamo chiederci se, considerando \cup come somma e \cdot come prodotto, $P(A^*)$ risulta essere un anello. Infatti la somma \cup è associativa, commutativa ed esiste l'elemento neutro 0 .

Sono verificate le leggi distributive, ossia $\forall L, M, N \in P(A^*)$ risulta:

$$L \cdot (M \cup N) = (L \cdot M) \cup (L \cdot N)$$

dove $L \cdot (M \cup N)$ per definizione di unione è la concatenazione di una parola di L con una parola di M oppure con una di N . Dunque è uguale a $(L \cdot M) \cup (L \cdot N)$ che è una parola di $L \cdot M$ oppure di $L \cdot N$. Analogamente si dimostra che $(M \cup N) \cdot L = (M \cdot L) \cup (N \cdot L)$.

Ciò che manca a questa struttura per essere un anello è l'esistenza dell'opposto rispetto alla somma che nel nostro caso è l'operazione \cup : $\forall L \in P(A^*) \setminus \{0\}$ non esiste alcun elemento di $P(A^*)$ che sommato ad L dia come risultato 0 .

In conclusione $(P(A^*), \cup, \cdot, 0, \{\varepsilon\})$ è un semianello ordinato.

Da tutto ciò segue che, se A è un alfabeto, $P(A^*)$ ha una struttura algebrica ricchissima:

$$(P(A^*), \subseteq, \cup, \cap, 0, A^*, \cdot, \{\varepsilon\})$$

Definiamo ora il concetto di DERIVATA di un insieme :

$$[L \setminus M] = \{w \mid L \cdot \{w\} \subseteq M\}$$

In altre parole la derivata sinistra di L rispetto ad M è uguale ad un elemento w tale che $L \cdot \{w\}$ è un sottoinsieme di M .

Esempio:

Preso un insieme

$$M = \{ab, abc, b, bb\}$$

Allora

$$[L \setminus M] = \{b, bc\}$$

In pratica la derivata sinistra di L rispetto ad M è uguale al più grande sottoinsieme di M che contiene M .

Esempio:

Preso un insieme:

$$\begin{aligned} M &= \{ab, abc, b, bb\} \\ L &= \{a, b\} \end{aligned}$$

Allora:

$$[L \setminus M] = \{b\}$$

Questo siccome: $L \cdot [L \setminus M] = \{a, b\} \cdot \{b\} = \{ab, bb\} \subseteq M$

Proposizione 6:

$$L \cdot N \subseteq M \Leftrightarrow N \subseteq [L \setminus M]$$

Ciò per la premessa che $[L \setminus M] =$ al più grande insieme tale che $L \cdot [L \setminus M] \subseteq M$.

Un'analogia della precedente proposizione è data dal:

Teorema 1: Teorema Deduttivo

$$\frac{A \wedge B \vdash C}{B \vdash A \rightarrow C}$$

Cioè A e B deduce C se e soltanto se B deduce A implica C .

Questo teorema permette di dimostrare che in una struttura esiste il complemento, mentre per la negazione è molto più difficile.

Definizione 20: Iterazione

Preso un insieme L , allora l'insieme L^* è l'iterazione di L , cioè sono tutte le concatenazioni finite di L .

Dimostrazione:

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^{n+1} &= L \cdot L^n \\ L^* &= \bigcup_{n \in \mathbb{N}} L^n \end{aligned}$$

Esempio:

$$\begin{aligned} L &= \{a\} \\ L^* &= \{\varepsilon, a, aa, aaa, aaaa, \dots\} = \{a^n\}_{n \in \mathbb{N}} \end{aligned}$$

Possiamo notare la somiglianza a strutture algebriche di matematica.
La grammatica dell'esempio precedente è:

$$X = aX \mid \varepsilon \quad \text{oppure} \quad X = aX + \varepsilon$$

e questo genera il linguaggio di L^* .

$$X = aX + \varepsilon \Rightarrow X = \{a\} \cdot X \cup \{\varepsilon\} \in P(A^*)$$

Si può notare che la stessa equazione $x = ax + 1$ nell'insieme dei Razionali ammette come soluzione $x = \frac{1}{1-a}$ dove sono state utilizzate le operazioni di differenza e di divisione. Tuttavia, per $0 \leq a < 1$, abbiamo che lo sviluppo in serie di Taylor ci dà: $x = 1 + a + a^2 + a^3 + \dots$

Trovare la soluzione dell'espressione $x = ax + \varepsilon$ è più complesso:

Dato che: $\{a^n\}_{n \in \mathbb{N}} = \{a^{n+1}\}_{n \in \mathbb{N}} \cup \{\varepsilon\}$ riesco a capire che L^* è soluzione, ma effettivamente qual è?

Provo per approssimazione:

- 1) Per $x = \emptyset$
 $\emptyset \stackrel{?}{=} a \cdot \emptyset + 1$ NO
- 2) Per $x = 1$
 $1 \stackrel{?}{=} a \cdot 1 + 1$ cioè $\varepsilon = a \cdot \varepsilon + \varepsilon$ NO
- 3) Per $x = a + 1$
 $a + 1 \stackrel{?}{=} a \cdot (a + 1) + 1$ NO
- 4)
- 5)

Andando all'infinito le differenze tra i due termini dell'equazione si assottigliano e all'infinito sono considerate UGUALI, cioè esattamente lo sviluppo in serie di Taylor visto prima. Questo oggetto lo chiameremo A^* .

Esempio:

Vediamone ora uno più complesso:

$$\begin{cases} X = aY + 1 \\ Y = bX + 1 \end{cases} \varepsilon$$

Per prima prendo la variabile X come variabile iniziale e provo a risolvere l'equazione, sostituendo Y:

$$\begin{cases} X = aY + 1 \\ Y = bX + 1 \end{cases} \Rightarrow X = a(bX + 1) + 1 = abX + a + 1$$

Per sostituzione avrò:

$$\begin{aligned} X_0 &= 0 \\ X_1 &= a + 1 && \text{perché ho sostituito 0 alla variabile X} \\ X_2 &= aba + ab + a + 1 && \text{perché ho sostituito } a+1 \text{ alla variabile X} \\ X_3 &= ababa + abab + aba + ab + a + 1 && \text{perché ho sostituito } aba + ab + a + 1 \text{ alla variabile X} \\ X_4 &= \dots\dots\dots \end{aligned}$$

quindi la soluzione sarà:

$$X = (ab)^* \cdot (a + 1)$$

Allo stesso modo avrei potuto risolvere per la variabile Y:

$$\begin{cases} X = aY + 1 \\ Y = bX + 1 \end{cases} \Rightarrow Y = b(aX + 1) + 1 = baY + b + 1$$

e quindi la soluzione sarà:

$$Y = b((ab)^* \cdot (a + 1)) + 1$$

Esempio:

Prendiamo una funzione del tipo:

$$x = ax + b$$

(ma potrebbe essere anche del tipo $x = (a + a')x + b$ oppure $x = (a + a')x + (b + b')$, dato che in ogni caso possiamo sempre ricondurci alla prima equazione poiché nell'ambiente in cui operiamo vale la proprietà distributiva e si può raccogliere a fattor comune), tale che:

$$P(A*) \xrightarrow{f=a \cdot _+ + b} P(A*).$$

Bisogna dimostrare che questa funzione sia continua, così da poter applicare il teorema del punto fisso e trovare il minimo punto fisso.

Intanto verifico che sia monotona :

prendo un Y tale che:

$$X \subseteq Y \Rightarrow aX + b \subseteq aY + b,$$

quindi è monotona.

Verifichiamo che sia continua:

$$X_1 \subseteq X_2 \subseteq X_3 \subseteq X_4 \subseteq \dots \subseteq \bigcup_i X_i$$

Cioè:

$$(4) \quad \bigcup_i aX_i + b \stackrel{?}{=} a \bigcup_i X_i + b \quad (5)$$

In (4) avremo aw_i oppure b , in (5) avremo $a \cdot w_i$ oppure b , quindi sono uguali, allora la nostra funzione è monotona e continua.

Applichiamo tutto ciò :

$$x = ax + b$$

$$x_0 = 0$$

$$x_1 = ax_0 + b \Rightarrow x = b$$

$$x_2 = ax_1 + b \Rightarrow x = ab + b$$

$$x_3 = ax_2 + b \Rightarrow x = a(ab + b) + b = a^2b + ab + b$$

.....

.....

.....

Quindi iterando il procedimento avremo che la minima soluzione di questa equazione, cioè il minimo punto fisso che sarà:

$$x = a^*b$$

Riassumendo:

$$\emptyset \subseteq f(\emptyset) = \{b\} \subseteq f(f(\emptyset)) = \{ab + b\} \subseteq f(f(f(\emptyset))) = \{a(ab + b) + b\} = \{a^2b + ab + b\} \subseteq \dots \subseteq a^*b$$

Vediamo come sarebbe la soluzione dell'equazione da un punto di vista algebrico:

$$x = \frac{1}{1-a} b$$

Ma per $a < 1$ sappiamo che :

$$\frac{1}{1-a} = 1 + a + a^2 + a^3 + \dots$$

Ci sono quindi delle somiglianze formali nei due diversi modi di vedere l'equazione. ■

Esempio:

Prendiamo una funzione del tipo:

$$x = axb + ab$$

cioè in modo più esplicito:

$$x = a \cdot _ \cdot b + ab$$

Applichiamo il metodo per sostituzione visto in precedenza:

$$x_0 = 0$$

$$x_1 = a \cdot 0 \cdot b + ab = ab$$

$$x_2 = a \cdot x_1 \cdot b + ab = aabb + ab = a^2b^2 + ab$$

$$x_3 = ax_2b + ab = a^3b^3 + a^2b^2 + ab$$

$$x_4 = \dots\dots\dots$$

.....

Questo procedimento andrà avanti all'infinito, e ogni linguaggio ottenuto conterrà il precedente:

$$\emptyset \subseteq \{ab\} \subseteq \{a^2b^2, ab\} \subseteq \{a^3b^3, a^2b^2, ab\} \subseteq \dots\dots\dots \subseteq \{a^n b^n\}$$

Quindi $a^n b^n$ sarà il minimo punto fisso di questa catena.



Teorema 2:

Sia G una grammatica libera da contesto con funzione associata f . Allora il minimo punto fisso di f è il linguaggio libero da contesto $L(G)$ generato da G. ■

Se siamo in presenza di grammatiche lineari destre, abbiamo l'operatore di iterazione " $()^*$ " che ci permette di scrivere proprio la soluzione.

Infatti se abbiamo:

$$P(A^*)^n \rightarrow P(A^*)^n$$

$$(x_1 \dots x_n) \rightarrow f(x_1 \dots x_n) = (x'_1, \dots, x'_n)$$

Il minimo punto fisso sarà una ennupla.

Esempio:

$$\begin{cases} X = aX + bY + 1 \\ Y = cX + dY + 1 \end{cases}$$

La corrispondente grammatica regolare è:

$$\begin{aligned} X &\rightarrow aX \mid bY \mid \varepsilon \\ Y &\rightarrow cX \mid dY \mid \varepsilon \end{aligned}$$

Cerchiamo la soluzione, sempre per sostituzione :

$$1. \quad X_0 = 0; Y_0 = 0;$$

$$\begin{cases} X_1 = 1 = \varepsilon \\ Y_1 = 1 = \varepsilon \end{cases}$$

$$2. \quad X_1 = 1; Y_1 = 1;$$

$$\begin{cases} X_2 = a + b + 1 \\ Y_2 = c + d + 1 \end{cases}$$

$$3. \quad X_2 = a + b + 1; Y_2 = c + d + 1;$$

$$\begin{cases} X_2 = a(a + b + 1) + b(c + d + 1) + 1 \\ Y_2 = c(a + b + 1) + d(c + d + 1) + 1 \end{cases}$$

$$4. \quad \dots\dots\dots$$

$$5. \quad \dots\dots\dots$$

Questo metodo, grazie al precedente teorema, è applicabile anche per le grammatiche libere da contesto, in cui a sinistra di ogni produzione c'è un simbolo non terminale.

Esempio:

$$\begin{aligned} X &\rightarrow aXa \mid b \\ Y &\rightarrow Ya^2 \mid bXY \end{aligned} \quad \Rightarrow \quad \begin{aligned} X &= aXa + b \\ Y &= Ya^2 + bXY \end{aligned}$$

Anche in questo caso è possibile trovare il minimo punto fisso:

$$1. \quad X_0 = 0;$$

$$Y_0 = 0;$$

$$2. \quad X_1 = b;$$

$$Y_1 = 0;$$

3. $X_2 = aba + b;$
 $Y_2 = 0;$
 4. $X_3 = aabaa + aba + b;$
 $Y_3 = 0;$
 5.
 6.
- n. $X_n = a^n ba^n;$ \Rightarrow Minimo Punto Fisso
 $Y_n = 0;$
-

Esempio:

$$\begin{cases} X = aXYa + b \\ Y = Xb + 1 \end{cases}$$

1. $X_0 = 0;$
 $Y_0 = 0;$
2. $X_1 = b;$
 $Y_1 = 1;$
3. $X_2 = aba + b;$
 $Y_2 = bb + 1;$
4. $X_3 = a(aba + b) \cdot (bb + 1) \cdot a + b;$
 $Y_3 = (aba + b) \cdot b + 1;$
5.
6.

Nelle grammatiche lineari destre è sicuramente e facilmente risolvibile il minimo punto fisso, cioè la soluzione dell'equazione, mentre nelle grammatiche libere da contesto non sempre è possibile. Comunque il fatto che la grammatica sia libera da contesto ci consente di poter sostituire passo passo le varie approssimazioni.

Esempio:

$$\begin{cases} X = aX + bY + 1 \\ Y = cX + dY + 1 \end{cases}$$

Questa è un'espressione con due equazioni, ma può essere trattata allo stesso modo degli esempi precedenti, attraverso una serie di sostituzioni.

Vediamo come procedere:

$$Y = cX + dY + 1 \Rightarrow Y = dY + (cX + 1) \quad \text{per la proprietà associativa}$$

e la soluzione sarà:

$$Y = d * (cX + 1)$$

Ora vado a sostituire questa soluzione nella prima equazione:

$$X = aX + bd * (cX + 1) + 1$$

per la proprietà distributiva destra

$$X = aX + bd * cX + bd * 1 + 1$$

per la propr. distributiva sinistra e associatività

$$X = \underbrace{(a + bd * c)}_a X + \underbrace{(bd * 1 + 1)}_b$$

Mi sono riportato al caso di un esempio visto in precedenza in cui l'equazione era $x = ax + b$ e la soluzione o minimo punto fisso era $x = a * b$, così adesso avremo :

$$X = (a + bd * c) * (bd * 1 + 1)$$

$$Y = d * (c(a + bd * c) * (bd * 1 + 1) + 1)$$

che rappresenta appunto il minimo punto fisso.

Si dimostra la validità per induzione su tutte le variabili.

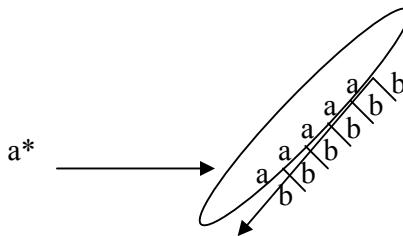
Riassumendo:

- Per grammatiche libere da contesto si utilizza il teorema del punto fisso, che per approssimazione ci farà trovare la soluzione.
- Se la grammatica è regolare possiamo addirittura trovare la soluzione per sostituzione.

Vediamo una rappresentazione grafica:

$$X = aX + b \quad \Rightarrow \quad X = aX \mid b$$

Abbiamo visto in precedenza che la soluzione è : $x = a * b$



3.5 Domini

Anche la semantica di un classico linguaggio di programmazione può essere interpretato come punto fisso di una funzione su una opportuna struttura ordinata.

Per esempio:

$$\underbrace{[while\ B\ do\ C]}_X = \left[\underbrace{if\ B\ then\ C}_a; \underbrace{(while\ B\ do\ C)}_X \underbrace{else\ skip}_+ \right]$$

L'insieme parzialmente ordinato continuo (c.p.o.) sarà $ZxZ \rightarrow ZxZ$.

In questo modo si è data una semantica ad un esempio sintattico, quindi l'interpretazione sarà:

$$\underbrace{[while\ B\ do\ C]}_X \stackrel{\text{uguaglianza semantica}}{=} \underbrace{(if\ B\ then\ C)}_a^*$$

3.5.1 C.P.O. Discreto e Dominio Piatto

Definizione 21:

Per ogni insieme X , data la seguente relazione d'uguaglianza:

$$x \subseteq x' \stackrel{def}{\Leftrightarrow} x = x' \quad (x, x' \in X)$$

abbiamo (D, \subseteq) chiamato il *c.p.o. discreto*.

Dato $X_\perp \stackrel{def}{=} X \cup \{\perp\}$, dove il simbolo \perp è un elemento non appartenente ad X .

Allora:

$$d \subseteq d' \stackrel{def}{\Leftrightarrow} (d = d') \vee (d = \perp) \quad (d, d' \in X_\perp)$$

abbiamo che (X_\perp, \subseteq) è un dominio, con almeno l'elemento \perp , chiamato *dominio piatto*.

È da notare che a causa dell'antisimmetria di \subseteq , l'insieme S ha al più un elemento minimo. Inoltre il più piccolo elemento di un sottoinsieme di un insieme parzialmente ordinato potrebbe non esistere (per esempio nell'insieme dei \mathbb{Z} con il suo usuale ordinamento parziale non ha il più piccolo elemento).

Se invece esiste, identificheremo tale elemento con il simbolo \perp_D oppure con solo \perp se è sottinteso l'insieme dal contesto. L'elemento minimo \perp è unicamente determinato dalla proprietà :

$$\forall d \in D . \perp \subseteq d$$

3.5.2 Definizione di Catena

Una catena in un insieme parzialmente ordinato D , è una sequenza di elementi di D tale che:

$$d_0 \subseteq d_1 \subseteq d_2 \subseteq d_3 \subseteq \dots$$

Un limite superiore per la catena è qualsiasi $d \in D$ tale che $\forall n \in \mathbb{N} . d_n \subseteq d$. Se esiste, il più piccolo limite superiore, (least upper bound – lub), della catena è dato da :

$$\bigcup_{n \geq 0} d_n$$

Allora avremo che :

- $\forall m \in \mathbb{N} \quad d_m \sqsubseteq \bigcup_{n \geq 0} d_n$
- Per alcuni $d \in D$ se $\forall m \in \mathbb{N} \quad d_m \sqsubseteq d$ allora $\bigcup_{n \geq 0} d_n \sqsubseteq d$

Una catena completa o chiusa in un insieme parzialmente ordinato (c.p.o.) è un insieme (D, \sqsubseteq) in cui tutte le catene incrementali $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq d_3 \sqsubseteq \dots$ hanno il più piccolo limite superiore, $\bigcup_{n \geq 0} d_n$:

- $\forall m \in \mathbb{N} \quad d_m \sqsubseteq \bigcup_{n \geq 0} d_n$ (Lub 1)
- $\forall d \in D \quad (\forall m \geq 0 \quad d_m \sqsubseteq d) \Rightarrow \bigcup_{n \geq 0} d_n \sqsubseteq d$ (Lub 2)

Un dominio è una catena di un insieme parzialmente ordinato che possiede il più piccolo elemento \perp :

$$\forall d \in D. \perp \sqsubseteq d$$

3.6 Domini di funzioni parziali

Un sottodominio di funzioni parziali f con dominio definito : $dom(f) \subseteq X$ con valori in Y .

Un ordine parziale è : $f \sqsubseteq g \Leftrightarrow dom(f) \subseteq dom(g) \quad e \quad \forall x \in dom(f) \quad f(x) = g(x)$

Il più piccolo limite superiore della catena $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \dots$ è la funzione parziale f tale che:

$$dom(f) = \bigcup_{n \geq 0} dom(f_n)$$

$$f(x) = \begin{cases} f_n(x) & \text{se } x \in dom(f_n), \text{ per qualche } n \\ \text{indefinita} & \text{altrimenti} \end{cases}$$

L'elemento minimo \perp è la funzione parziale totale indefinita.

3.7 Monotonicità, Continuità ed Esattezza.

Una funzione $f : D \rightarrow E$ insiemi parzialmente ordinati è monotona se e solo se :

$$\forall d, d' \in D \quad d \sqsubseteq d' \Rightarrow f(d) \sqsubseteq f(d')$$

Se D ed E sono insiemi parzialmente ordinati, la funzione f è continua se e solo se è monotona e contiene i più piccoli estremi superiori delle catene del tipo $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \dots \in D$, ed in questo caso avremo che :

$$f\left(\bigcup_{n \geq 0} d_n\right) = \bigcup_{n \geq 0} f(d_n) \text{ in } E$$

Se D ed E hanno un elemento minimo, allora la funzione f è esatta se e solo se $f(\perp) = \perp$.

È da notare che se $f : D \rightarrow E$ è monotona e $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \dots \in D$ è una catena, allora applicando la funzione f otterremo una catena del tipo $f(d_0) \sqsubseteq f(d_1) \sqsubseteq f(d_2) \dots \in E$.

Inoltre se d è un estremo superiore della prima catena, allora $f(d)$ è un estremo superiore della seconda e quindi è più grande dei minimi estremi superiori. Quindi se $f : D \rightarrow E$ è una funzione monotona tra insiemi di tipo c.p.o., avremo che :

$$\bigcup_{n \geq 0} f(d_n) \sqsubseteq f\left(\bigcup_{n \geq 0} d_n\right)$$

Allora utilizzando la proprietà dell'antisimmetria di \sqsubseteq , per provare che una funzione f monotona tra insiemi di tipo c.p.o. è sufficiente verificare per ogni catena $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \dots \in D$ che :

$$f\left(\bigcup_{n \geq 0} d_n\right) \sqsubseteq \bigcup_{n \geq 0} f(d_n)$$

Tutte appartenenti all'insieme E .

3.8 Domini Piatti

Dato un insieme X , la relazione d'uguaglianza:

$$x \sqsubseteq x' \stackrel{def}{\Leftrightarrow} x = x' \quad (x, x' \in X)$$

definisce (X, \sqsubseteq) come un insieme parzialmente ordinato continuo, chiamato c.p.o. discreto dell'insieme X .

Dato $X_\perp \stackrel{def}{=} X \cup \{\perp\}$, dove il simbolo \perp è un elemento non appartenente all'insieme X .

Allora:

$$d \sqsubseteq d' \stackrel{def}{=} (d = d') \vee (d = \perp) \quad (d, d' \in X_\perp)$$

fornisce (X_\perp, \sqsubseteq) come un dominio con almeno l'elemento \perp , chiamato dominio piatto.

Esempio:

Il dominio piatto dei numeri naturali, \mathbb{N}_\perp :

$$\begin{array}{ccccccccccc} 0 & 1 & 2 & 3 & 4 & \dots & n & n+1 & \dots \\ & & & & \dots & & & & \\ & & & & \perp & & & & \end{array}$$

oppure il dominio piatto dei valori booleani, B_\perp :

$$\begin{array}{ccc} \text{true} & & \text{false} \\ & & \\ & & \perp \end{array}$$

Proposizione 7:

Data una funzione parziale $f : X \rightarrow Y$, allora $f_\perp : X_\perp \rightarrow Y_\perp$:

$$f_\perp(d) = \begin{cases} f(d) & \text{se } d \in X \text{ e } f \text{ è definita in } d \\ \perp & \text{se } d \in X \text{ e } f \text{ non è definita in } d \\ \perp & \text{se } d = \perp \end{cases}$$

definisce una funzione continua tra i corrispondenti domini piatti.

Proposizione 8:

Per ogni dominio D , la funzione:

$$if : B_\perp \times (D \times D) \rightarrow D$$

$$if(x, (d, d')) \stackrel{def}{=} \begin{cases} d & \text{se } x = \text{true} \\ d' & \text{se } x = \text{false} \\ \perp_D & \text{se } x = \perp \end{cases}$$

è continua.

3.9 Prodotto tra Domini

Il prodotto di due domini parzialmente ordinati continui (D_1, \sqsubseteq_1) e (D_2, \sqsubseteq_2) è dato da:

$$D_1 \times D_2 = \{(d_1, d_2) \mid d_1 \in D_1 \text{ e } d_2 \in D_2\}$$

e l'ordine parziale \sqsubseteq definisce che:

$$(d_1, d_2) \sqsubseteq (d'_1, d'_2) \stackrel{def}{\Leftrightarrow} d_1 \sqsubseteq_1 d'_1 \& d_2 \sqsubseteq_2 d'_2$$

Le catene chiuse sono calcolate componente per componente:

$$\bigcup_{n \geq 0} (d_{1,n}, d_{2,n}) = \left(\bigcup_{i \geq 0} d_{1,i}, \bigcup_{j \geq 0} d_{2,j} \right)$$

Se (D_1, \sqsubseteq_1) e (D_2, \sqsubseteq_2) sono domini come $(D_1 \times D_2, \sqsubseteq)$, allora:

$$\perp_{D_1 \times D_2} = (\perp_{D_1}, \perp_{D_2})$$

Proposizione 9:

Dati D, E, F insiemi parzialmente ordinati continui, una funzione $f : D \times E \rightarrow F$ è monotona se e solo se è monotona in ogni argomento separatamente:

$$\begin{aligned} \forall d, d' \in D, e \in E. d \sqsubseteq d' &\Rightarrow f(d, e) \sqsubseteq f(d', e) \\ \forall d \in D, e, e' \in E. e \sqsubseteq e' &\Rightarrow f(d, e) \sqsubseteq f(d, e'). \end{aligned}$$

Inoltre la funzione f è continua se e solo se preserva catene chiuse in ogni argomento separato, cioè:

$$\begin{aligned} f\left(\bigcup_{m \geq 0} d_m, e\right) &= \bigcup_{m \geq 0} f(d_m, e) \\ f\left(d, \bigcup_{n \geq 0} e_n\right) &= \bigcup_{n \geq 0} f(d, e_n) \end{aligned}$$

Dimostrazione:

(\Leftarrow)

Se $d \sqsubseteq d'$ allora:

$$(d, e) \sqsubseteq (d', e)$$

e

$$\left(\bigcup_{m \geq 0} d_m, e\right) = \bigcup_{m \geq 0} (d_m, e)$$

come anche i fattori comuni dell'argomento di destra.

(\Rightarrow)

Supponiamo che la funzione f sia monotona per ogni argomento separatamente. Allora avremo che:

$$(d, e) \sqsubseteq (d', e') \in D \times E$$

Dalla definizione di ordinamento parziale sui prodotti binari abbiamo che:

$$\begin{aligned} d &\sqsubseteq d' \in D \\ e &\sqsubseteq e' \in E \end{aligned}$$

Quindi:

$$\begin{aligned} f(d, e) &\sqsubseteq f(d', e) && \text{Per la monotonicità del primo argomento} \\ &\sqsubseteq f(d', e') && \text{Per la monotonicità del secondo argomento} \end{aligned}$$

E quindi per la transitività, avremo:

$$f(d, e) \sqsubseteq f(d', e')$$

come richiesto.

Ora supponiamo che la funzione f sia continua in ogni argomento separatamente. Prendiamo una catena così costituita:

$$(d_0, e_0) \sqsubseteq (d_1, e_1) \sqsubseteq (d_2, e_2) \sqsubseteq (d_3, e_3) \sqsubseteq \dots$$

per il prodotto binario avremo:

$$\begin{aligned} f\left(\bigcup_{n \geq 0} (d_n, e_n)\right) &= f\left(\bigcup_{i \geq 0} d_i, \bigcup_{j \geq 0} e_j\right) && \text{per le proprietà del prodotto tra domini} \\ &= \bigcup_{i \geq 0} f\left(d_i, \bigcup_{j \geq 0} e_j\right) && \text{per la continuità del primo argomento} \\ &= \bigcup_{i \geq 0} \left(\bigcup_{j \geq 0} f(d_i, e_j)\right) && \text{per la continuità del secondo argomento} \\ &= \bigcup_{n \geq 0} f(d_n, e_n) && \text{per il lemma sulla diagonalizzazione delle catene doppie} \end{aligned}$$

Ne consegue la continuità della funzione f . ■

Lemma 3: Diagonalizzazione delle catene doppie

Dato un insieme parzialmente ordinato continuo D , supponiamo che la famiglia ad indice doppio di elementi del tipo $d_{m,n} \in D$ ($m, n \geq 0$) soddisfino :

$$m \leq m' \ \& \ n \leq n' \Rightarrow d_{m,n} \sqsubseteq d_{m',n'} \quad (6)$$

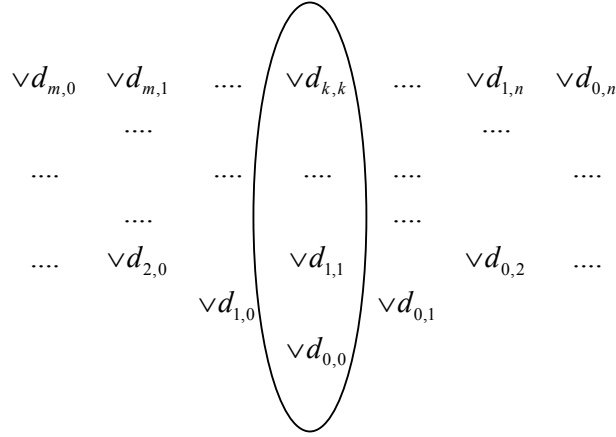
Allora avremo:

$$\bigcup_{n \geq 0} d_{0,n} \sqsubseteq \bigcup_{n \geq 0} d_{1,n} \sqsubseteq \bigcup_{n \geq 0} d_{2,n} \sqsubseteq \bigcup_{n \geq 0} d_{3,n} \sqsubseteq \dots$$

e quindi:

$$\bigcup_{m \geq 0} \left(\bigcup_{n \geq 0} d_{m,n} \right) = \bigcup_{k \geq 0} d_{k,k} = \bigcup_{n \geq 0} \left(\bigcup_{m \geq 0} d_{m,n} \right) \quad (7)$$

Graficamente avremo:



La colonna al centro rappresenta $\bigcup_{k \geq 0} d_{k,k}$

Dimostrazione:

Per la dimostrazione utilizzeremo le proprietà viste in precedenza sul più piccolo limite superiore (lub) di una catena.

Per primo vediamo che se $m \leq m'$ allora:

$$\begin{aligned} d_{m,n} &\sqsubseteq d_{m',n} && \text{per la proprietà (6) vista in precedenza} \\ &\sqsubseteq \bigcup_{n' \geq 0} d_{m',n'} && \text{per la proprietà (lub 1)} \end{aligned}$$

per tutti gli $n \geq 0$,

$$\bigcup_{n \geq 0} d_{m,n} \sqsubseteq \bigcup_{n' \geq 0} d_{m',n'} \quad \text{per la proprietà (lub2)}$$

Così avremo una catena di minimi limiti superiori :

$$\bigcup_{n \geq 0} d_{0,n} \sqsubseteq \bigcup_{n \geq 0} d_{1,n} \sqsubseteq \bigcup_{n \geq 0} d_{2,n} \sqsubseteq \bigcup_{n \geq 0} d_{3,n} \sqsubseteq \dots$$

Possiamo costruire la catena: $\bigcup_{m \geq 0} \bigcup_{n \geq 0} d_{m,n}$

Utilizzando la proprietà (lub 1) abbiamo:

$$d_{k,k} \sqsubseteq \bigcup_{n \geq 0} d_{k,n} \sqsubseteq \bigcup_{m \geq 0} \bigcup_{n \geq 0} d_{m,n}$$

per ogni $k \geq 0$, e quindi per la proprietà (lub 2) :

(8)

$$\bigcup_{k \geq 0} d_{k,k} \sqsubseteq \bigcup_{m \geq 0} \bigcup_{n \geq 0} d_{m,n}$$

Al contrario, per ogni $m, n \geq 0$, abbiamo che :

$$\begin{aligned} d_{m,n} &\sqsubseteq d_{\max\{m,n\}, \max\{m,n\}} && \text{per la proprietà (6)} \\ &\sqsubseteq \bigcup_{k \geq 0} d_{k,k} && \text{per la proprietà (lub1)} \end{aligned}$$

Ora applicando la proprietà (lub 2) ad entrambi abbiamo:

$$\bigcup_{m \geq 0} \bigcup_{n \geq 0} d_{m,n} \sqsubseteq \bigcup_{k \geq 0} d_{k,k} \quad (9)$$

Combinando la (8) e la (9) con la proprietà dell'antisimmetria di \sqsubseteq si ha l'uguaglianza desiderata. Otterremmo in ogni caso l'uguaglianza anche cambiando l'ordine di m ed n . ■

3.10 Domini di Funzioni

Proposizione 10:

Dati due insiemi (D, \sqsubseteq_D) e (E, \sqsubseteq_E) di tipo c.p.o., la funzione c.p.o. $(D \rightarrow E, \sqsubseteq)$ è così definita:

$$D \rightarrow E \stackrel{def}{=} \{f \mid f : D \rightarrow E \text{ è una funzione continua}\}$$

ed è parzialmente ordinata così:

$$f \sqsubseteq f' \stackrel{def}{\Leftrightarrow} \forall d \in D \quad f(d) \sqsubseteq_E f'(d)$$

Il più piccolo estremo superiore della catena è calcolato elemento per elemento:

$$\left(\bigcup_{n \geq 0} f_n \right)(d) = \bigcup_{n \geq 0} f_n(d)$$

Se E è un dominio allora avremo che $D \rightarrow E$ e $\perp_{D \rightarrow E}(d) = \perp_E$ per ogni $d \in D$.

Dimostrazione:

Dobbiamo dimostrare che l'estremo superiore di una catena di funzioni, $\bigcup_{n \geq 0} f_n$, sia continua. Data una catena nell'insieme D :

$$\left(\bigcup_{n \geq 0} f_n \right) \left(\left(\bigcup_{m \geq 0} d_m \right) \right) = \bigcup_{n \geq 0} \left(f_n \left(\bigcup_{m \geq 0} d_m \right) \right) \quad \text{Per definizione di } \bigcup_{n \geq 0} f_n$$

$$\begin{aligned}
 &= \bigcup_{n \geq 0} \left(\bigcup_{m \geq 0} f_n(d_m) \right) && \text{Per la continuità di ogni } f_n \\
 &= \bigcup_{m \geq 0} \left(\bigcup_{n \geq 0} f_n(d_m) \right) && \text{Per la proprietà (7) sulla diagonalizzazione delle catene} \\
 &= \bigcup_{m \geq 0} \left(\left(\bigcup_{n \geq 0} f_n \right) (d_m) \right) && \text{Per definizione di } \bigcup_{n \geq 0} f_n \blacksquare
 \end{aligned}$$

3.11 La funzione Evaluation e la funzione Currying

Proposizione 11:

Dati due insiemi D ed E di tipo c.p.o., la funzione:

$$\begin{aligned}
 &ev : (D \rightarrow E) \times D \rightarrow E \\
 &ev(f, d) \stackrel{def}{=} f(d)
 \end{aligned}$$

è continua.

Data una funzione continua $f : D' \times D \rightarrow E$ con D' di tipo c.p.o., per ogni $d' \in D'$ la funzione $d \in D \mapsto f(d, d')$ è continua e determina un elemento della funzione di tipo c.p.o. $D \rightarrow E$ che chiameremo $cur(f)(d')$.

Allora:

$$\begin{aligned}
 &cur(f) : D' \rightarrow (D \rightarrow E) \\
 &cur(f)(d') \stackrel{def}{=} \lambda d \in D \quad f(d', d)
 \end{aligned}$$

è una funzione continua.

Dimostrazione:

$$\begin{aligned}
 ev \left(\bigcup_{n \geq 0} (f_n, d_n) \right) &= ev \left(\bigcup_{i \geq 0} f_i, \bigcup_{j \geq 0} d_j \right) && \text{per i minimi estremi sup. nelle produzioni} \\
 &= \left(\bigcup_{i \geq 0} f_i \right) \left(\bigcup_{j \geq 0} d_j \right) && \text{per la def. della funzione } ev \\
 &= \bigcup_{i \geq 0} f_i \left(\bigcup_{j \geq 0} d_j \right) && \text{per i minimi estremi sup. nelle funzioni di tipo c.p.o.} \\
 &= \bigcup_{i \geq 0} \bigcup_{j \geq 0} f_i(d_j) && \text{per la continuità di ogni } f_i \\
 &= \bigcup_{n \geq 0} f_n(d_n) && \text{per il lemma sulla diagonalizzazione}
 \end{aligned}$$

$$= \bigcup_{n \geq 0} ev(f_n, d_n) \quad \text{per la def. della funzione } ev$$

In questo modo abbiamo dimostrato che la funzione ev è continua.

Per dimostrare la continuità di ogni $cur(f)(d')$ e quindi di $cur(f)$ segue dal fatto che ogni minimo estremo superiore delle catene $D_1 \times D_2$ possono essere calcolate componente per componente. ■

Così abbiamo dimostrato che esiste una funzione ev che sia continua e definita in $[D \times E] \times D \rightarrow E$. Inoltre abbiamo dimostrato che presa un'altra funzione $f \in D' \times D$, allora esiste ed è unica la funzione continua $cur(f)$ in modo da costruire il seguente grafo:

$$\begin{array}{ccc} & & \xrightarrow{ev} \\ ev : [D \rightarrow E] \times D & \nearrow & E \\ \uparrow cur(g) \times I_D & & \\ D' \times D & & \end{array}$$

3.12 Punto Fisso e Minimo Punto Prefisso

Un *punto fisso* per una funzione $f : D \rightarrow D$ è l'elemento $d \in D$ che soddisfa $f(d) = d$. Se D è un insieme parzialmente ordinato, possiamo considerare la nozione più debole di *punto prefisso*.

Dato D un insieme parzialmente ordinato e $f : D \rightarrow D$ una funzione. Un elemento $d \in D$ è un punto prefisso di f se soddisfa $f(d) \sqsubseteq d$.

Il minimo punto prefisso di f , se esiste, è dato dall'operatore:

$$fix(f)$$

È così specificato univocamente dalle seguenti due proprietà:

$$\bullet \quad f(fix(f)) \sqsubseteq fix(f) \quad (10)$$

$$\bullet \quad \forall d \in D \quad f(d) \sqsubseteq d \Rightarrow fix(f) \sqsubseteq d \quad (11)$$

Proposizione 12:

Supponiamo un insieme parzialmente ordinato D e una funzione $f : D \rightarrow D$ che possiede un minimo punto prefisso $fix(f)$. Visto che la funzione f è monotona, $fix(f)$ è un particolare punto fisso di f (e da ora è il più piccolo elemento dell'insieme dei punti fissi di f).

Dimostrazione:

Dalla definizione, $fix(f)$ soddisfa la proprietà (10). Se la funzione f è monotona, possiamo applicare f ad entrambi i lati di (10), così avremo:

$$f(f(fix(f))) \sqsubseteq f(fix(f))$$

Applicando la proprietà (11) con $d = f(fix(f))$, avremo:

$$fix(f) \sqsubseteq f(fix(f))$$

Combinando così la proprietà (10) e la proprietà dell'antisimmetria dell'ordinamento parziale \sqsubseteq , avremo che $f(fix(f)) = fix(f)$, come richiesto. ■

3.13 Teorema del punto fisso di TARSKI:

Teorema 3:

Data la funzione $f : D \rightarrow D$ continua. Allora:

- F possiede un minimo punto prefisso dato da :

$$fix(f) = \bigcup_{n \geq 0} f^n(\perp)$$

- Inoltre $fix(f)$ è un punto fisso di f se soddisfa $f(fix(f)) = fix(f)$ e quindi è il minimo punto fisso di f . ■

Questo teorema ci fornisce la chiave sulle funzioni continue su domini e ci consente di dare dei programmi di semantica denotazionale che hanno caratteristiche ricorsive.

La notazione $f^n(\perp)$ usato nel precedente teorema è così definito:

$$\begin{cases} f^0(\perp) & \stackrel{def}{=} \perp \\ f^{n+1}(\perp) & \stackrel{def}{=} f(f^n(\perp)) \end{cases} \quad (12)$$

Abbiamo che $\forall d \in D \quad \perp \sqsubseteq d$ e quindi:

$$f^0(\perp) = \perp \sqsubseteq f^1(\perp)$$

e per la monotonicità della funzione f :

$$f^n(\perp) \sqsubseteq f^{n+1}(\perp) \Rightarrow f^{n+1}(\perp) = f(f^n(\perp)) \sqsubseteq f(f^{n+1}(\perp)) = f^{n+2}(\perp)$$

Quindi per induzione su $n \in \mathbb{N}$, abbiamo che $\forall n \in \mathbb{N} \quad f^n(\perp) \sqsubseteq f^{n+1}(\perp)$. In altre parole gli elementi $f^n(\perp)$ formano una catena nell'insieme D . Così finché l'insieme D è di tipo c.p.o., ha un significato il più piccolo estremo superiore usato per definire $\text{fix}(f)$ nel teorema del punto fisso di Tarski.

Dimostrazione del Teorema del Punto Fisso di Tarski:

Notiamo che :

$$\begin{aligned}
 f(\text{fix}(f)) &= f\left(\bigcup_{n \geq 0} f^n(\perp)\right) \\
 &= \bigcup_{n \geq 0} f(f^n(\perp)) && \text{per la continuità di } f \\
 &= \bigcup_{n \geq 0} f^{n+1}(\perp) && \text{per la proprietà (12)} \\
 &= \bigcup_{n \geq 0} f^n(\perp) && \text{perché lasciare un elemento della} \\
 & && \text{catena non comporta effetti} \\
 &= \text{fix}(f)
 \end{aligned}$$

Così $\text{fix}(f)$ è davvero un punto fisso per f ed in particolare soddisfa la proprietà (10) sui punti fissi. Per verificare la proprietà (11) dobbiamo cercare il minimo punto prefisso, supponendo che $d \in D$ soddisfa $f(d) \sqsubseteq d$.

Allora finché \perp è il più piccolo elemento nell'insieme D ,

$$f^0(\perp) = \perp \sqsubseteq d$$

e

$$\begin{aligned}
 f^n(\perp) \sqsubseteq d \quad \Rightarrow \quad f^{n+1}(\perp) = f(f^n(\perp)) &\sqsubseteq f(d) && \text{per la monotonicità di } f \\
 &\sqsubseteq d && \text{perché } f(d) \sqsubseteq d
 \end{aligned}$$

Quindi per induzione su $n \in \mathbb{N}$ abbiamo che $\forall n \in \mathbb{N} \quad f^n(\perp) \sqsubseteq d$. Allora d è un estremo superiore della catena e quindi :

$$\text{fix}(f) = \bigcup_{n \geq 0} f^n(\perp) \sqsubseteq d$$

Come richiesto dalla proprietà (11). ■

3.14 Continuità dell'Operatore di Punto Fisso

Proposizione 13:

Dato un dominio D , per il teorema del punto fisso di Tarski sappiamo che ogni funzione continua $f \in (D \rightarrow D)$ possiede un minimo punto fisso, chiamato $\text{fix}(f) \in D$.

Allora la funzione:

$$\text{fix} : (D \rightarrow D) \rightarrow D$$

è continua.

Dimostrazione:

Dobbiamo per primo provare che $\text{fix} : (D \rightarrow D) \rightarrow D$ sia una funzione monotona. Supponiamo che $f_1 \sqsubseteq f_2$ siano nel dominio di funzioni $D \rightarrow D$. Dobbiamo provare che $\text{fix}(f_1) \sqsubseteq \text{fix}(f_2)$.

Ma:

$$\begin{aligned} f_1(\text{fix}(f_2)) &\sqsubseteq f_2(\text{fix}(f_2)) && \text{perché } f_1 \sqsubseteq f_2 \\ &\sqsubseteq \text{fix}(f_2) && \text{per la propr. (10)} \\ &&& \text{applicata a } \text{fix}(f_2) \end{aligned}$$

Così $\text{fix}(f_2)$ è un punto prefisso di f_1 e quindi per la proprietà (11) su $\text{fix}(f_1)$ abbiamo che $\text{fix}(f_1) \sqsubseteq \text{fix}(f_2)$, come richiesto.

Mantenendo i minimi estremi superiori delle catene, supponiamo $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq f_3 \dots$ in $D \rightarrow D$.

Richiamando le proprietà di monotonicità e continuità delle funzioni viste in precedenza, dobbiamo provare che :

$$\text{fix}\left(\bigcup_{n \geq 0} f_n\right) \sqsubseteq \bigcup_{n \geq 0} \text{fix}(f_n)$$

e per la proprietà (11) dei minimi punti prefissi è sufficiente far vedere che $\bigcup_{n \geq 0} \text{fix}(f_n)$ è un punto prefisso per la funzione $\bigcup_{n \geq 0} f_n$.

Questo perché:

$$\begin{aligned} \left(\bigcup_{m \geq 0} f_m\right)\left(\bigcup_{n \geq 0} \text{fix}(f_n)\right) &= \bigcup_{m \geq 0} f_m\left(\bigcup_{n \geq 0} \text{fix}(f_n)\right) \\ &= \bigcup_{m \geq 0} \bigcup_{n \geq 0} f_m(\text{fix}(f_n)) && \text{per la continuità di ogni } f_m \\ &= \bigcup_{k \geq 0} f_k(\text{fix}(f_k)) && \text{per il lemma sulla diagonalizzazione} \\ &\sqsubseteq \bigcup_{k \geq 0} \text{fix}(f_k) && \text{delle catene doppie} \\ &&& \text{per la proprietà (10) di ogni } f_k \blacksquare \end{aligned}$$

Un ciclo come “While B do C” non è facile da esporre composizionalmente. La semantica transizionale di un ciclo while sarà:

$$\langle \text{while } B \text{ do } C, s \rangle \rightarrow \langle \text{if } B \text{ then } C; (\text{while } B \text{ do } C) \text{ else skip}, s \rangle$$

Suggerisce che la denotazione è una funzione parziale da uno stato all'altro in modo da soddisfare:

$$\llbracket \text{while } B \text{ do } C \rrbracket = \llbracket \text{if } B \text{ then } C; (\text{while } B \text{ do } C) \text{ else skip} \rrbracket \quad (13)$$

Questa non può essere utilizzata direttamente per definire $\llbracket \text{while } B \text{ do } C \rrbracket$, finquando il lato destro contiene una sottofrase la cui denotazione proviamo a definire.

Utilizzando la semantica denotazionale della composizione sequenziale e la “if”, oltre al fatto che la denotazione di “skip” è la funzione $\lambda s \in \text{State}.s$, la (13) è come dire che $\llbracket \text{while } B \text{ do } C \rrbracket$ è soluzione dell’equazione di punto fisso seguente.

3.15 Proprietà del punto fisso di $\llbracket \text{while } B \text{ do } C \rrbracket$:

Dato:

$$\llbracket \text{while } B \text{ do } C \rrbracket = f_{\llbracket B \rrbracket \llbracket C \rrbracket}(\llbracket \text{while } B \text{ do } C \rrbracket)$$

dove $\forall b : \text{State} \rightarrow \{\text{true}, \text{false}\}$ e $c, w : \text{State} \rightarrow \text{State}$, definiamo:

$$f_{b,c}(w) = \lambda s \in \text{State}. \text{if } (b(s), w(c(s)), s) \quad (14)$$

Le equazioni di punto fisso si presentano molto spesso per poter fornire una semantica denotazionale ai linguaggi con caratteristiche ricorsive. Dana Scott iniziò gli studi negli anni '60, creando una teoria matematica chiamata *teoria dei domini*, sviluppando un metodo con cui sarebbe stato possibile non solo trovare la soluzione delle equazioni di punto fisso presentate da semantiche denotazionali, ma anche trovare soluzioni che sarebbero state minimali, in modo da avere una corrispondenza esatta tra la semantica denotazionale e quella operativa. L’idea è di considerare un ordine parziale tra gli oggetti matematici usati come denotazioni, in cui si esprime che un oggetto è “approssimato per” o “possiede più informazioni di” o “più definito di”. Allora la soluzione minima di un’equazione di punto fisso può essere costruita come il limite di una catena crescente d’approssimazioni della soluzione.

Vediamo un esempio per capire come funziona.

Esempio:

$$\text{while } X > 0 \text{ do } (Y := X * Y; X := X - 1) \quad (15)$$

dove X e Y sono due distinte locazioni di memoria intere (variabili). Possiamo così prendere uno stato con una coppia di interi (x,y), salvando il contenuto corrente di X e Y rispettivamente.

Quindi $\text{State} = \mathbb{Z} \times \mathbb{Z}$, (ma potrebbe anche essere $\text{State} = \mathbb{N} \times \mathbb{N}$ indifferentemente), proviamo a definire la (15) come una funzione parziale $w : (\mathbb{Z} \times \mathbb{Z}) \rightarrow (\mathbb{Z} \times \mathbb{Z})$. Questa denotazione ha soluzione del tipo presentato in (14).

Per l’espressione booleana $B = (X > 0)$ ed il comando $C = (Y := X * Y; X := X - 1)$, la funzione $f_{\llbracket B \rrbracket \llbracket C \rrbracket}$ coincide con la funzione f che ora vedremo.

Dati:

$$\overset{\text{def}}{\text{State}} = \mathbb{Z} \times \mathbb{Z} \quad \text{coppie di interi}$$

$$D \stackrel{def}{=} State \rightarrow State \quad \text{funzioni parziali}$$

Data $\llbracket \text{while } X > 0 \text{ do } Y := X * Y; X := X - 1 \rrbracket \in D$, cercheremo la soluzione minima per $w = f(w)$ dove $f : D \rightarrow D$ è definita da:

$$f(w)(x, y) = \begin{cases} (x, y) & \text{se } x \leq 0 \\ w(x-1, x * y) & \text{se } x > 0 \end{cases} \quad (16)$$

Consideriamo l'ordinamento parziale \sqsubseteq tra elementi di $D = State \rightarrow State$, cioè dati $State \stackrel{def}{=} \mathbb{Z} \times \mathbb{Z}$ e $D \stackrel{def}{=} State \rightarrow State$, $w \sqsubseteq w'$ se e solo se $\forall (x, y) \in State$ se w è definito come (x, y) allora anche w' è definito come (x, y) ed inoltre $w(x, y) = w'(x, y)$.

Ricordiamo inoltre che il più piccolo elemento $\perp \in D$ è:

$$\perp \stackrel{def}{=} \text{funzione parziale totalmente indefinita}$$

in cui $\perp \sqsubseteq w, \forall w \in D$.

Iniziamo la procedura con l'elemento minimo \perp , applicando ricorsivamente la funzione costruendo così una sequenza di funzioni parziali w_0, w_1, w_2, \dots :

$$\begin{cases} w_0 \stackrel{def}{=} \perp \\ w_{n+1} \stackrel{def}{=} f(w_n) \end{cases}$$

Utilizzando la (16) troviamo che :

$$w_1(x, y) = f(\perp)(x, y) = \begin{cases} (x, y) & \text{se } x \leq 0 \\ \text{indefinita}(\perp) & \text{se } x \geq 1 \end{cases}$$

$$w_2(x, y) = f(w_1)(x, y) = \begin{cases} (x, y) & \text{se } x \leq 0 \\ (0, y) & \text{se } x = 1 \\ \text{indefinita}(\perp) & \text{se } x \geq 2 \end{cases}$$

$$w_3(x, y) = f(w_2)(x, y) = \begin{cases} (x, y) & \text{se } x \leq 0 \\ (0, y) & \text{se } x = 1 \\ (0, 1 * 2 * y) & \text{se } x = 2 \\ \text{indefinita}(\perp) & \text{se } x \geq 3 \end{cases}$$

$$w_4(x, y) = f(w_3)(x, y) = \begin{cases} (x, y) & \text{se } x \leq 0 \\ (0, y) & \text{se } x = 1 \\ (0, 1 * 2 * y) & \text{se } x = 2 \\ (0, 1 * 2 * 3 * y) & \text{se } x = 3 \\ \text{indefinita}(\perp) & \text{se } x \geq 4 \end{cases}$$

In generale avremo:

$$w_n(x, y) = f(w_{n-1})(x, y) = \begin{cases} (x, y) & \text{se } x \leq 0 \\ (0, (!x) * y) & \text{se } 0 < x < n \\ \text{indefinita}(\perp) & \text{se } x \geq n \end{cases}$$

Otteniamo così una sequenza crescente di funzioni parziali:

$$w_0 \sqsubseteq w_1 \sqsubseteq w_2 \sqsubseteq \dots \sqsubseteq w_n \sqsubseteq \dots$$

definite su insiemi sempre più grandi degli stati (x, y) e coincidenti secondo la loro definizione. L'unione di tutte queste funzioni parziali è l'elemento $w_\infty \in D$ dato da:

$$w_\infty(x, y) = \begin{cases} (x, y) & \text{se } x \leq 0 \\ (0, (!x) * y) & \text{se } x > 0 \end{cases}$$

Da notare che w_∞ è un punto fisso della funzione f , e per ogni (x, y) abbiamo:

$$\begin{aligned} f(w_\infty)(x, y) &= \begin{cases} (x, y) & \text{se } x \leq 0 \\ w_\infty(x-1, x * y) & \text{se } x > 0 \end{cases} && \text{dalla def. di } f \\ &= \begin{cases} (x, y) & \text{se } x \leq 0 \\ 0, 1 * y & \text{se } x = 1 \\ (0, !(x-1) * x * y) & \text{se } x > 1 \end{cases} && \text{per def. di } w_\infty \\ &= w_\infty(x, y) \end{aligned}$$

Infatti si può notare che w_∞ è il minimo punto fisso della funzione f , nel senso che $\forall w \in D$:

$$w = f(w) \quad \Rightarrow \quad w_\infty \sqsubseteq w$$

Questo minimo punto fisso lo consideriamo come la denotazione di:

$$\text{while } X > 0 \text{ do } (Y := X * Y; X := X - 1)$$

Il suo costrutto è un'istanza del teorema 3 del punto fisso di Tarski visto in precedenza.

3.16 Catene chiuse e insiemi ammissibili

Abbiamo già visto che il minimo punto fisso di una funzione continua $f : D \rightarrow D$ su un dominio D può essere espresso come il minimo estremo superiore di una catena costruita applicando ripetutamente la funzione f partendo dall'elemento minimo $\perp \in D$: $\text{fix}(f) = \bigcup_{n \geq 0} f^n(\perp)$ (vedi teorema 3).

Dato D un insieme di tipo c.p.o., un sottoinsieme $S \subseteq D$ è chiamato catena chiusa se e solo se per ogni catena $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq d_3 \sqsubseteq \dots \in D$ avremo che:

$$(\forall n \geq 0 \quad d_n \in S) \Rightarrow \left(\bigcup_{n \geq 0} d_n \right) \in S$$

Se D è un dominio, $S \subseteq D$ è ammissibile se e solo se è una catena chiusa sottoinsieme di D e $\perp \in S$. Ricordiamo che la proprietà $\Phi(d)$ degli elementi $d \in D$ è chiamata catena chiusa o ammissibile se e solo se $\{d \in D \mid \Phi(d)\}$ è una catena chiusa o ammissibile sottoinsieme di D .

Esempio:

Prendo i numeri naturali verticali Ω :

ω

\dots

$n+1$

n

\dots

2

1

0

- Ogni sottoinsieme finito di Ω è una catena chiusa.
- Il sottoinsieme $\{0, 2, 4, 6, \dots\}$ non è una catena chiusa.
- Il sottoinsieme $\{0, 2, 4, 6, \dots\} \cup \{\omega\}$ è una catena chiusa.

3.17 Principio di Induzione di Scott sui Punti Fissi

Teorema 4:

Data una funzione $f : D \rightarrow D$ continua su un dominio D . Per ogni sottoinsieme ammissibile $S \subseteq D$, per provare che il minimo punto fisso della funzione f è in S , cioè:

$$\text{fix}(f) \in S$$

è sufficiente dimostrare che :

$$\forall d \in D \quad (d \in S \Rightarrow f(d) \in S).$$

Esempio:

Supponiamo che D sia un dominio e che $f : (D \times (D \times D)) \rightarrow D$ sia una funzione continua. Data $g : (D \times D) \rightarrow (D \times D)$ una funzione continua così definita:

$$g(d_1, d_2) \stackrel{\text{def}}{=} (f(d_1, (d_1, d_2)), f(d_1, (d_2, d_2))) \quad (d_1, d_2 \in D)$$

Allora $u_1 = u_2$ quando $(u_1, u_2) \stackrel{\text{def}}{=} \text{fix}(g)$.

Dimostrazione:

Dobbiamo dimostrare che $\text{fix}(g) \in \Delta$ dove l'elemento Δ è così definito:

$$\Delta \stackrel{\text{def}}{=} \{(d_1, d_2) \in D \times D \mid d_1 = d_2\}$$

Non è difficile notare che Δ è un sottoinsieme ammissibile del dominio $D \times D$. Così attraverso il Principio di Induzione sui Punti Fissi di Scott possiamo verificare che:

$$\forall (d_1, d_2) \in D \times D \quad ((d_1, d_2) \in \Delta \Rightarrow g(d_1, d_2) \in \Delta)$$

o equivalentemente che :

$$\forall (d_1, d_2) \in D \times D \quad (d_1 = d_2 \Rightarrow f(d_1, d_1, d_2) = f(d_1, d_2, d_2))$$

che è chiaramente vero. ■

Esempio:

Data una funzione $f : D \rightarrow D$ continua in cui il minimo punto fisso è la denotazione del comando :

while $X > 0$ *do* ($Y := X * Y$; $X := X - 1$)

Useremo l'Induzione di Scott per provare che:

$$\forall x, y \geq 0, \quad x, y \in \mathbb{N} \quad \text{fix}(f)(x, y) \neq \perp \Rightarrow \text{fix}(f)(x, y) = (0, (!x) * y) \quad (17)$$

dove per $w \in D = (\mathbb{Z} \times \mathbb{Z}) \rightarrow (\mathbb{Z} \times \mathbb{Z})$ scriveremo $w(x, y) \neq \perp$, cioè che la funzione parziale w è definita in (x, y) . In altre parole possiamo identificare l'insieme D come il dominio delle funzioni continue dall'insieme discreto $(\mathbb{Z} \times \mathbb{Z})$ di tipo c.p.o. al dominio piatto $(\mathbb{Z} \times \mathbb{Z})_{\perp}$.

Dimostrazione:

Definito S come:

$$S \stackrel{\text{def}}{=} \{ w \in D \mid \forall x, y \geq 0 \quad w(x, y) \neq \perp \Rightarrow w(x, y) = (0, (!x) * y) \}$$

Non è difficile vedere che S è ammissibile. Quindi per provare la proprietà (17), attraverso l'Induzione di Scott è sufficiente verificare che $w \in S$ implica che $f(w) \in S, \forall w \in D$.

Supponiamo $w \in S$, $x, y \geq 0$ e che $f(w)(x, y) \neq \perp$. Dobbiamo far vedere che $f(w)(x, y) = (0, (!x) * y)$. Consideriamo i due casi $x=0$ e $x>0$ separatamente.

Se $x=0$, per la proprietà (16) della funzione f avremo che:

$$f(w)(x, y) = (x, y) = (0, y) = (0, 1 * y) = (0, (!0) * y) = (0, (!x) * y)$$

Se $x>1$, per la proprietà (16) della funzione f avremo che:

$$w(x-1, x * y) = f(w)(x, y) \neq \perp$$

e finché $w \in S$ e $x-1, x * y \geq 0$, noi avremo che :

$$w(x-1, x * y) = (0, !(x-1) * (x * y))$$

e quindi che:

$$f(w)(x, y) = w(x-1, x * y) = (0, !(x-1) * (x * y)) = (0, (!x) * y) \quad \blacksquare$$

La difficoltà del principio di induzione di Scott sui punti fissi in alcuni particolari casi è nell'identificazione di un sottoinsieme appropriato S ammissibile, per esempio la ricerca di un'adeguatamente forte "ipotesi di induzione".

Il prossimo esempio illustra ciò:

Esempio:

Dato un dominio D e tre funzioni continue p, h, k così definite $p : D \rightarrow Bool_{\perp}$; $h, k : D \rightarrow D$ con h esatta, cioè che $h(\perp) = \perp$.

Date $f_1, f_2 : (D \times D) \rightarrow D$ come le minime funzioni continue tali che $\forall d_1, d_2 \in D$:

$$\begin{aligned} f_1(d_1, d_2) &= if(p(d_1), d_2, h(f_1(k(d_1), d_2))) \\ f_2(d_1, d_2) &= if(p(d_1), d_2, f_2(k(d_1), h(d_2))) \end{aligned}$$

dove:

$$if(b, d_1, d_2) = \begin{cases} d_1 & se\ b = true \\ d_2 & se\ b = false \\ \perp & se\ b = \perp \end{cases}$$

Allora $f_1 = f_2$.

Per la dimostrazione abbiamo bisogno di definire altre funzioni.

Date D, p, h e k definite nell'esempio precedente, abbiamo $E : (D \times D) \rightarrow D$ tale che :

$$g \stackrel{def}{=} \langle g_1, g_2 \rangle : (E \times E) \rightarrow (E \times E)$$

Dove $g_1, g_2 : (E \times E) \rightarrow E$ sono funzioni continue così definite:

$$\begin{aligned} g_1(u_1, u_2)(d_1, d_2) &\stackrel{def}{=} \begin{cases} d_2 & se\ p(d_1) = true \\ h(u_1(k(d_1), d_2)) & se\ p(d_1) = false \\ \perp & se\ p(d_1) = \perp \end{cases} \\ g_2(u_1, u_2)(d_1, d_2) &\stackrel{def}{=} \begin{cases} d_2 & se\ p(d_1) = true \\ u_2(k(d_1), h(d_2)) & se\ p(d_1) = false \\ \perp & se\ p(d_1) = \perp \end{cases} \end{aligned}$$

$$\forall u_1, u_2 \in E \quad e \quad d_1, d_2 \in D.$$

Dimostrazione:

Per definizione di f_1 e f_2 abbiamo che $(f_1, f_2) = fix(g)$ dove g è la funzione continua definita in precedenza. Quindi per provare che $f_1 = f_2$ è sufficiente far vedere che $fix(g)$ appartiene al sottoinsieme ammissibile $\{(u_1, u_2) \in E \times E \mid u_1 = u_2\}$.

Per poter utilizzare il principio di induzione di Scott su questo sottoinsieme ammissibile, dobbiamo verificare che:

$$\forall (u_1, u_2) \in E \times E \quad ((u_1, u_2) \in \Delta \Rightarrow g(u_1, u_2) \in \Delta)$$

cioè che $\forall u \in E \quad g_1(u, u) = g_2(u, u)$.

E' evidente che per come sono state definite in precedenza g_1 e g_2 abbiamo che:

$$g_1(u, u)(d_1, d_2) = g_2(u, u)(d_1, d_2)$$

e quindi avremo che:

$$h(u(k(d_1), d_2)) = u(k(d_1), h(d_2))$$

Sfortunatamente, non c'è ragione per cui l'ultima condizione potrebbe essere soddisfatta da un arbitrario elemento $u \in E$.

Possiamo aggirare il problema applicando il principio di induzione di Scott al sottoinsieme $\{(u_1, u_2) \in E \times E \mid u_1 = u_2\}$ così che:

$$S \stackrel{def}{=} \{(u_1, u_2) \in E \times E \mid u_1 = u_2 \ \& \ \forall d_1, d_2 \in D \times D \quad h(u_1(d_1, d_2)) = u_1(d_1, h(d_2))\}$$

Dobbiamo per prima cosa verificare che l'insieme S sia ammissibile.

E' una catena completa perché se $(u_{1,0}, u_{2,0}) \sqsubseteq (u_{1,1}, u_{2,1}) \sqsubseteq (u_{1,2}, u_{2,2}) \sqsubseteq \dots$ è una catena in $E \times E$ ogni elemento è in S, allora :

$$\bigcup_{n \geq 0} (u_{1,n}, u_{2,n}) = \left(\bigcup_{i \geq 0} u_{1,i}, \bigcup_{j \geq 0} u_{2,j} \right)$$

Tutti questi elementi sono in S finché:

$$\bigcup_{n \geq 0} u_{1,n} = \bigcup_{n \geq 0} u_{2,n} \quad \text{perché } u_{1,n} = u_{2,n} \ \forall n$$

Quindi:

$$\begin{aligned} h\left(\left(\bigcup_{n \geq 0} u_{1,n}\right)(d_1, d_2)\right) &= h\left(\bigcup_{n \geq 0} u_{1,n}(d_1, d_2)\right) \\ &= \bigcup_{n \geq 0} h(u_{1,n}(d_1, d_2)) \quad \text{h è continua} \\ &= \bigcup_{n \geq 0} u_{1,n}(d_1, h(d_2)) \quad \text{ogni } u_{1,n}, u_{2,n} \text{ è in S} \end{aligned}$$

quindi è chiuso per catene.

Inoltre, l'insieme S contiene il minimo elemento $(\perp, \perp) \in E \times E$, perché quando $(u_1, u_2) = (\perp, \perp)$ avremo che $u_1 = u_2$ ed inoltre $\forall (d_1, d_2) \in D \times D$:

$$\begin{aligned}
 h(u_1(d_1, d_2)) &= h(\perp(d_1, d_2)) \\
 &= h(\perp) && \text{dalla def. di } \perp \in (D \times D) \rightarrow D \\
 &= \perp \\
 &= \perp(d_1, h(d_2)) && \text{dalla def. di } \perp \in (D \times D) \rightarrow D \\
 &= u_1(d_1, h(d_2)).
 \end{aligned}$$

Per provare che $f_1 = f_2$ bisogna verificare che $(f_1, f_2) = \text{fix}(g) \in S$; siccome l'insieme S è ammissibile, per l'induzione di Scott è sufficiente provare $\forall (u_1, u_2) \in E \times E$ che:

$$(u_1, u_2) \in S \Rightarrow (g_1(u_1, u_2), g_2(u_1, u_2)) \in S$$

Supponiamo che $(u_1, u_2) \in S$, che $u_1 = u_2$ e che:

$$\forall (d_1, d_2) \in D \times D \quad h(u_1(d_1, d_2)) = u_1(d_1, h(d_2)) \quad (18)$$

Risulta evidente dalla definizione di g_1 e g_2 che $u_1 = u_2$ e la proprietà (18) implica che $g_1(u_1, u_2) = g_2(u_1, u_2)$. Per provare che $(g_1(u_1, u_2), g_2(u_1, u_2)) \in S$ dobbiamo verificare che $h(g_1(u_1, u_2)(d_1, d_2)) = g_1(u_1, u_2)(d_1, h(d_2))$ questo per ogni $(d_1, d_2) \in D \times D$.

Ma:

$$\begin{aligned}
 h(g_1(u_1, u_2)(d_1, d_2)) &= \begin{cases} h(d_2) & \text{se } p(d_1) = \text{true} \\ h(h(u_1(k(d_1), d_2))) & \text{se } p(d_1) = \text{false} \\ h(\perp) & \text{se } p(d_1) = \perp \end{cases} \\
 g_1(u_1, u_2)(d_1, h(d_2)) &= \begin{cases} h(d_2) & \text{se } p(d_1) = \text{true} \\ h(u_1(k(d_1), h(d_2))) & \text{se } p(d_1) = \text{false} \\ \perp & \text{se } p(d_1) = \perp \end{cases}
 \end{aligned}$$

Avremo così che $h(h(u_1(k(d_1), d_2))) = h(u_1(k(d_1), h(d_2)))$ per la proprietà (18) e siccome $h(\perp) = \perp$, avremo ciò che cercavamo. ■

4. P.C.F. - Programming Computable Function

Il linguaggio P.C.F. è un semplice linguaggio di programmazione che fu usato ampiamente come esempio nello sviluppo della teoria sia della semantica denotazionale che operativa.

La sua sintassi fu introdotta da Dana Scott nel 1969 come parte di “Logic of Computable Functions” e fu studiata come linguaggio di programmazione nel 1977 da Plotkin.

Ora vedremo la sintassi e la semantica operativa di una versione di P.C.F.. Successivamente descriveremo una semantica denotazionale utilizzando domini e funzioni continue.

4.1 Termini e Tipi

Tipi:

$$\tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau$$

Espressioni:

$$M ::= 0 \mid \text{succ}(M) \mid \text{pred}(M) \mid \text{zero}(M) \mid \text{true} \mid \text{false} \mid \text{if } M \text{ then } M \text{ else } M \mid x \mid \text{fn } x : \tau. M \mid M \ M \mid \text{fix}(M)$$

Dove $x \in V$ è un insieme infinito di variabili.

Identifichiamo le espressioni come α -conversioni di variabili vincolate, create dalla forma d'espressione fn : per definizione un termine P.C.F. è una classe α -equivalente di espressioni.

Vediamo il significato delle forme sintattiche precedenti:

- Nat: è il tipo di numeri naturali, 0,1,2,3,... In P.C.F. questi sono generati partendo dal numero 0 e ripetendo l'operatore di successore, $\text{succ}(-)$, il cui significato è quello di aggiungere 1 al suo argomento. L'operatore predecessore, $\text{pred}(-)$, sottrae 1 dal numero naturale strettamente positivo.
- Bool: è il tipo booleano, true e false. L'operazione $\text{zero}(-)$ testa se l'argomento è zero o positivo e ritorna true o false rispettivamente. L'espressione condizionale $\text{if } M_1 \text{ then } M_2 \text{ else } M_3$ si comporta come M_2 oppure M_3 a seconda che M_1 sia vero o falso.
- Una variabile P.C.F. x , è una espressione sconosciuta. P.C.F. è un linguaggio funzionale puro, non ci sono stati che variano durante la valutazione dell'espressione ed in particolare le variabili sono identificate come espressioni fisse, piuttosto che variabili di programma il cui contenuto potrebbe cambiare durante la loro valutazione.
- $\tau \rightarrow \tau'$ è il tipo di funzione parziale che dato un singolo argomento di tipo τ restituisce, se possibile, un risultato di tipo τ' . Mentre $\text{fn } x : \tau. M$ è la notazione per le funzioni astratte; è da notare che il tipo τ delle variabili astratte x sono dati esplicitamente. L'applicazione della funzione M_1 all'argomento M_2 è indicato da $M_1 \ M_2$. Solitamente, lo scopo di una funzione astratta estende alla destra del punto dove possibile e le funzioni associate alla sinistra, per esempio $M_1 \ M_2 \ M_3$ significa $(M_1 \ M_2) M_3$ e non $M_1 (M_2 \ M_3)$.
- L'espressione $\text{fix}(M)$ indica un elemento x ricorsivamente definito da $x = Mx$. Il lambda - calcolo equivalente è YM , dove Y è una combinazione di punti fissi.

4.2 Variabili libere, variabili vincolate e sostituzioni

Il linguaggio P.C.F. contiene una variabile legata: libere occorrenze di x in M vanno al limite in $\text{fn } x : \tau.M$. L'insieme finito delle variabili libere di un'espressione M , $\text{fv}(M)$, è definito per induzione sulla sua struttura:

- $\text{fv}(0) = \text{fv}(\text{true}) = \text{fv}(\text{false}) \stackrel{\text{def}}{=} \emptyset$
- $\text{fv}(\text{succ}(M)) = \text{fv}(\text{pred}(M)) = \text{fv}(\text{zero}(M)) = \text{fv}(\text{fix}(M)) \stackrel{\text{def}}{=} \text{fv}(M)$
- $\text{fv}(\text{if } M \text{ then } M' \text{ else } M'') \stackrel{\text{def}}{=} \text{fv}(M) \cup \text{fv}(M') \cup \text{fv}(M'')$
- $\text{fv}(MM') \stackrel{\text{def}}{=} \text{fv}(M) \cup \text{fv}(M')$
- $\text{fv}(x) \stackrel{\text{def}}{=} \{x\}$
- $\text{fv}(\text{fn } x : \tau.M) \stackrel{\text{def}}{=} \{x' \in \text{fv}(M) \mid x' \neq x\}$

Si dice che M è chiuso se $\text{fv}(M) = \emptyset$, aperto altrimenti.

Come visto in precedenza, le espressioni P.C.F. α -convertibili sono quelle che differiscono solo per i nomi delle loro variabili vincolate. Così un termine P.C.F. è una classe equivalente di espressioni P.C.F. per le relazioni di equivalenza di α -conversioni. Comunque, ci si riferisce al termine come ad una espressione rappresentativa, solitamente scegliendone una in cui le variabili vincolate sono tutte distinte tra loro e da qualsiasi altra variabile del contesto in cui i termini sono usati.

L'operazione di sostituzione di un termine M per tutte le occorrenze libere di una variabile x in un termine M' si scrivono:

$$M'[M/x]$$

L'operazione viene eseguita attraverso la sostituzione testuale di un'espressione rappresentante M per ogni occorrenza libera di x in un'espressione rappresentante M' le cui variabili vincolate sono distinte dalle variabili libere di M .

4.3 Tipi

P.C.F. è un linguaggio tipato: i tipi sono assegnati ai termini attraverso relazioni che significano: se $\forall x \in \text{dom}(\Gamma)$ ha tipo $\Gamma(x)$ allora M ha tipo τ .

Vediamo:

- Γ è un tipo di ambiente, per esempio una funzione parziale finita che mappa le variabili ai tipi
- M è un termine
- τ è un tipo

La relazione è induttivamente definita dagli assiomi e dalle regole che vedremo.

$M : \tau$ significa che M è chiuso e che $\emptyset \vdash M : \tau$.

$$\text{P.C.F.}_\tau \stackrel{\text{def}}{=} \{M \mid M : \tau\}$$

1. $\Gamma \vdash 0 : nat$
2. $\frac{\Gamma \vdash M : nat}{\Gamma \vdash succ(M) : nat}$
3. $\frac{\Gamma \vdash M : nat}{\Gamma \vdash pred(M) : nat}$
4. $\frac{\Gamma \vdash M : nat}{\Gamma \vdash zero(M) : nat}$
5. $\Gamma \vdash b : bool \quad (b = true, false)$
6. $\frac{\Gamma \vdash M_1 : bool \quad \Gamma \vdash M_2 : \tau \quad \Gamma \vdash M_3 : \tau}{\Gamma \vdash if \ M_1 \ then \ M_2 \ else \ M_3 : \tau}$
7. $\Gamma \vdash x : \tau \quad if \ x \in dom(\Gamma) \ \& \ \Gamma(x) = \tau$
8. $\frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{\Gamma \vdash fn \ x : \tau. M : \tau \rightarrow \tau'} \quad if \ x \notin dom(\Gamma)$
9. $\frac{\Gamma \vdash M_1 : \tau \rightarrow \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 M_2 : \tau'}$
10. $\frac{\Gamma \vdash M : \tau \rightarrow \tau'}{\Gamma \vdash fix(M) : \tau}$

Nella regola n°8, $\Gamma[x \mapsto \tau]$ denota il tipo di ambiente interpretando x con τ .

Proposizione 14:

- i. Se $\Gamma \vdash M : \tau$, allora $(M) \subseteq dom(\Gamma)$. Se entrambi $\Gamma \vdash M : \tau$ e $\Gamma \vdash M : \tau'$ sono vere, allora $\tau = \tau'$. In particolare un termine chiuso ha al più un tipo.
- ii. Se $\Gamma \vdash M : \tau$ e $\Gamma[x \mapsto \tau] \vdash M' : \tau'$ sono entrambi veri, allora avremo che $\Gamma \vdash M' \left[\frac{M}{x} \right] : \tau'$

Dimostrazione:

Queste proprietà definite induttivamente sono provate facilmente secondo le regole induzione. Il fatto che un termine possiede al più un tipo per un'assegnazione data di tipi alle sue variabili libere conta per il fatto che i tipi di variabili vincolate sono fornite esplicitamente nelle funzioni astratte. ■

Esempio (Funzioni Parziali Ricorsive di tipo P.C.F.):

Benché la sintassi PCF sia piuttosto succinta, la combinazione di incremento, decremento, test dello zero, condizioni, funzioni astratte e applicazioni, punti fissi ricorsivi, forniscono ad esso espressività di Turing, nel senso che tutte le funzioni parziali ricorsive possono essere codificate.

Per esempio la funzione parziale $h : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ definita attraverso la ricorsione primitiva da $f : \mathbb{N} \rightarrow \mathbb{N}$ e $g : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ soddisfa che per ogni $x, y \in \mathbb{N}$:

$$\begin{cases} h(x, 0) &= f(x) \\ h(x, y+1) &= g(x, y, h(x, y)) \end{cases}$$

Così se la funzione f viene codificata in P.C.F. dal termine $F : nat \rightarrow nat$ e g dal termine $G : nat \rightarrow (nat \rightarrow (nat \rightarrow nat))$, allora h può essere codificata nel seguente modo:

$$H \stackrel{def}{=} \text{fix} \left(\lambda h : nat \rightarrow (nat \rightarrow nat) \lambda x : nat \lambda y : nat \text{ if } \text{zero}(y) \text{ then } Fx \text{ else } Gx \text{ pred}(y) (hx \text{ pred}(y)) \right)$$

Separatamente dalle ritorsioni primitive, gli altri costrutti per definire le funzioni ricorsive parziali sono minimizzati. Per esempio, la funzione parziale $m : \mathbb{N} \rightarrow \mathbb{N}$ definita da $k : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ attraverso la minimizzazione che per ogni $x \in \mathbb{N}$:

$$m(x) = \text{il più piccolo } y \geq 0 \text{ tale che } k(x, y) = 0 \text{ e } \forall z \ 0 \leq z \leq y \Rightarrow k(x, z) > 0$$

Ciò può anche essere espresso utilizzando la teoria sui punti fissi, sebbene non sia facile come con la ricorsione primitiva.

Se k fosse codificato in P.C.F. dal termine $K : nat \rightarrow (nat \rightarrow nat)$, allora il fattore m potrebbe essere codificato come $\lambda x : nat \ M'x0$ dove:

$$M' \stackrel{def}{=} \text{fix} \left(\lambda m' : nat \rightarrow (nat \rightarrow nat) \lambda x : nat \lambda y : nat \text{ if } \text{zero}(Kxy) \text{ then } y \text{ else } m'x \text{ succ}(y) \right)$$

4.4 Valutazioni

Diamo la semantica operativa del linguaggio P.C.F. in termini di una relazione definita induttivamente di valutazione la cui forma vedremo successivamente. Il risultato delle valutazioni sono termini P.C.F. di una particolare forma, chiamati *valori* oppure *forme canoniche*. I soli valori di tipo `bool` sono `true` o `false`. I valori di tipo `nat` sono unarie rappresentazioni di numeri naturali, $\text{succ}^n(0)$ ($n \in \mathbb{N}$), dove abbiamo che:

$$\begin{cases} \text{succ}^0(0) \stackrel{def}{=} 0 \\ \text{succ}^{n+1}(0) \stackrel{def}{=} \text{succ}(\text{succ}^n(0)) \end{cases}$$

Valori a tipi di funzione, essendo astrazioni di funzione $\lambda x : \tau. M$, sono più “intenzionali” di quelli di tipi di dati arrotondati, finché M è un termine P.C.F. non valutabile.

4.4.1 Relazione di Valutazione P.C.F.

Data la forma:

$$M \Downarrow_{\tau} V$$

dove:

- τ è un tipo P.C.F.
- $M, V \in PCF_{\tau}$ sono termini chiusi P.C.F. di tipo τ
- V è un valore, allora:

$$V ::= 0 \mid succ(V) \mid true \mid false \mid fn\ x : \tau . M$$

Vediamo ora gli assiomi e le regole per generare la valutazione delle relazioni:

$$\begin{array}{lcl}
 (\Downarrow_{val}) & & V \Downarrow_{\tau} V \\
 (\Downarrow_{succ}) & & \frac{M \Downarrow_{nat} V}{succ(M) \Downarrow_{nat} succ(V)} \\
 (\Downarrow_{pred}) & & \frac{M \Downarrow_{nat} succ(V)}{pred(M) \Downarrow_{nat} V} \\
 (\Downarrow_{zero1}) & & \frac{M \Downarrow_{nat} 0}{zero(M) \Downarrow_{bool} true} \\
 (\Downarrow_{zero2}) & & \frac{M \Downarrow_{nat} succ(V)}{zero(M) \Downarrow_{bool} false} \\
 (\Downarrow_{if1}) & & \frac{M_1 \Downarrow_{bool} true \quad M_2 \Downarrow_{\tau} V}{if\ M_1\ then\ M_2\ else\ M_3 \Downarrow_{\tau} V} \\
 (\Downarrow_{if2}) & & \frac{M_1 \Downarrow_{bool} false \quad M_3 \Downarrow_{\tau} V}{if\ M_1\ then\ M_2\ else\ M_3 \Downarrow_{\tau} V} \\
 (\Downarrow_{cbn}) & & \frac{M_1 \Downarrow_{\tau \rightarrow \tau'} fn\ x : \tau . M'_1 \quad M'_1[M_2/x] \Downarrow_{\tau'} V}{M_1 M_2 \Downarrow_{\tau'} V} \\
 (\Downarrow_{fix}) & & \frac{M\ fix(M) \Downarrow_{\tau} V}{fix(M) \Downarrow_{\tau} V}
 \end{array}$$

Proposizione 15:

La valutazione in P.C.F. è deterministica: se $M \Downarrow_{\tau} V$ e $M \Downarrow_{\tau} V'$ sono entrambe vere, allora $V = V'$.

Dimostrazione:

Per le regole di induzione, notiamo che:

$$\{(M, \tau, V) \mid M \Downarrow_{\tau} V \ \& \ \forall V' (M \Downarrow_{\tau} V' \Rightarrow V = V')\}$$

ed è chiuso per gli assiomi e le regole definite con l'operatore \Downarrow .

In particolare avremo che:

$$\begin{array}{c}
 0 \Downarrow 0 \\
 true \Downarrow true \\
 false \Downarrow false \\
 fn\ x.M \Downarrow fn\ x.M
 \end{array}$$

Esempio:

La precedente proposizione fa vedere che ogni termine chiuso è valutato da al più un valore. Sicuramente ci sono alcuni termini che non possono essere valutati.

Scriviamo $M \not\Downarrow_\tau$ se e solo se $M : \tau$ e $\nexists V. M \Downarrow_\tau V$.

Per esempio:

$$\Omega_\tau \stackrel{\text{def}}{=} \text{fix}(fn\ x : \tau.x) \quad (\text{Punto fisso dell'identità})$$

questa soddisfa $\Omega_\tau \not\Downarrow_\tau$.

Per qualche V si può dimostrare che $\text{fix}(fn\ x : \tau.x) \Downarrow_\tau V$, scegliendone uno di peso minimo, che chiameremo \wp .

Allora:

$$\frac{\frac{\frac{fn\ x : \tau.x \Downarrow fn\ x : \tau.x}{fn\ x : \tau.x \Downarrow fn\ x : \tau.x} (\Downarrow_{val}) \quad \text{fix}(fn\ x : \tau.x) \Downarrow V}{(fn\ x : \tau.x)(\text{fix}(fn\ x : \tau.x)) \Downarrow V} (\Downarrow_{cbn})}{\text{fix}(fn\ x : \tau.x) \Downarrow V} (\Downarrow_{fix})$$

Dove \wp' è la dimostrazione più corta di $\text{fix}(fn\ x : \tau.x) \Downarrow_\tau V$, che contraddice la minimalità di \wp .

4.5 Equivalenza contestuale contro l'uguaglianza nella denotazione

Gli assiomi e le regole viste in precedenza, le riscriviamo in modo semantico:

$$\text{zero}(0) \rightarrow_{bool} true$$

$$\text{pred}(\text{succ}(V)) \rightarrow_{nat} V \quad (\text{dove } V \text{ è un valore di tipo } nat)$$

$$\text{zero}(\text{succ}(V)) \rightarrow_{bool} false \quad (\text{dove } V \text{ è un valore di tipo } nat)$$

$$\frac{M \rightarrow_{nat} M'}{op(M) \rightarrow_\tau op(M')} \quad (\text{dove } op = succ, pred \& \tau = nat, \text{ or } op = zero \& \tau = bool)$$

$$\frac{M_1 \rightarrow_{bool} M'_1}{if\ M_1\ then\ M_2\ else\ M_3 \rightarrow_\tau if\ M'_1\ then\ M_2\ else\ M_3}$$

$$if\ true\ then\ M_1\ else\ M_2 \rightarrow_\tau M_1$$

$$if\ false\ then\ M_1\ else\ M_2 \rightarrow_\tau M_2$$

$$\frac{M_1 \rightarrow_{\tau \rightarrow \tau'} M'_1}{M_1 M_2 \rightarrow_{\tau'} M'_1 M_2}$$

$$(fn x : \tau. M_1) M_2 \rightarrow_{\tau} M_1 [M_2/x]$$

$$fix(M) \rightarrow_{\tau} M \ fix(M)$$

4.5.1 Semantica Denotazionale del linguaggio P.C.F.

Vogliamo costruire la semantica denotazionale di M :

- P.C.F. di tipo $\tau \mapsto$ domini $\llbracket \tau \rrbracket$.
- Termini P.C.F. chiusi $M : \tau \mapsto$ elementi $\llbracket M \rrbracket \in \llbracket \tau \rrbracket$. Più genericamente le denotazioni di termini aperti saranno funzioni continue.
- Composizionalità: $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow \llbracket C[M] \rrbracket = \llbracket C[M'] \rrbracket$,
con $C ::= - \mid 0 \mid succ(C) \mid pred(C) \mid if\ C\ then\ C\ else\ C \mid \dots$
- Correttezza: $\forall\ tipo\ \tau, M \Downarrow_{\tau} V \Rightarrow \llbracket M \rrbracket = \llbracket V \rrbracket$.
- Adeguatezza: $\tau = bool\ o\ nat$, $\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket \Rightarrow M \Downarrow_{\tau} V$

Le proprietà di correttezza e di adeguatezza forniscono una connessione tra le semantiche operazionali e denotazionali. Si noti che la proprietà di adeguatezza implica solo i tipi di dati *nat* o *bool*. Uno non può aspettarsi che una tale proprietà tenga a tipi di funzione a causa della natura intenzionale di valori a tali tipi. Infatti la proprietà dell'adeguatezza a tipi di funzione contraddirebbe le proprietà di composizionalità e di correttezza che si vuole per $\llbracket - \rrbracket$, come si vede nel prossimo esempio.

Esempio:

Consideriamo i seguenti due termini del valore P.C.F. di tipo $nat \rightarrow nat$:

$$\begin{aligned} V &\stackrel{def}{=} fn\ x : nat. (fn\ y : nat. y) 0 \\ V' &\stackrel{def}{=} fn\ x : nat. 0 \end{aligned}$$

Ora $V \not\Downarrow_{nat \rightarrow nat} V'$, utilizzando l'assioma (\Downarrow_{val}) , $V \Downarrow_{nat \rightarrow nat} V \neq V'$ e per la ultima proposizione vista si ha che la valutazione è deterministica. Tuttavia le proprietà di correttezza e composizionalità di $\llbracket - \rrbracket$ implica che $\llbracket V \rrbracket = \llbracket V' \rrbracket$.

Utilizzando gli assiomi (\Downarrow_{val}) e (\Downarrow_{cbn}) abbiamo che :

$$(fn\ y : nat. y) 0 \Downarrow_{nat} 0$$

Così per la proprietà di correttezza:

$$\llbracket (fn\ y : nat. y) 0 \rrbracket = \llbracket 0 \rrbracket$$

Quindi per la proprietà della composizionalità, siccome $C[-] \stackrel{def}{=} fn\ x : nat. -$, abbiamo che:

$$\llbracket C[(\text{fn } y : \text{nat}.y)0] \rrbracket = \llbracket C[0] \rrbracket$$

Di conseguenza $\llbracket V \rrbracket = \llbracket V' \rrbracket$.

Definizione 22:

Nel precedente esempio la notazione $C[M]$ indica un termine P.C.F. contenente occorrenze di un termine M , e $C[M']$ è il termine che risulta dalla sostituzione di queste occorrenze con M' . Più precisamente, i contesti P.C.F. sono generati attraverso la grammatica per le espressioni P.C.F. aumentate dal simbolo ‘-’ rappresentante un posto o un buco che può essere riempito con un termine P.C.F.:

$$C ::= - \mid 0 \mid \text{succ}(C) \mid \text{pred}(C) \mid \text{zero}(C) \mid \text{true} \mid \text{false} \mid \text{if } C \text{ then } C \text{ else } C \mid x \mid \text{fn } x : \tau. C \mid CC \mid \text{fix}(C)$$

Dato un tale contesto C , scriviamo $C[M]$ per l’espressione P.C.F. che risulta dalla sostituzione di tutte le occorrenze di ‘-’ in C attraverso M . Questa forma di sostituzione può implicare la cattura delle variabili libere in M vincolate in C . Per esempio se C è $\text{fn } x : \tau. -$, allora $C[x]$ è $\text{fn } x : \tau. x$.

Tuttavia è possibile vedere che se M e M' sono α -convertibili, lo sono anche $C[M]$ e $C[M']$. Perciò l’operazione sulle espressioni P.C.F. mandando M in $C[M]$ induce un’operazione ben definita sui termini P.C.F. (classi α -equivalenti di espressioni).

4.5.2 Equivalenza Contestuale

Due frasi di un linguaggio di programmazione sono contestualmente equivalenti se ogni occorrenza della prima frase in un programma completo può essere sostituita dalla seconda frase senza influire in maniera visibile sui risultati dell’esecuzione del programma.

Questa è la nozione generale dell’equivalenza contestuale delle frasi di un linguaggio di programmazione. E’ una nozione parametrizzata dalle particolari scelte che si possono prendere per come è costruito un programma nel linguaggio e cosa sono i risultati visibili dell’esecuzione dei programmi. Per i linguaggi di tipo P.C.F. è ragionevole dare ai programmi termini chiusi del tipo *nat* o *bool* e osservare i valori che si avranno dalla valutazioni di tali termini.

Questi comandi sono dati dalla seguente definizione.

4.5.3 Equivalenza contestuale dei termini di tipo P.C.F.

Dati dei termini di tipo P.C.F. M_1, M_2 , un P.C.F. di tipo τ e un ambiente di tipo Γ , la relazione $\Gamma \vdash M_1 \cong_{\text{ctx}} M_2 : \tau$ è definita se e solo se:

- Entrambi i tipi $\Gamma \vdash M_1 : \tau$ e $\Gamma \vdash M_2 : \tau$ sono validi
- Per ogni contesto di tipo P.C.F. C per cui $C[M_1]$ e $C[M_2]$ sono termini chiusi di tipo γ , dove $\gamma = \text{nat}$ oppure $\gamma = \text{bool}$, e per tutti i valori $V : \gamma$,

$$C[M_1] \Downarrow_{\gamma} V \Leftrightarrow C[M_2] \Downarrow_{\gamma} V$$

Nota:

Per i termini chiusi di tipo P.C.F., scriviamo :

$$\begin{array}{c} M_1 \cong_{ctx} M_2 : \tau \\ \text{per } 0 \vdash M_1 \cong_{ctx} M_2 : \tau \end{array}$$

Sebbene \cong_{ctx} è una nozione naturale dell'equivalenza semantica per i P.C.F. date dalle semantiche operazionali, è difficile lavorarci a causa della quantificazione universale su contesti che occorrono nella definizione.

Come vedremo con il prossimo teorema, se abbiamo delle semantiche denotazionali di tipo P.C.F. che soddisfano le proprietà di chiusura, composizionalità, correttezza e adeguatezza viste in precedenza, potremmo usarle per stabilire equivalenze contestuali facendo notare che i termini hanno denotazioni identiche.

In alcuni casi è un passaggio facile provare l'equivalenza contestuale direttamente dalla definizione. Nel prossimo teorema vedremo che se le funzioni continue che sono le denotazioni di due termini aperti sono uguali, allora i termini sono contestualmente equivalenti.

Teorema 5:

Per ogni tipo τ e termine chiuso $M_1, M_2 \in PCF_\tau$, se $\llbracket M_1 \rrbracket$ e $\llbracket M_2 \rrbracket$ sono elementi uguali del dominio $\llbracket \tau \rrbracket$, allora $M_1 \cong_{ctx} M_2 : \tau$.

Dimostrazione:

$$\begin{array}{ll} C[M_1] \Downarrow_{nat} V \Rightarrow \llbracket C[M_1] \rrbracket = \llbracket V \rrbracket & \text{(correttezza)} \\ \Rightarrow \llbracket C[M_2] \rrbracket = \llbracket V \rrbracket & \text{(composizionalità su } \llbracket M_1 \rrbracket = \llbracket M_2 \rrbracket \text{)} \\ \Rightarrow C[M_2] \Downarrow_{nat} V & \text{(adeguatezza)} \end{array}$$

e simmetricamente.

5. Semantiche denotazionali P.C.F.

5.1 Tipi di Denotazioni

Per ogni P.C.F. di tipo τ , definiamo un dominio $\llbracket \tau \rrbracket$ attraverso l'induzione sulla struttura di τ descritta qui di seguito:

- $\llbracket nat \rrbracket \stackrel{def}{=} \mathbb{N}_\perp$ dominio piatto
- $\llbracket bool \rrbracket \stackrel{def}{=} B_\perp$ dominio piatto
- $\llbracket \tau \rightarrow \tau' \rrbracket \stackrel{def}{=} \llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket$ dominio di funzione

dove $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ e $B = \{true, false\}$

5.2 Denotazione di termini

Per ogni termine P.C.F. M e tipo di ambiente Γ , utilizzando la proposizione sui tipi del linguaggio P.C.F. visto in precedenza, c'è al più un tipo τ per cui la relazione $\Gamma \vdash M : \tau$ è derivabile dagli assiomi e regole sui tipi. Forniamo solo una semantica denotazionale ai termini scrivibili. Specificatamente per M e Γ , definiamo una funzione continua tra i domini :

$$\llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket \quad (19)$$

dove τ è un tipo per cui $\Gamma \vdash M : \tau$ è valido, e dove $\llbracket \Gamma \rrbracket$ è la conseguente dipendenza del prodotto tra domini:

$$\llbracket \Gamma \rrbracket \stackrel{def}{=} \prod_{x \in dom(\Gamma)} \llbracket \Gamma(x) \rrbracket \quad (20)$$

Gli elementi del dominio (20) saranno chiamati Γ -ambienti: ci sono funzioni ρ che trasformano ogni variabile x nel dominio di definizione di Γ verso un elemento $\rho(x) \in \llbracket \Gamma(x) \rrbracket$ nel dominio che è la denotazione di tipo $\Gamma(x)$ assegnato da x attraverso l'ambiente Γ . La funzione continua (19) è definita per induzione sulla struttura di M , o equivalentemente per induzione sulla derivazione delle relazioni di tipo $\Gamma \vdash M : \tau$.

Le definizioni che vedremo riguardano le funzioni ρ su Γ -ambienti.

5.2.1 Semantiche Denotazionali di termini P.C.F.

- $\llbracket \Gamma \vdash 0 \rrbracket(\rho) \stackrel{def}{=} 0 \in \llbracket nat \rrbracket$
- $\llbracket \Gamma \vdash true \rrbracket(\rho) \stackrel{def}{=} true \in \llbracket bool \rrbracket$
- $\llbracket \Gamma \vdash false \rrbracket(\rho) \stackrel{def}{=} false \in \llbracket bool \rrbracket$

- $\llbracket \Gamma \vdash x \rrbracket(\rho) \stackrel{def}{=} \rho(x) \in \llbracket \Gamma(x) \rrbracket \quad (x \in \text{dom}(\Gamma))$
- $\llbracket \Gamma \vdash \text{succ}(M) \rrbracket(\rho) \stackrel{def}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) + 1 & \text{se } \llbracket \Gamma \vdash \text{succ}(M) \rrbracket(\rho) \neq 1 \\ \perp & \text{se } \llbracket \Gamma \vdash \text{succ}(M) \rrbracket(\rho) = 1 \end{cases}$
- $\llbracket \Gamma \vdash \text{pred}(M) \rrbracket(\rho) \stackrel{def}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) - 1 & \text{se } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\ \perp & \text{se } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0, \perp \end{cases}$
- $\llbracket \Gamma \vdash \text{zero}(M) \rrbracket(\rho) \stackrel{def}{=} \begin{cases} \text{true} & \text{se } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0 \\ \text{false} & \text{se } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\ \perp & \text{se } \llbracket \Gamma \vdash M \rrbracket(\rho) = \perp \end{cases}$
- $\llbracket \Gamma \vdash \text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rrbracket(\rho) \stackrel{def}{=} \begin{cases} \llbracket \Gamma \vdash M_2 \rrbracket(\rho) & \text{se } \llbracket \Gamma \vdash M_1 \rrbracket(\rho) = \text{true} \\ \llbracket \Gamma \vdash M_3 \rrbracket(\rho) & \text{se } \llbracket \Gamma \vdash M_1 \rrbracket(\rho) = \text{false} \\ \perp & \text{se } \llbracket \Gamma \vdash M_1 \rrbracket(\rho) = \perp \end{cases}$
- $\llbracket \Gamma \vdash M_1 M_2 \rrbracket(\rho) \stackrel{def}{=} (\llbracket \Gamma \vdash M_1 \rrbracket(\rho))(\llbracket \Gamma \vdash M_2 \rrbracket(\rho))$
- $\llbracket \Gamma \vdash \text{fn } x : \tau. M \rrbracket(\rho) \stackrel{def}{=} \lambda d \in \llbracket \tau \rrbracket. \llbracket \Gamma[x \mapsto \tau] \vdash M \rrbracket(\rho[x \mapsto d]) \quad \text{dove } x \notin \text{dom}(\Gamma)$
- $\llbracket \Gamma \vdash \text{fix}(M) \rrbracket(\rho) \stackrel{def}{=} \text{fix}(\llbracket \Gamma \vdash M \rrbracket(\rho))$
- Se $M \in PCF_\tau$, per definizione $0 \vdash M : \tau$ è vera, allora $\llbracket 0 \vdash M \rrbracket : \llbracket 0 \rrbracket \rightarrow \llbracket \tau \rrbracket$. Quando $\Gamma = 0$, il solo Γ -ambiente è la funzione parziale totalmente indefinita chiamata \perp . In questo caso $\llbracket \Gamma \rrbracket$ è un dominio con un unico elemento, cioè $\{\perp\}$. Le funzioni continue $f : \{\perp\} \rightarrow D$ sono in corrispondenza con gli elementi $f(\perp) \in D$, ed in particolare possiamo identificare la denotazione dei termini P.C.F. chiusi con gli elementi del dominio indicanti il loro tipo:

$$\llbracket M \rrbracket \stackrel{def}{=} \llbracket 0 \vdash M \rrbracket(\perp) \in \llbracket \tau \rrbracket \quad (M \in PCF_\tau)$$

$\llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$ è una funzione continua ben definita siccome i casi di base delle definizioni viste prima sono funzioni continue e ad ogni passo induttivo, data la denotazione di una frase composta in termini di denotazioni delle sue sottofrasi, possiamo utilizzare i costrutti preservando la continuità, come indicato.

- ❖ 0, true, false : La denotazione di questi termini sono tutte le funzioni che sono costantemente uguali ad un particolare valore. Esempio: dati D ed E di tipo c.p.o., per ogni $e \in E$ è facile vedere che la funzione costante $D \rightarrow E$ con valore e , $\forall d \in D. e$, è continua.
- ❖ Variabili: La denotazione di una variabile è una proiezione di una funzione. Abbiamo visto in passato che tali funzioni sono continue, siccome i minimi estremi superiori sono trovati in domini di prodotto dipendenti.

5.2.2 La Composizione preserva la Continuità

Proposizione 16:

Se $f : D \rightarrow E$ e $g : E \rightarrow F$ sono funzioni continue di tipo c.p.o., allora la loro composizione:

$$g \circ f : D \rightarrow F$$

$$(g \circ f)(d) \stackrel{\text{def}}{=} g(f(d))$$

ed è continua.

- ❖ Succ, pred, zero: Abbiamo bisogno di fare uso del fatto che la composizione di funzioni preserva la continuità. In particolare la denotazione $\text{succ}(M)$ è la composizione di:

$$s_{\perp} \circ \llbracket \Gamma \vdash M \rrbracket$$

dove per ipotesi di induzione $\llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \mathbb{N}_{\perp}$ è una funzione continua, e dove $s_{\perp} : \mathbb{N}_{\perp} \rightarrow \mathbb{N}_{\perp}$ è la funzione continua sul dominio piatto \mathbb{N}_{\perp} , attraverso la funzione $s : \mathbb{N} \rightarrow \mathbb{N}$ interpretando ogni n in $n+1$. Allo stesso modo $\llbracket \Gamma \vdash \text{pred}(M) \rrbracket = p_{\perp} \circ \llbracket \Gamma \vdash M \rrbracket$ e $\llbracket \Gamma \vdash \text{zero}(M) \rrbracket = z_{\perp} \circ \llbracket \Gamma \vdash M \rrbracket$ per le funzioni adatte $p : \mathbb{N} \rightarrow \mathbb{N}$ e $z : \mathbb{N} \rightarrow \mathbb{B}$. (Solo p è una funzione parziale propria, indefinita allo 0; s e z sono funzioni totalmente definite.)

- ❖ Condizionale: per ipotesi di induzione abbiamo funzioni continue $\llbracket \Gamma \vdash M_1 \rrbracket : \llbracket \Gamma \rrbracket \rightarrow B_{\perp}$, $\llbracket \Gamma \vdash M_2 \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$, $\llbracket \Gamma \vdash M_3 \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$. Allora $\llbracket \Gamma \vdash \text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rrbracket$ è continua perchè possiamo esprimere la definizione vista in precedenza in termini di composizione, accoppiando l'operazione vista con la proposizione sui prodotti dei domini con la funzione continua $B_{\perp} \times (\llbracket \tau \rrbracket \times \llbracket \tau \rrbracket) \rightarrow \llbracket \tau \rrbracket$:

$$\llbracket \Gamma \vdash \text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rrbracket = \text{if} \circ \langle \llbracket \Gamma \vdash M_1 \rrbracket, \langle \llbracket \Gamma \vdash M_2 \rrbracket, \llbracket \Gamma \vdash M_3 \rrbracket \rangle \rangle.$$

- ❖ Applicazione: per ipotesi di induzione abbiamo le funzioni continue $\llbracket \Gamma \vdash M_1 \rrbracket : \llbracket \Gamma \rrbracket \rightarrow (\llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket)$, e $\llbracket \Gamma \vdash M_2 \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$. Allora $\llbracket \Gamma \vdash M_1 M_2 \rrbracket$ è continua perchè possiamo esprimere la definizione vista in precedenza in termini di composizione accoppiandola con la funzione “evaluation” $ev : (\llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket) \times \llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket$ la quale sappiamo essere continua :

$$\llbracket \Gamma \vdash M_1 M_2 \rrbracket = ev \circ \langle \llbracket \Gamma \vdash M_1 \rrbracket, \llbracket \Gamma \vdash M_2 \rrbracket \rangle$$

- ❖ Funzione astratta: Per ipotesi di induzione abbiamo la funzione continua $\llbracket \Gamma[x \mapsto \tau] \vdash M \rrbracket : \llbracket \Gamma[x \mapsto \tau] \rrbracket \rightarrow \llbracket \tau' \rrbracket$ con $x \notin \text{dom}(\Gamma)$. Da notare che ogni $\llbracket \Gamma[x \mapsto \tau] \rrbracket$ -ambiente, $\rho' \in \llbracket \Gamma[x \mapsto \tau] \rrbracket$, può essere espresso unicamente come $\rho[x \mapsto d]$, dove ρ è la restrizione della funzione ρ' in $\text{dom}(\Gamma)$ e dove $d = \rho'(x)$; inoltre l'ordine parziale rispetta questa decomposizione: $\rho_1[x \mapsto d_1] \sqsubseteq \rho_2[x \mapsto d_2]$ in $\llbracket \Gamma[x \mapsto \tau] \rrbracket$ se e solo se $\rho_1 \sqsubseteq \rho_2$ in $\llbracket \Gamma \rrbracket$ e $d_1 \sqsubseteq d_2$ in $\llbracket \tau \rrbracket$. Così possiamo identificare $\llbracket \Gamma[x \mapsto \tau] \rrbracket$ con il prodotto binario di dominio $\llbracket \Gamma \rrbracket \times \llbracket \tau \rrbracket$. Possiamo applicare l'operazione di “currying” per ottenere una funzione continua :

$$cur(\llbracket \Gamma[x \mapsto \tau] \vdash M \rrbracket) : \llbracket \Gamma \rrbracket \rightarrow (\llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket) = \llbracket \tau \rightarrow \tau' \rrbracket$$

Ma questa è precisamente la funzione utilizzata per definire $\llbracket \Gamma \vdash fn\ x : \tau.M \rrbracket$ nelle semantiche denotazionali.

- ❖ Punti fissi: Per ipotesi di induzione abbiamo la funzione continua $\llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rightarrow \tau \rrbracket$. Ora $\llbracket \tau \rightarrow \tau \rrbracket$ è la funzione dominio $\llbracket \tau \rrbracket \rightarrow \llbracket \tau \rrbracket$ e per la definizione sulle semantiche denotazionali abbiamo che la funzione $fix : (\llbracket \tau \rrbracket \rightarrow \llbracket \tau \rrbracket) \rightarrow \llbracket \tau \rrbracket$ assegna i minimi punti fissi, di cui abbiamo già visto la continuità.

5.3 Composizionalità

Il fatto che le semantiche denotazionali dei termini P.C.F. siano composizionali, per esempio che la denotazione di un termine è la funzione di una denotazione del suo immediato sottotermino, è una parte della definizione di $\llbracket \Gamma \vdash M \rrbracket$ per induzione sulla struttura di M . In particolare, ognuno dei modi di costruzione dei termini in P.C.F. rispetta l'uguaglianza delle denotazioni. Allora la proprietà di chiusura dei termini :

$$\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow \llbracket C[M] \rrbracket = \llbracket C[M'] \rrbracket$$

Segue da questa per induzione sulla struttura del contesto $C[-]$.

Più genericamente per i termini aperti abbiamo:

Proposizione 17:

Supponiamo:

$$\llbracket \Gamma \vdash M \rrbracket = \llbracket \Gamma \vdash M' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$$

e che $C[-]$ è un contesto P.C.F. cosicché $\Gamma' \vdash C[M] : \tau'$ e $\Gamma' \vdash C[M'] : \tau'$ sono vere per qualche tipo τ' e per qualche tipo di ambiente Γ' .

Allora:

$$\llbracket \Gamma' \vdash C[M] \rrbracket = \llbracket \Gamma' \vdash C[M'] \rrbracket : \llbracket \Gamma' \rrbracket \rightarrow \llbracket \tau' \rrbracket$$

5.3.1 Proprietà della sostituzione di $\llbracket - \rrbracket$

Proposizione 18:

Supponiamo

$$\begin{array}{l} \Gamma \vdash M : \tau \\ \Gamma[x \mapsto \tau] \vdash M' : \tau' \end{array}$$

Allora per tutti $\rho \in \llbracket \Gamma \rrbracket$:

$$\llbracket \Gamma \vdash M' [M/x] \rrbracket (\rho) = \llbracket \Gamma [x \mapsto \tau] \vdash M' \rrbracket (\rho [x \mapsto \llbracket \Gamma \vdash M \rrbracket])$$

In particolare quando $\Gamma = 0$, $\llbracket x \mapsto \tau \vdash M' \rrbracket : \llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket$ e soprattutto:

$$\llbracket M' [M/x] \rrbracket = \llbracket x \mapsto \tau \vdash M' \rrbracket (\llbracket M \rrbracket).$$

La proprietà di sostituzione fornisce un ulteriore aspetto della natura composizionale delle semantiche denotazionali di tipo P.C.F.. Ciò può essere provato per induzione sulla struttura del termine M' .

Vediamo ora le proprietà di composizionalità di $\llbracket - \rrbracket$:

- If $\llbracket \Gamma \vdash M \rrbracket = \llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \text{nat}$, then $\llbracket \Gamma \vdash \text{op}(M) \rrbracket = \llbracket \Gamma \vdash \text{op}(M') \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$
(dove $\text{op} = \text{succ}, \text{pred}$ e $\tau = \text{nat}$ oppure $\text{op} = \text{zero}$ e $\tau = \text{bool}$)
- If $\llbracket \Gamma \vdash M_1 \rrbracket = \llbracket \Gamma \vdash M_1' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \text{bool} \rrbracket$, $\llbracket \Gamma \vdash M_2 \rrbracket = \llbracket \Gamma \vdash M_2' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$ and $\llbracket \Gamma \vdash M_3 \rrbracket = \llbracket \Gamma \vdash M_3' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$ then $\llbracket \Gamma \vdash \text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rrbracket = \llbracket \Gamma \vdash \text{if } M_1' \text{ then } M_2' \text{ else } M_3' \rrbracket : \llbracket \Gamma \rrbracket$
- If $\llbracket \Gamma \vdash M_1 \rrbracket = \llbracket \Gamma \vdash M_1' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rightarrow \tau' \rrbracket$ and $\llbracket \Gamma \vdash M_2 \rrbracket = \llbracket \Gamma \vdash M_2' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$ then $\llbracket \Gamma \vdash M_1 M_2 \rrbracket = \llbracket \Gamma \vdash M_1' M_2' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau' \rrbracket$
- If $\llbracket \Gamma [x \mapsto \tau] \vdash M \rrbracket = \llbracket \Gamma [x \mapsto \tau] \vdash M' \rrbracket : \llbracket \Gamma [x \mapsto \tau] \rrbracket \rightarrow \llbracket \tau' \rrbracket$ then $\llbracket \Gamma \vdash \text{fn } x : \tau. M \rrbracket = \llbracket \Gamma \vdash \text{fn } x : \tau. M' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rightarrow \tau' \rrbracket$
- If $\llbracket \Gamma \vdash M \rrbracket = \llbracket \Gamma \vdash M' \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rightarrow \tau' \rrbracket$ then $\llbracket \Gamma \vdash \text{fix}(M) \rrbracket = \llbracket \Gamma \vdash \text{fix}(M') \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$

5.4 Correttezza:

Il secondo degli scopi menzionati precedentemente è far vedere che se un termine chiuso P.C.F. M , valuta a un valore V nella semantica operativa, allora M e V hanno la stessa denotazione.

Teorema 6:

Per ogni tipo τ di tipo P.C.F. e per tutti i termini chiusi $M, V \in P.C.F._\tau$ con V valore, se $M \Downarrow_\tau V$ è derivabile dagli assiomi e regole per la valutazione di P.C.F. allora $\llbracket M \rrbracket$ e $\llbracket V \rrbracket$ sono elementi uguali del dominio $\llbracket \tau \rrbracket$.

Dimostrazione:

Utilizzare la regola di induzione per definire induttivamente la relazione \Downarrow . Specificatamente, definendo:

$$\Phi(M, \tau, V) \stackrel{def}{\Leftrightarrow} \llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket$$

si può notare che $\Phi(M, \tau, V)$ è chiuso per gli assiomi e regole della valutazione di P.C.F..

Caso (\Downarrow_{cbn}) :

Supponiamo

$$\llbracket M_1 \rrbracket = \llbracket \text{fn } x : \tau. M'_1 \rrbracket \in \llbracket \tau \rightarrow \tau' \rrbracket \quad (21)$$

$$\llbracket M'_1[M_2/x] \rrbracket = \llbracket V \rrbracket \in \llbracket \tau' \rrbracket \quad (22)$$

Dobbiamo provare che $\llbracket M_1 M_2 \rrbracket = \llbracket V \rrbracket \in \llbracket \tau' \rrbracket$. Ma :

$$\begin{aligned} \llbracket M_1 M_2 \rrbracket &= \llbracket M_1 \rrbracket(\llbracket M_2 \rrbracket) && \text{Per le regole sulle semantiche operazionali sui termini P.C.F.} \\ &= \llbracket \text{fn } x : \tau. M'_1 \rrbracket(\llbracket M'_2 \rrbracket) && \text{Per la proprietà (21)} \\ &= (\lambda d \in \llbracket \tau \rrbracket. \llbracket x \mapsto \tau \vdash M'_1 \rrbracket(d))(\llbracket M'_2 \rrbracket) && \text{Per le regole sulle semantiche operazionali sui termini P.C.F.} \\ &= \llbracket x \mapsto \tau \vdash M'_1 \rrbracket(\llbracket M'_2 \rrbracket) \\ &= \llbracket M'_1[M_2/x] \rrbracket && \text{Per le regole sulle semantiche operazionali sui termini P.C.F.} \\ &= \llbracket V \rrbracket && \text{Per la proprietà (22)} \end{aligned}$$

Caso (\Downarrow_{fix}) :

Supponiamo

$$\llbracket M \text{ fix } (M) \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket \quad (23)$$

Dobbiamo provare che $\llbracket \text{fix}(M) \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket$. Ma:

$$\begin{aligned} \llbracket \text{fix}(M) \rrbracket &= \text{fix}(\llbracket M \rrbracket) && \text{Per le regole sulle semantiche operazionali sui termini P.C.F.} \\ &= \llbracket M \rrbracket(\text{fix}(\llbracket M \rrbracket)) && \text{Per la proprietà dei punti fissi di fix} \\ &= \llbracket M \rrbracket \llbracket \text{fix}(M) \rrbracket && \text{Per le regole sulle semantiche operazionali sui termini P.C.F.} \\ &= \llbracket M \text{ fix}(M) \rrbracket && \text{Per le regole sulle semantiche operazionali sui termini P.C.F.} \end{aligned}$$

$$= \llbracket V \rrbracket$$

Per la proprietà (23)

Abbiamo dimostrato due delle tre proprietà delle semantiche denotazionali di tipo P.C.F.. La terza proprietà, l'adeguatezza, è più difficile da provare rispetto alle prime due. La dimostrazione sarà effettuata più avanti. ■

6. Relazioni Denotazionali e Semantiche Operazionali

Abbiamo visto che le semantiche denotazionali di tipo P.C.F. sono corrette per le semantiche operazionali. Ora proveremo a definire la proprietà dell'adequatezza. Per ogni termine chiuso di tipo P.C.F. M e V di tipo $\tau = nat$ o $bool$ con V valore, allora :

$$\llbracket M \rrbracket = \llbracket V \rrbracket \Rightarrow M \Downarrow_{\tau} V$$

Sorprendentemente non è facile da provare. Si impiegherà un metodo creato da Plotkin nel 1977 e ripreso da Mulmuley nel 1987 in cui si fa uso della nozione di “approssimazione formale” tra relazioni.

6.1 Approssimazione formale di Relazioni

Definiamo una famiglia di relazioni binarie:

$$\triangleleft_{\tau} \subseteq \llbracket \tau \rrbracket \times PCF_{\tau}$$

indicizzate dal tipo P.C.F. τ . Così ognuno degli elementi \triangleleft_{τ} del dominio $\llbracket \tau \rrbracket$ riporta ad un termine chiuso P.C.F. di tipo τ . Utilizziamo la notazione infissa e scriviamo $d \triangleleft_{\tau} M$ invece di $(d, M) \in \triangleleft_{\tau}$. La definizione di queste relazioni \triangleleft_{τ} si ricava attraverso l'induzione sulla struttura di tipo τ e verrà fornita più avanti.

La proprietà chiave della relazione \triangleleft_{τ} è quello di rispettare le varie operazioni sintattiche del linguaggio P.C.F.. Anche questo verrà affrontato più avanti.

Definizione 23:

Per ogni tipo di ambiente Γ (una funzione parziale finita da variabili a tipi P.C.F.), una Γ -sostituzione σ è una funzione che mappa ogni variabile $x \in dom(\Gamma)$ ad termine chiuso P.C.F. $\sigma(x)$ di tipo $\Gamma(x)$. Ricordiamo che un Γ -ambiente ρ è una funzione che mappa ogni variabile $x \in dom(\Gamma)$ ad un elemento $\rho(x)$ del dominio $\llbracket \Gamma(x) \rrbracket$. Definiamo allora:

$$\rho \triangleleft_{\Gamma} \sigma \stackrel{def}{\Leftrightarrow} \forall x \in dom(\Gamma). \rho(x) \triangleleft_{\Gamma(x)} \sigma(x)$$

Definizione 24: $d \triangleleft_{\tau} M$ ($d \in \llbracket \tau \rrbracket, M \in PCF_{\tau}$)

$$d \triangleleft_{nat} M \stackrel{def}{\Leftrightarrow} (d \in \mathbb{N} \Rightarrow M \Downarrow_{nat} succ^d(0))$$

$$d \triangleleft_{bool} M \stackrel{def}{\Leftrightarrow} (d = true \Rightarrow M \Downarrow_{bool} true) \& (d = false \Rightarrow M \Downarrow_{bool} false)$$

$$d \triangleleft_{\tau \rightarrow \tau'} M \stackrel{def}{\Leftrightarrow} \forall e, N (e \triangleleft_{\tau} N \Rightarrow d(e) \triangleleft_{\tau'} MN)$$

6.1.1 Proprietà fondamentali delle relazioni c:

Proposizione 19:

Se $\Gamma \vdash M : \tau$ è un tipo valido P.C.F., allora per ogni Γ - ambiente ρ e per ogni Γ - sostituzione σ :

$$\rho \triangleleft_{\Gamma} \sigma \Rightarrow \llbracket \Gamma \vdash M \rrbracket(\rho) \triangleleft_{\tau} M[\sigma]$$

Si ricorda che $\rho \triangleleft_{\Gamma} \sigma$ significa che $\rho(x) \triangleleft_{\Gamma(x)} \sigma(x)$ è valido per ogni $x \in \text{dom}(\Gamma)$, ed inoltre che $M[\sigma]$ è il termine P.C.F. risultante dalla simultanea sostituzione di $\sigma(x)$ per x in M , per ogni $x \in \text{dom}(\Gamma)$.

E' da notare che la proprietà fondamentale di \triangleleft_{τ} vista in precedenza, nel caso in cui $\Gamma = 0$ abbiamo che :

$$\llbracket M \rrbracket \triangleleft_{\tau} M$$

per tutti i tipi τ e per tutti i termini chiusi P.C.F. $M : \tau$.

Visto ciò, possiamo completare la dimostrazione della proprietà dell'adeguatezza.

Dimostrazione di $\llbracket M \rrbracket = \llbracket V \rrbracket \Rightarrow M \Downarrow_{\tau} V$ ($\tau = \text{nat}, \text{bool}$)

Caso 1 : $\tau = \text{nat}$

$V = \text{succ}^n(0)$ per qualche $n \in \mathbb{N}$ e perciò

$$\begin{aligned} \llbracket M \rrbracket &= \llbracket \text{succ}^n(0) \rrbracket \\ &\Rightarrow n = \llbracket M \rrbracket \triangleleft_{\tau} M && \text{Per la proprietà fondamentale} \\ &\Rightarrow M \Downarrow_{\text{succ}^n(0)} && \text{Per la definizione di } \triangleleft_{\text{nat}} \end{aligned}$$

Caso 2 : $\tau = \text{bool}$

La dimostrazione è simile al caso 1. ■

6.2 Dimostrazione della proprietà fondamentale di \triangleleft :

Per tale dimostrazione abbiamo bisogno delle seguenti proprietà sull'approssimazione formale delle relazioni.

Lemma 4:

- i. $\perp \triangleleft_{\tau} M$ è vero per ogni $M \in PCF_{\tau}$

- ii. Per ogni $M \in PCF_\tau$, $\{d \mid d \triangleleft_\tau M\}$ è una catena chiusa sottoinsieme del dominio $\llbracket \tau \rrbracket$.
Perciò attraverso la i è un sottoinsieme ammissibile.
- iii. Se $d_2 \sqsubseteq d_1, d_1 \triangleleft_\tau M_1$, e $\forall V (M_1 \Downarrow_\tau V \Rightarrow M_2 \Downarrow_\tau V)$, allora $d_2 \triangleleft_\tau M_2$

Dimostrazione del lemma 4:

Ognuna di queste proprietà segue facilmente per induzione sulla struttura di τ , utilizzando la definizione di \triangleleft_τ e la valutazione della relazione \Downarrow_τ .

Riprendiamo la dimostrazione.

Utilizziamo la regola di induzione per l'induttività definita per tipare la relazione $\Gamma \vdash M : \tau$.
Definiamo :

$$\Phi(\Gamma, M, \tau) \stackrel{def}{\Leftrightarrow} \Gamma \vdash M : \tau \ \& \ \forall \rho, \sigma (\rho \triangleleft_\tau \sigma \Rightarrow \llbracket \Gamma \vdash M \rrbracket(\rho) \triangleleft_\tau M[\sigma])$$

Ora è sufficiente far vedere che Φ è chiuso per gli assiomi e regole fornite per le relazioni di tipo P.C.F.

Caso (\cdot_0) : $\Phi(\Gamma, 0, nat)$ è vero perché $0 \triangleleft_{nat} 0$.

Caso (\cdot_{succ}) : Dobbiamo provare che $\Phi(\Gamma, 0, nat)$ implica $\Phi(\Gamma, succ(M), nat)$. Ma questo è vero verificando facilmente che :

$$d \triangleleft_{nat} M \Rightarrow s_\perp(d) \triangleleft_{nat} succ(M)$$

dove $s_\perp : \mathbb{N}_\perp \rightarrow \mathbb{N}_\perp$ è la funzione continua utilizzata per descrivere la denotazione del termine successore, $succ(M)$.

Caso (\cdot_{pred}) e (\cdot_{zero}) sono simili al precedente.

Caso (\cdot_{bool}) : $\Phi(\Gamma, true, bool)$ è vero perché $true \triangleleft_{bool} true$. La stessa cosa per $\Phi(\Gamma, false, bool)$.

Caso (\cdot_{if}) : E' sufficiente far vedere che se $d_1 \triangleleft_{bool} M_1, d_2 \triangleleft_\tau M_2$ e $d_3 \triangleleft_\tau M_3$, allora:

$$if(d_1, (d_2, d_3)) \triangleleft_\tau if M_1 then M_2 else M_3 \quad (24)$$

dove la if è la funzione continua : $B_\perp \times (\llbracket \tau \rrbracket \times \llbracket \tau \rrbracket) \rightarrow \llbracket \tau \rrbracket$ di una proposizione che fu usata per descrivere la denotazione dei termini condizionali. Se $d_1 = \perp \in B_\perp$, allora $if(d_1, (d_2, d_3)) = \perp$ e la (14) è vera per il punto i del lemma 4. Così possiamo assumere $d_1 \neq \perp$, in cui o $d_1 = true$ oppure $d_1 = false$. Consideriamo il caso $d_1 = true$; per $d_1 = false$ la dimostrazione è simile.

Finché $true = d_1 \triangleleft_{bool} M_1$ per la definizione di \triangleleft_{bool} abbiamo che $M_1 \Downarrow_{bool} true$. Segue dalla regola (\Downarrow_{if1}) che :

$$\forall V (M_2 \Downarrow_{\tau} V \Rightarrow \text{if } M_1 \text{ then } M_2 \text{ else } M_3 \Downarrow_{\tau} V)$$

Così per il punto *iii* del lemma 4 applicato a $d_2 \triangleleft_{\tau} M_2$ produce che :

$$d_2 \triangleleft_{\tau} \text{if } M_1 \text{ then } M_2 \text{ else } M_3$$

ed allora finché $d_2 = \text{if } (true, (d_2, d_3)) = \text{if } (d_1, (d_2, d_3))$ verifichiamo la proprietà (24), come richiesto.

Caso $(:_{var})$: $\Phi(\Gamma, x, \Gamma(x))$ è vero perché se $\rho \triangleleft_{\Gamma} \sigma$ allora per tutti gli $x \in \text{dom}(\Gamma)$ abbiamo che:

$$\llbracket \Gamma \vdash x \rrbracket(\rho) \stackrel{def}{=} \rho(x) \triangleleft_{\Gamma(x)} \sigma(x) \stackrel{def}{=} x[\sigma].$$

Caso $(:_{fn})$: Supponiamo $\Phi(\Gamma[x \mapsto \tau], M, \tau')$ e $\rho \triangleleft_{\Gamma} \sigma$ siano vere. Dobbiamo far vedere che $\llbracket \Gamma \vdash fn\ x : \tau.M \rrbracket(\rho) \triangleleft_{\tau \rightarrow \tau'} (fn\ x : \tau.M)[\sigma]$, per esempio che $d \triangleleft_{\tau} N$ implica:

$$\llbracket \Gamma \vdash fn\ x : \tau.M \rrbracket(\rho)(d) \triangleleft_{\tau'} ((fn\ x : \tau.M)[\sigma])N \quad (25)$$

Per le semantiche denotazionali dei termini P.C.F. abbiamo che :

$$\llbracket \Gamma \vdash fn\ x : \tau.M \rrbracket(\rho)(d) = \llbracket \Gamma[x \mapsto \tau] \vdash M \rrbracket(\rho[x \mapsto d]) \quad (26)$$

Finché $(fn\ x : \tau.M)[\sigma] = fn\ x : \tau.M[\sigma]$ e $(M[\sigma])[N/x] = M[\sigma[x \mapsto N]]$, per la regola $(:_{cbn})$ abbiamo :

$$\forall V (M[\sigma[x \mapsto N]] \Downarrow_{\tau'} V) \Rightarrow ((fn\ x : \tau.M)[\sigma])N \Downarrow_{\tau'} V \quad (27)$$

Finché $\rho \triangleleft_{\Gamma} \sigma$ e $d \triangleleft_{\tau} N$, abbiamo che $\rho[x \mapsto d] \triangleleft_{\Gamma[x \mapsto \tau]} \sigma[x \mapsto N]$; così grazie a $\Phi(\Gamma[x \mapsto \tau], M, \tau')$ abbiamo che :

$$\llbracket \Gamma[x \mapsto \tau] \vdash M \rrbracket(\rho[x \mapsto d]) \triangleleft_{\tau'} M[\sigma[x \mapsto N]]$$

Allora la (25) segue da questa per l'applicazione della proprietà *iii* del lemma 4 alla (26) ed alla (27).

Caso $(:_{app})$: E' sufficiente far vedere che se $d_1 \triangleleft_{\tau \rightarrow \tau'} M_1$ e $d_2 \triangleleft_{\tau} M_2$, allora $d_1(d_2) \triangleleft_{\tau'} M_1 M_2$. Ma questo segue immediatamente dalla definizione di $\triangleleft_{\tau \rightarrow \tau'}$.

Caso $(\cdot)_{fix}$: Supponiamo $\Phi(\Gamma, M, \tau \rightarrow \tau)$ vera. Per alcuni $\rho \triangleleft_\tau \sigma$, dobbiamo provare che:

$$\llbracket \Gamma \vdash fix(M) \rrbracket(\rho) \triangleleft_\tau fix(M)[\sigma] \quad (28)$$

Ricordando le semantiche denotazionali dei termini P.C.F. abbiamo che $\llbracket \Gamma \vdash fix(M) \rrbracket(\rho) = fix(f)$, dove $f \stackrel{def}{=} \llbracket \Gamma \vdash M \rrbracket(\rho)$. Per la proprietà ii del lemma 4:

$$S \stackrel{def}{=} \{d \mid d \triangleleft_\tau fix(M)[\sigma]\}$$

ed è un sottoinsieme ammissibile del dominio $\llbracket \tau \rrbracket$. Così per il Principio di induzione del punto fisso di Scott per verificare la proprietà (28) basta provare :

$$\forall d \in \llbracket \tau \rrbracket (d \in S \Rightarrow f(d) \in S)$$

Quindi finché $\rho \triangleleft_\tau \sigma$, da $\Phi(\Gamma, M, \tau \rightarrow \tau)$ e dalla definizione della f abbiamo $f \triangleleft_{\tau \rightarrow \tau} M[\sigma]$. Così se $d \in S$, per esempio $d \triangleleft_\tau fix(M)[\sigma]$, allora per la definizione di $\triangleleft_{\tau \rightarrow \tau}$, questo è il caso in cui :

$$f(d) \triangleleft_\tau (M[\sigma])(fix(M)[\sigma]) \quad (29)$$

La regola $(\cdot)_{fix}$ implica che:

$$\forall V ((M[\sigma])(fix(M)[\sigma]) \Downarrow_\tau V \Rightarrow fix(M)[\sigma] \Downarrow_\tau V) \quad (30)$$

Allora applicando la proprietà iii del lemma 4 alle proprietà (29) e (30), si avrà $f(d) \triangleleft_\tau fix(M)[\sigma]$, per esempio $f(d) \in S$, come richiesto. ■

6.3 Estensionalità

Il *preordine contestuale* è definito più avanti. Chiaramente :

$$\Gamma \vdash M_1 \cong_{ctx} M_2 : \tau \Leftrightarrow (\Gamma \vdash M_1 \leq_{ctx} M_2 : \tau \ \& \ \Gamma \vdash M_2 \leq_{ctx} M_1 : \tau)$$

Solitamente scriviamo $M_1 \leq_{ctx} M_2 : \tau$ per $0 \vdash M_1 \leq_{ctx} M_2 : \tau$ nel caso che M_1 e M_2 sono termini chiusi.

La relazione formale approssimata \triangleleft_τ attualmente caratterizza il preordine contestuale P.C.F. tra termini chiusi, nel senso che vedremo più avanti.

6.3.1 Preordine Contestuale tra termini P.C.F.

Dati i termini P.C.F. M_1 e M_2 , il tipo P.C.F. τ e il tipo di ambiente Γ , la relazione :

$$\Gamma \vdash M_1 \leq_{ctx} M_2 : \tau$$

è vera se e solo se:

- Entrambi i tipi $\Gamma \vdash M_1 : \tau$ e $\Gamma \vdash M_2 : \tau$ sono veri
- Per ogni contesto P.C.F. C per cui $C[M_1]$ e $C[M_2]$ sono termini chiusi di tipo γ , dove $\gamma = nat$ o $\gamma = bool$, e per ogni valore $V : \gamma$:

$$C[M_1] \Downarrow_{\gamma} V \Rightarrow C[M_2] \Downarrow_{\gamma} V$$

6.3.2 Preordine Contestuale da approssimazioni formali

Proposizione 20:

Per ogni tipo P.C.F. τ e per tutti i termini chiusi $M_1, M_2 \in PCF_{\tau}$: $M_1 \leq_{ctx} M_2 : \tau \Leftrightarrow \llbracket M_1 \rrbracket \triangleleft_{\tau} M_2$.

Dimostrazione:

(\Leftarrow)

Non è difficile far vedere che per i termini chiusi $M_1, M_2 \in PCF_{\tau}$, $M_1 \leq_{ctx} M_2 : \tau$ è vero se e solo se per ogni $M \in PCF_{\tau \rightarrow bool}$:

$$MM_1 \Downarrow_{bool} true \Rightarrow MM_2 \Downarrow_{bool} true$$

Se $\llbracket M_1 \rrbracket \triangleleft_{\tau} M_2$, allora per un qualunque $M \in PCF_{\tau \rightarrow bool}$ per la proprietà fondamentale di \triangleleft abbiamo che $\llbracket M \rrbracket \triangleleft_{\tau \rightarrow bool} M$, la definizione di $\triangleleft_{\tau \rightarrow bool}$ implica che :

$$\llbracket MM_1 \rrbracket = \llbracket M \rrbracket (\llbracket M_1 \rrbracket) \triangleleft_{bool} MM_2 \quad (31)$$

Così se $MM_1 \Downarrow_{bool} true$, allora $\llbracket MM_1 \rrbracket = true$ (per la proprietà della correttezza) e perciò per la definizione di \triangleleft_{bool} per la proprietà (31) abbiamo che $MM_2 \Downarrow_{bool} true$. Così utilizzando la caratterizzazione di \leq_{ctx} menzionato precedentemente, abbiamo $M_1 \leq_{ctx} M_2 : \tau$.

(\Rightarrow)

Basta provare che :

$$(d \triangleleft_{\tau} M_1 \ \& \ M_1 \leq_{ctx} M_2 : \tau) \Rightarrow d \triangleleft_{\tau} M_2 \quad (32)$$

Allora se $M_1 \leq_{ctx} M_2 : \tau$, finché $\llbracket M_1 \rrbracket \triangleleft_\tau M_1$ (per la proprietà fondamentale), la proprietà (32) implica $\llbracket M_1 \rrbracket \triangleleft_\tau M_2$. La proprietà (32) segue dall'induzione sulla struttura del tipo τ , utilizzando la semplice verifica della proprietà di \leq_{ctx} :

- Se $\tau = nat$ o $bool$ allora $M_1 \leq_{ctx} M_2 : \tau$ implica $\forall V : \tau (M_1 \Downarrow_\tau V \Rightarrow M_2 \Downarrow_\tau V)$
- Se $M_1 \leq_{ctx} M_2 : \tau \rightarrow \tau'$ allora $M_1 M \leq_{ctx} M_2 M : \tau'$ per ogni $M : \tau$. ■



La doppia implicazione ci permette di trasferire le proprietà di estensionalità utilizzate nell'ordine parziale di un dominio \sqsubseteq verso il preordine contestuale, come vedremo fra poco.

6.3.3 Proprietà di estensionalità di \leq_{ctx} :

Per $\tau = bool$ o nat , $M_1 \leq_{ctx} M_2 : \tau$ è vera se e solo se:

$$\forall V : \tau (M_1 \Downarrow_\tau V \Rightarrow M_2 \Downarrow_\tau V)$$

Ad una funzione di tipo $\tau \rightarrow \tau'$, $M_1 \leq_{ctx} M_2 : \tau \rightarrow \tau'$ è vera se e solo se

$$\forall M : \tau (M_1 M \leq_{ctx} M_2 M : \tau')$$

Dimostrazione:

Il “solo se” è una conseguenza della definizione di \leq_{ctx} .

Per il “se”, nel caso $\tau = bool$ o nat , abbiamo :

$$\begin{aligned} \llbracket M_1 \rrbracket = \llbracket V \rrbracket &\Rightarrow M_1 \Downarrow_\tau V && \text{per la proprietà dell'adeguatezza} \\ &\Rightarrow M_2 \Downarrow_\tau V && \text{per assunzione} \end{aligned}$$

e quindi $\llbracket M_1 \rrbracket \triangleleft_\tau M_2$ per definizione di \triangleleft su questi tipi. Ora applichiamo la proposizione sul preordine contestuale da approssimazioni formali. Per il “se”, nel caso di una funzione di tipo $\tau \rightarrow \tau'$, abbiamo che:

$$\begin{aligned} d \triangleleft_\tau M &\Rightarrow \llbracket M_1 \rrbracket(d) \triangleleft_{\tau'} M_1 M && \text{finché } \llbracket M_1 \rrbracket \triangleleft_\tau M_1 \\ &\Rightarrow \llbracket M_1 \rrbracket(d) \triangleleft_{\tau'} M_2 M && \text{per la propr. (32), finché } M_1 M \leq_{ctx} M_2 M : \tau' \text{ per} \\ &&& \text{assunzione} \end{aligned}$$

e perciò $\llbracket M_1 \rrbracket \triangleleft_{\tau \rightarrow \tau'} M_2$ per definizione di \triangleleft di tipo $\tau \rightarrow \tau'$. Così ancora una volta possiamo applicare la proposizione sul preordine contestuale da approssimazioni formali per terminare la dimostrazione. ■

7 Astrazione completa

7.1 Fallimento dell'astrazione completa

Come abbiamo visto in passato, la proprietà di adeguatezza implica che l'equivalenza contestuale di due termini P.C.F. può essere provata facendo vedere che hanno uguale denotazione: $\llbracket M_1 \rrbracket = \llbracket M_2 \rrbracket \in \llbracket \tau \rrbracket \Rightarrow M_1 \cong_{ctx} M_2 : \tau$. Sfortunatamente il viceversa è falso: ci sono equivalenze contestuali di termini P.C.F. con denotazioni diverse. In generale si dice che una semantica denotazionale è completamente astratta se l'equivalenza contestuale coincide esattamente con la denotazione. Così la semantica denotazionale di P.C.F. utilizza domini e funzioni continue non valide per essere completamente astratte. Il classico esempio che dimostra questo fallimento è dato da Plotkin (1977) e implica "l'or-parallelo" o "por".

7.1.1 Funzione di OR-Parallelo

La funzione continua $por : B_{\perp} \rightarrow (B_{\perp} \rightarrow B_{\perp})$ è così definita:

por	true	false	\perp
true	true	true	true
false	true	false	\perp
\perp	true	\perp	\perp

La funzione por contrasta con la funzione "or-sequenziale" o "orelse" che ora vedremo. Entrambe le funzioni forniscono la funzione booleana "or" ristrette a $\{true, false\}$, ma differiscono nel loro comportamento negli argomenti che implicano l'elemento \perp , il quale rappresenta la non terminazione. Da notare che $por(d_1, d_2) = true$ se d_1 oppure d_2 sono veri, anche se gli altri argomenti sono \perp ; mentre $orelse(d_1, d_2) = true$ implica che $d_1 \neq \perp$.

7.1.2 Funzione di OR-Sequenziale

La funzione continua $orelse : B_{\perp} \rightarrow (B_{\perp} \rightarrow B_{\perp})$ è così definita:

orelse	true	false	\perp
true	true	true	true
false	true	false	\perp
\perp	\perp	\perp	\perp

è la denotazione del termine P.C.F.:

$$fn x : bool \ fn x' : bool \ if x \ then true \ else x'$$

di tipo $bool \rightarrow (bool \rightarrow bool)$.

La funzione "orelse" può essere definita come P.C.F., nel senso che c'è un termine chiuso P.C.F. $M : bool \rightarrow (bool \rightarrow bool)$ con $\llbracket M \rrbracket = orelse$. Questo termine M esegue il test se il suo primo argomento è *true* o *false*, nel primo caso restituisce *true* e nel secondo caso restituisce il secondo argomento. Per quanto riguarda la funzione "por" non si darà una dimostrazione. Plotkin (1977)

provò a fornirne una attraverso l' "Activity Lemma", ma ci sono approcci alternativi utilizzando funzioni continue più stabili (Gunter, 1992) o utilizzando "relazioni logiche sequenziali" (Sieber, 1992). L'idea chiave è che la valutazione in P.C.F. proceda in modo sequenziale. Cosicché qualunque cosa sia P , la valutazione di PM_1M_2 deve implicare in qualche punto una valutazione completa di M_1 o M_2 (P non può ignorare il suo argomento se restituisce *true* in alcuni casi e *false* in altri); mentre un algoritmo esegue la funzione "por" a una coppia di argomenti, deve eseguire i valori di tali argomenti in parallelo, nel caso uno diverge mentre l'altro produce il valore *true*.

Uno può sfruttare l'indefinibilità della funzione "por" in P.C.F. per creare un paio di equivalenze contestuali di termini chiusi P.C.F. con denotazioni diverse. Questo lo vedremo con le prossime proposizioni.

7.1.3 Indefinibilità dell' OR-Parallelo

Proposizione 21:

Non esistono termini P.C.F. chiusi $P : bool \rightarrow (bool \rightarrow bool)$ che soddisfano $\llbracket P \rrbracket = por$.

7.1.4 Fallimento della proprietà di astrazione completa

Proposizione 22:

Per $i=1,2$ definiamo:

$$\begin{aligned} T_i &\stackrel{def}{=} \lambda f : bool \rightarrow (bool \rightarrow bool) \\ &\quad \text{if } (f \text{ true } \Omega) \text{ then} \\ &\quad \text{if } (f \ \Omega \text{ true}) \text{ then} \\ &\quad \quad \text{if } (f \text{ false false}) \text{ then } \Omega \text{ else } B_i \\ &\quad \text{else } \Omega \\ &\quad \text{else } \Omega \end{aligned}$$

where $B_1 \stackrel{def}{=} true$, $B_2 \stackrel{def}{=} false$, and $\Omega \stackrel{def}{=} fix(\lambda x : bool.x)$. Then

$$\begin{aligned} T_1 &\cong_{ctx} T_2 : (bool \rightarrow (bool \rightarrow bool)) \rightarrow bool \\ \llbracket T_1 \rrbracket &\neq \llbracket T_2 \rrbracket \in (B_\perp \rightarrow (B_\perp \rightarrow B_\perp)) \rightarrow B_\perp \end{aligned}$$

Dimostrazione:

Per la definizione di "por" e per la definizione di $\llbracket - \rrbracket$, abbiamo che:

$$\llbracket T_i \rrbracket (por) = \begin{cases} true & \text{se } i = 1 \\ false & \text{se } i = 2 \end{cases}$$

Così $\llbracket T_1 \rrbracket (por) \neq \llbracket T_2 \rrbracket (por)$ e quindi $\llbracket T_1 \rrbracket \neq \llbracket T_2 \rrbracket$.

Per vedere che $T_1 \cong_{ctx} T_2 : (bool \rightarrow (bool \rightarrow bool)) \rightarrow bool$ utilizziamo le proprietà di estensionalità di \cong_{ctx} .

Dobbiamo far vedere che per ogni $M : bool \rightarrow (bool \rightarrow bool)$ e $V \in \{true, false\}$:

$$T_1 M \Downarrow_{bool} V \Leftrightarrow T_2 M \Downarrow_{bool} V \quad (33)$$

Ma per la definizione di T_i abbiamo che $T_i M \Downarrow_{bool} V$ è vero solo se:

$$M \text{ true } \Omega \Downarrow_{bool} \text{ true}, M \Omega \text{ true } \Downarrow_{bool} \text{ true}, M \text{ false false } \Downarrow_{bool} \text{ false}$$

Per la proprietà della correttezza abbiamo che:

$$\llbracket M \rrbracket(true)(\perp) = true, \llbracket M \rrbracket(\perp)(true) = true, \llbracket M \rrbracket(false)(false) = false$$

Si ricorda che $\llbracket \Omega \rrbracket = \perp$.

Ne consegue che la funzione continua $\llbracket M \rrbracket : (B_\perp \times B_\perp) \rightarrow B_\perp$ coincide con la funzione “por”.

Ma è impossibile per la proposizione sull’inddefinibilità dell’or-parallelo. Quindi la proprietà (33) è banalmente soddisfatta per ogni M , e così T_1 e T_2 sono contestualmente equivalenti. ■

Bibliografia

- Appunti del corso di Semantica dei Linguaggi di Programmazione tenuto dalla Prof. Anna Labella per il Corso di “Semantica dei Linguaggi di Programmazione” per il Corso di Laurea in Informatica nell’anno 2004-2005.
- Dispense del corso “Matematiche Complementari” tenuto dalla Prof. Anna Labella per il Corso di Laurea in Matematica nell’anno 1999-2000.
- Note su “Semantiche Denotazionali” del Prof. Glynn Winskel dell’Università di Cambridge dell’anno 2003.

Sommario

0	Introduzione.....	2
1	I Numeri Naturali.....	4
1.1	Il Principio di Induzione.....	4
2	Semantica dei Linguaggi di Programmazione.....	7
2.1	Semantica Statica.....	7
2.2	Semantica Dinamica.....	7
2.2.1	Semantica Operazionale.....	7
2.2.2	Semantica Denotazionale.....	8
2.2.3	Semantica Assiomatica.....	8
2.3	Caratteristiche della Semantica Denotazionale.....	9
3	Strutture Algebriche e d'Ordine.....	10
3.1	Monoidi.....	10
3.2	Insiemi Ordinati.....	13
3.3	Insiemi Parzialmente Ordinati.....	16
3.4	Algebra dei Linguaggi.....	20
3.5	Domini.....	30
3.5.1	C.P.O. Discreto e Dominio Piatto.....	30
3.5.2	Definizione di Catena.....	31
3.6	Domini di Funzioni Parziali.....	32
3.7	Monotonicità, Continuità ed Esattezza.....	32
3.8	Domini Piatti.....	33
3.9	Prodotto tra Domini.....	34
3.10	Domini di Funzioni.....	37
3.11	La funzione Evaluation e la funzione Currying.....	38
3.12	Punto Fisso e Minimo Punto Fisso.....	40
3.13	Teorema del Punto Fisso di Tarski.....	40
3.14	Continuità dell'Operatore di Punto Fisso.....	42
3.15	Proprietà del Punto Fisso di $\llbracket \text{while } B \text{ do } C \rrbracket$	43
3.16	Catene chiuse e insiemi ammissibili.....	46
3.17	Principio di Induzione di Scott sui Punti Fissi.....	47
4	P.C.F. – Programming Computable Function.....	53
4.1	Termini e Tipi.....	53
4.2	Variabili libere, variabili vincolate e sostituzioni.....	54
4.3	Tipi.....	54
4.4	Valutazioni.....	56
4.4.1	Relazione di Valutazione P.C.F.....	56
4.5	Equivalenza contestuale contro l'uguaglianza nella denotazione.....	58
4.5.1	Semantica Denotazionale del linguaggio P.C.F.....	59
4.5.2	Equivalenza Contestuale.....	60
4.5.3	Equivalenza Contestuale dei termini di tipo P.C.F.....	60
5	Semantiche Denotazionali P.C.F.....	62
5.1	Tipi di Denotazioni.....	62
5.2	Denotazioni di termini.....	62
5.2.1	Semantiche Denotazionali di termini P.C.F.....	62
5.2.2	La Composizionalità preserva la Continuità.....	63
5.3	Composizionalità.....	65
5.3.1	Proprietà della sostituzione di $\llbracket - \rrbracket$	65
5.4	Correttezza.....	66

6	Relazioni Denotazionali e Semantiche Operazionali.....	69
6.1	Approssimazione formale di Relazioni.....	69
6.1.1	Proprietà fondamentali delle relazioni \triangleleft_r	70
6.2	Dimostrazione della proprietà fondamentale di \triangleleft	70
6.3	Estensionalità.....	73
6.3.1	Preordine Contestuale tra termini P.C.F.....	74
6.3.2	Preordine Contestuale da approssimazioni formali.....	74
6.3.3	Proprietà di estensionalità di \leq_{ctx}	75
7	Astrazione Completa.....	76
7.1	Fallimento dell'astrazione completa.....	76
7.1.1	Funzione di OR-Parallelo.....	76
7.1.2	Funzione di OR-Sequenziale.....	76
7.1.3	Indefinibilità dell'OR-Parallelo.....	77
7.1.4	Fallimento della proprietà di astrazione completa.....	77
	Bibliografia.....	79