UNIVERSITÀ DEGLI STUDI DI PADOVA | DIPARTIMENTO MATEMATICA

Università degli Studi di Padova
Dipartimento di Matematica "Tullio Levi-Civita"
Corso di Laurea Magistrale in Informatica

Esame di Teoria dei Tipi

Teoria dei Tipi
Elaborato scritto - Settembre 2020
Eleonora Signor, 1237581

Indice

Capitolo 1

Introduzione

1.1 La triplice faccia della teoria dei tipi

La teoria dei tipi offre una base teorica a fondamento dello sviluppo di:

- **Matematica**: nella teoria degli insiemi;
- **Logica**: come fondamento dei connettivi logici e dei quantificatori, con trattazione mediate tecniche di *proof-teory* per dimostrarne la non falsità o non contradditorietà;
- **Informatica**: per la correttezza dei programmi, da una semantica operazionale a un certo tipo di operazioni.
  Con riferimento alla teoria degli insiemi, visto come linguaggio di programmazione funzionale, è possibile specificare con formule l'obiettivo di un programma e dimostrarne la correttezza attraverso la specifica.

La teoria dei tipi nasce per garantire la *Certified Proof Correctness*. Ovvero la correttezza dei programmi, volta a costruire gli assistenti automatici per le formalizzazioni.

1.2 Come nasce la teoria dei tipi

Gli errori di programmazione sono stati preponderanti alla nascita di metodi automatici, che assicurassero la correttezza del *software*. Alcuni di questi, degni di nota, sono stati:

- Incidente nel lancio dell'Apollo 11;
- Tragedie sanitarie: incidenti avvenuti tra il 1985-1987, in cui dei pazienti ricevettero una massiccia *overdose* di radiazioni e per la quali alcuni morirono;
- Errori di vita civile: riserva di solo due cifre per il campo età all'interno dei *database*. Ecco che una signora danese ricevette per il suo 107-esimo compleanno, una mail dalle autorità della scuola locale per iscriversi alla prima elementare.

Per la matematica la correttezza delle dimostrazioni è irrilevante solo quando la soluzione è certa (come accade con il cubo di *Rubik*, dove so che la soluzione è corretta quando ognuno dei lati è uniformemente colorato); e in generale questo è difficile che accada.
Un'esempio di problema, dove la soluzione non è certa, è il Teorema dei Quattro Colori, risolto da un *computer* e la cui prova di correttezza della dimostrazione fu data dal *proof assistant* Coq. Quest'ultimo basato sulla teoria dei tipi e intellegibile dall'essere umano.

Una citazione importante va al matematico Russo V.V. *Voevodsky*, vincitore della medaglia *Fields*. Esso si battè per la creazione di un *proof assistant*, per rendere le dimostrazioni da informali, per problemi complessi, a completamente formalizzate, con l'impiego della teoria dei tipi. I suoi studi trovano principale applicazione in campo algebrico e geometrico; ma i concetti emersi assunsero delle 1connotazioni più ampie. *Voevodsky*, difatti, si rese conto che formalizzare equivale a programmare. Ciò significa che la teoria dei tipi permette di vedere una dimostrazione come un programma.

Esiste la certezza assoluta per una certa teoria, esclusivamente, quando ha un numero di assiomi, accettati per fede, molto limitato. In quanto assiomalizzabile da un calcolatore.

In coclusione formalizzare in una teoria dei tipi (come quella degli insiemi) equivale a programmare un programma.

1.3 Il Paradosso di Russell

La base della teoria dei tipi, compresa quella di *Martin-Löf*, si deve a B. *Russell*. Siamo nel 1907 quando nasce la teoria dei tipi, sviluppata nei *Principia Mathematica* da B. *Russel* assieme ad A.N. *Whitehead*. Tale teoria, intesa come logica e non informatica, nasce come soluzione alternativa alla teoria degli insiemi, di allora, con lo scopo di fondare la matematica su un sistema formale accettabile e non contradditorio.
Di seguito espongo un sistema contradditorio della teoria degli insiemi.

Linguaggio *L* di una teoria degli insiemi *F*

- L linguaggio del primo ordine (=, &, $\rightarrow$, $\vee$, $\forall x$, $\exists x$), con l'aggiunta del predicato $\in$ "appartiene"
- variabili VAR $\ni$ {x, y, z, w,. . . }

dove x, y, z sono da intendersi come insiemi e x ϵ y = "x appartiene a y".

All'interno di L c'è una teoria degli insiemi. Tra cui prende posto l'**assioma di comprensione di *Frege***, definito nel modo seguente:
Per ogni formula ϕ(x) vale che $\exists$z $\forall$y (y$\in$z $\Leftrightarrow$ ϕ(y)) [$\equiv$ $\exists$z z = {x $\mid$ ϕ(x)}]

Teorema (o Paradosso) di *Russell*: la teoria F è contradditoria.

Dimostrazione:
ϕ(x) = x$\notin$x ($\equiv \neg$ (x $\in$ x))
Per l'assioma di comprensione $\exists$z z = {x | x$\notin$x} ($\exists$z $\forall$y (y$\in$z $\Leftrightarrow$ y$\notin$y))
Ponendo y=z ottengo che z$\in$z $\Leftrightarrow$ z$\notin$z, risulta una **contraddizione**.

L'assioma di comprensione è contradditorio perchè permette di formare insiemi che non appartengono a se stessi.

Come correggere la contraddizione?
La soluzione accettabile è porre agli insiemi una **gerarchia di tipi**. In questo modo l'assioma di comprensione diventa:
$\exists$z $\forall$y (y$\in$a $\rightarrow$ (y$\in$z $\Leftrightarrow$ ϕ(y)) $\equiv$ z = {x$\in$a $\mid$ ϕ(x)}

In questo modo non posso più creare il Paradosso di *Russell*.

Al momento questa teoria dei tipi non è utilizzata. Una sua evoluzione diretta è la teoria dei tipi di *Martin-Löf*.

L'idea di *Russell* fu dunque quella di costruire insiemi partendo da una gerarchia.

1.4 Idee principali nelle teorie di tipo moderne

Le teorie di tipo moderne (chiamate λ-calcolo tipato) nascono, nel corso degli anni '30, dalla combinazione della teoria di tipo di *Russell* con il λ-calcolo di *Church*.

1.4.1 Richiamo della teoria del λ-calcolo di *Church*

Ha origine dalla logica, è un linguaggio in grado di trattare le funzioni e rivolto alla loro formalizzazione. Consiste in un linguaggio formale, le cui componenti principali sono programmi chiamati termini (pensati come funzioni).
La grammatica è la seguente:

$$\text{t} := \text{x} \mid \text{b}_1(\text{b}_2) \mid \lambda\text{x.t}$$

Esempio di applicazione: tg(x) $\equiv \lambda$x.tg(x)

Regole di computazione di base

$$(\lambda x.t)(b) \rightarrow t[\frac{x}{b}] \qquad \frac{b_1 \rightarrow b_2 \qquad a_1 \rightarrow a_2}{b_1(a_1) \rightarrow b_2(a_2)} \qquad \frac{b_1 \rightarrow b_2}{\lambda x.b_1 \rightarrow \lambda x.b_2}$$

Si dice che un programma si riduca a un altro, cioè converge, solo se c'è una sequenza di riduzioni (applicazione di regole e/o assiomi), che connettono il primo programma con l'ultimo. Si parla, in questo modo, di **chiusura transitiva e simmetrica**, che si conclude quando il programma non è più riducibile. Quanto appena descritto può venire espresso nel seguente modo:

$t \rightarrow t'$ sse esiste un numero finito di passi per cui t si riduce a t', ovvero esiste $b_1 \dots b_m$ t.c. $t \rightarrow b_1 \rightarrow b_2 \cdots \rightarrow b_m \rightarrow t'$.

Il λ calcolo permette di codificare qualsiasi programma scritto in qualunque linguaggio (imperativo, dichiarativo, Java, C++, BASIC, ...). Tuttavia tale linguaggio non codifica solo programmi che terminano, ma anche programmi che non lo fanno. Un esempio di applicazione, per quest'ultima categoria, è un programma con computazione infinita: $\lambda x.x(x)$
$\lambda x.x(x)$ lo applichiamo a se stesso. Perció diventa $\Lambda \equiv (\lambda x.x(x))(\lambda x.x(x))$ che seguendo la computazione si riduce a:

$$x(x)[\frac{x}{\lambda x.x(x)}] \equiv (\lambda x.x(x))(\lambda x.x(x))$$

Dunque esiste una catena di $(t_i)_{i\in\mathbb{N}}$ di termini $t_i \to t_{i+1}$. Ciò significa che Λ non termina in qualunque linguaggio sia interpretato.

Λ risulta un buon metodo per rappresentare le funzioni, ma non è completo, rispetto all'intuizione matematica di funzione. È necessario, per questo, tipare le variabili; ovvero λx.x$\in$A$\to$B(x$\in$A).

Il λ-cacolo tipato, nato dal λ-calcolo "puro", è anch'esso un linguaggio di programmazione. Essendo tipato può essere trattato come una teoria degli insiemi.

1.5 Che cosa è un tipo?

Per rispondere a questa domanda è necessario fornire la semantica intuitiva di tipo. Per farlo è utile pensare alla teoria dei tipi come paradigma di fondazione sia logico che matematico che informatico.

Sintassi in teoria dei tipi moderna	**Sintassi in teoria degli insiemi**	**Sintassi in un linguaggio logico/per una logica (anche predicativo)**	**Sintassi in un linguaggio di programmazione**
A type	A set	A prop	A data type
a$\in$A	a$\in$A set	a$\in$A	a$\in$A

Tabella 1.1: Sintassi per i diversi paradigmi funzionali.

Per la sintassi:

- nella **teoria dei tipi moderna** a rappresenta un termine e A un tipo;
- nella sintassi in una **teoria degli insiemi** a è un elemento e A un insieme. Coincidendo con la corrispondenza originale in mente da *Russell*.
- nella sitassi in un **linguaggio logico** a rappresenta una proposizione di A, inteso come insieme di tipo delle sue dimostrazioni. Perciò un *proof-term* affermante come la proposizione di A sia vera.
- nella teoria in una sintassi di un **linguaggio di programmazione** a rappresenta un programma e A una specifica.

Dunque quando parliamo di tipo ci riferiamo a un insieme, una proposizione o *data type*, a seconda dell'applicazione di tipo che si ha in mente.

Dal punto di vista logico non si hanno solo proposizioni, ma anche predicati. Parlare solo di tipo non risulta quindi sufficiente. Per questo se si vuole rappresentare non una proposizione, ma un predicato A(x) si usa la seguente sitassi: **A(x) prop[x∈D]**.
Dalla logica si sa che i predicati ϕ(x) hanno x senza un dominio specifico, perchè la sintasssi non determina che cosa è in x. Al seguito di tutto questo i predicati hanno una variabile che deve essere tipata come **ϕ(x) prop[x∈D]**.
Dunque (definizione di predicato)

$$\exists\ \mathbf{z} \qquad \mathbf{z}=\{\mathbf{x}\in\ \mathbf{a}|\ \phi(\mathbf{x})\} \qquad \equiv \qquad \phi(\mathbf{x})\mathbf{prop}[\mathbf{x}\in\ \mathbf{a}]$$

Quanto appena definito da origine al concetto di **tipo dipendente**, nel quale vengono tipate tutte le variabili che appartengono ad una **famiglia di tipo**.

Le famiglie di tipo sono indispensabili per rappresentare il concetto di predicato. Di seguito ho riassunto in forma tabellare le diverse famiglie.

di tipo	negli insiemi	in logica	dati dipendenti
A(x) prop[x∈D]	A(x) set[x∈D]	A(x) prop[x∈D]	A(x) datatype[x∈D]

Tabella 1.2: Famiglia di tipi.

Il concetto di tipo dipendente è stato introdotto per la prima volta da *Martin-Löf*. *Russell* si era limitato a definire esclusivamente il concetto di funzione proposizionale dipendente da un tipo.

1.6 Esempi di tipi

A type	A set	A prop	A data type
N_1 singoletto	l'insieme singoletto	*tt* costante vero	tipo Unit
N_0 vuoto	l'insieme vuoto	$\perp$ costante falso	vuoto come data-type
BxC (tipo prodotto)	l'insieme prodotto cartesiano dell'insieme B con l'insieme C	B&C congiunzione della proposizione B e della proposizione C	tipo prodotto cartesiamo (come in *set theory*)
B+C (tipo somma binaria)	l'insieme unione disgiunta dell'insieme B con l'insieme C	B$\vee$C disgiunta della proposizione B e della proposizione C	tipo unione disgiunta con codifica
B$\rightarrow$C	l'insieme delle funzioni dall'insieme B verso l'insieme C: A$\rightarrow$B $\equiv$ {f | f: B$\rightarrow$C}	B$\rightarrow$C, implicazione della proposizone B e della proposizione C	insieme delle funzioni dal *datatype* B al *datatype* C

Tabella 1.3: Famiglia di tipi.

1.6.1 I tipi dipendenti

A(x)type[x$\in$B]
tipo indiciato $\prod_{x \epsilon B} C(x)$
tipo somma disgiunta indiciata $\sum_{x \epsilon B} C(x)$

A(x)set[x$\in$B]	A(x)prop[x$\in$B]	A(x)datatype[x$\in$B]
$\{f : B \rightarrow \coprod_{x \epsilon B} C(x)\}$ $\coprod_{x \epsilon B} C(x) = \{b, c \mid b \epsilon B \quad c \epsilon C(b)\}$	$\forall x \epsilon B \quad C(x)$	tipo prodotto indiciato come in *set theory* (non esiste un *data-type* specifico)
$\dot{\bigcup}_{x \epsilon B} C(x)$ $\coprod_{x \epsilon B} C(x) = \{b, c \mid b \epsilon B \quad c \epsilon C(b)\}$	$\exists x \epsilon B \quad C(x)$	non è primitivo deriva sempre dalla *set theory*

Tabella 1.4: Tipi dipendenti.

Lo slogan principale della teoria dei tipi è quello di tipare le variabili in un linguaggio formale set teorico/computazionale.

Esite anche il **tipo uguaglianza**:

- intensionale: Id(B,c,d);
- estensionale: Eq(B,c,d).

Introdotte da *Martin-Löf*.
E i costrutti degli **universi**, in cui U è universo di proposizioni e di insiemi.

1.7 Regole paradigmatiche per caratterizzare la teoria dei tipi

La teoria dei tipi è stata formalizzata usando la nozione di **giudizio**, dove si asserisce qualcosa come vero.
Ci sono quattro forme di giudizio (nelle quali Γ identifica il contesto):

- **A type[Γ]**: A è un tipo, possibilmente indicato da variabili nel contesto Γ, dipendente da Γ stesso. Rappresenta il giudizio di tipo.
- **A = B type[Γ]**: il tipo A dipendente da Γ è uguale al tipo B dipendente da Γ. Rappresenta il giudizio di uguaglianza di tipo.
- **a $\in$ A [Γ]**: a è un elemento del tipo A, possibilmente indiciato, ovvero dipendente da Γ e dalle sue variabili di contesto. Un esempio di tipo dipendente è l'array, che ha termini di funzioni che dipendono da Γ. Invece il termine non è dipendente quando si parla di funzione costante senza variabili.
- **a = b $\in$ A [Γ]**: a come elemento del tipo A dipende da Γ ed è uguale in modo definizionale/computazionale al termine b. Quest'ultimo, difatti, è elemento del tipo A dipendente da Γ.

All'interno di ogni singolo giudizio si lavora con la teoria dei tipi.
I giudizi solo esclusivamente asserzioni, dicono solo qualcosa quando è vero (non si usano i quantificatori). Essi limitano le frasi che si possono fare per codificare la logica intuizionistica.

1.7.1 Simbolo $\in$

Il significato di a$\in$A in teoria dei tipi è differente da quello insiemistico. Espongo il concetto con un esempio trattato a lezione:

$$1 \in Nat \tag{1.1}$$

- In ***set theory*** usuale $\in$ è tra insiemi. Nell'equazione 1.1, 1 rappresenta lui stesso un'insieme e Nat l'insieme dei numeri Naturali. Risulta vero che $1 \equiv \{\varnothing\}$, poichè $0 \equiv \varnothing$.
- Invece in **teoria dei tipi** (di *Martin-Löf* come di *Russell*) 1 rappresenta un elemento ma non un tipo e Nat il tipo dei Naturali. Vi è dunque la distinzione tra elemento e tipo (come esiste quella tra programmi e tipi).

1.7.2 Uguaglianza definizionale vs uguaglianza proposizionale

Specifico **a = b∈ A[Γ]** come l'uguaglianza computazionale/definizionale, che viene data come primitiva e non va confusa con l'uguaglianza proposizionale/estensionale tra a e b.
L'uguaglianza proposizionale a $=_A$ b è rappresentata non da un giudizio, che asserisce solo ciò che è vero, ma bensì da un tipo Eq(A,a,b) che può anche essere senza termini e/o essere falso, dal punto di vista logico.

Visti come programmi, a e b rappresentano lo stesso programma. In λ-calcolo a$\rightarrow$b oppure b$\rightarrow$a (si riducono). Inoltre a e b possono essere sia termini finali che trovarsi in mezzo alla computazione.

1.7.3 Generazione di contesti

Esiste anche un quinto giudizio ausiliario (§ 2.1.1) *F-C*, che permette di generare i contesti. Tale giudizio, a differenza dei primi quattro, rimane immutato in ogni teoria dei tipi.

1.8 Esercizi

1.8.1 λ-calcolo puro

1.8.2 subsec:λ-calcolo-puro

1 Dato il seguente termine, elencare quali sono le sue variabili libere e le sue variabili legate con i lambda relativi.

$$\lambda z.(((\lambda x.\lambda x.yx)x)(v\lambda z.\lambda w.v))$$

Soluzione

$\lambda z.(((\lambda x.\lambda \mathbf{x}.y\mathbf{x})\mathbf{x})(\mathbf{v}\lambda z.\lambda w.\mathbf{v}))$

- le variabili libere (colorate in verde) sono *y, x* e *v*
- le variabili legate con i relativi lambda termini (colorate in rosso) sono la *x*

2 Rinominare le variabili legate nel seguente termine in modo che non ci siano due variabili legate con lo stesso nome.

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

Soluzione

$x(\lambda \mathbf{x}.((\lambda \mathbf{x}.x)x))$

Le variabili legate sono le x all'interno delle parentesi tonde. Una possibile rinomina, per evitare che queste variabili legate abbiano lo stesso nome, è $x(\lambda x.((\lambda y.y)x))$, dove la x della parentesi più interna è stata sostituita con la y.

3 Evidenziare di due colori diversi quali sono le variabili libere e quali quelle legate.

$$(\lambda x.(z(\lambda z.((xyz)x))zx))x(\lambda x.((\lambda y.yy)(\lambda z.zz)))$$

Soluzione

$(\lambda x.(\mathbf{z}(\lambda z.((\mathbf{x}\mathbf{y}\mathbf{z})\mathbf{x}))\mathbf{z}\mathbf{x}))\mathbf{x}(\lambda x.((\lambda y.\mathbf{y}\mathbf{y})(\lambda z.\mathbf{z}\mathbf{z})))$

- le variabili libere sono le *z, y* e *x* colorate in verde
- le variabili legate sono le *x, y* e *z* colorate in rosso

4 Descrivere un termine del λ-calcolo, descritto in §1.4.1, che è convergente con almeno un passo di riduzione rispetto ad una specifica strategia di riduzione deterministica.

Soluzione

Per definizione un termine t è convergente, rispetto a una strategia di riduzione deterministica, se esiste un numero finito n >= 1 di termini s_1,..., s_n tale che $s_1 \equiv$ t e $s_i \rightarrow_1 s_{i+1}$ per i = 1,..., n-1 e s_n non è riducibile ad alcun termine. Prendendo in considerazione una strategia di riduzione deterministica **call-by value**, usata per la semantica nei linguaggi di programmazione, allora un esempio di termine t del λ-calcolo convergente in almeno un passo di riduzione è:

$$t \equiv ((\lambda x.x)z)((\lambda y.y)w) \rightarrow_1 z((\lambda y.y)w) \rightarrow_1 z(w) \equiv s_n \equiv s$$

5 Descrivere due termini diversi del λ-calcolo, descritto in §1.4.1, che non sono convergenti sempre rispetto a una strategia di riduzione deterministica.

Soluzione

Per definizione un termine t diverge (è non convergente), rispetto a una strategia di riduzione deterministica, se esiste una quantità numerabile di termini s_i al variare di i $\in$ Nat tale che $s_1 \equiv$ t e $s_i \rightarrow_1 s_{i+1}$. Per ogni i $\in$ Nat (ossia esiste una lista infinita di passi computazioni a partire da t). Prendendo in considerazione una strategia di riduzione deterministica *call-by value*, usata per la semantica nei linguaggi di programmazione, allora un esempio di due termine diversi del λ-calcolo che sono convergenti è:

$(\lambda x\ x(x))(\lambda y\ y(y)) \rightarrow_1 (\lambda y\ y(y))(\lambda y\ y(y)) \rightarrow_1 \ldots \rightarrow_1 (\lambda y\ y(y))(\lambda y\ y(y)) \rightarrow_1 \ldots$

Si riduce sempre a se stessa, a qualunque passo di computazione. Perciò ammette computazione infinita (diverge) non raggiungendo mai un valore finale.

6 Che relazione c'è tra il lambda-calcolo puro con le regole di riduzione date in §1.4.1, rispetto a quello in cui, adottando la stessa sintassi di termini, imponiamo la seguente definizione di riduzione $\rightarrow_1^*$.

- per ogni termine t e b $(\lambda x.t)(b) \rightarrow_1^* t[\frac{x}{b}]$

- per ogni termine b, b_1, b_2 e a, a_1, a_2

$$R_I \frac{a1 \rightarrow_1^* a2 \qquad b1 \rightarrow_1^* b2}{a1(b1) \rightarrow_1^* a2(b2)} \qquad R_{II} \frac{t1 \rightarrow_1^* t2}{\lambda\, x.t1 \rightarrow_1^* \lambda x.t2}$$

Soluzione

Idea: devo provare che relazione esiste tra $\rightarrow_1^*$ e $\rightarrow$. Per cui verifico cosa accade per ($\rightarrow_1^* \subseteq \rightarrow_1$) e ($\rightarrow_1 \subseteq \rightarrow_1^*$)

Sia L(T) l'insieme dei λ termini che è possibile ridurre in forma normale, con la strategia di riduzione T. Dimostro che valgono le seguenti relazioni tra $\rightarrow_1^*$ e $\rightarrow$:

1. $L(\rightarrow_1^*) \nsubseteq L(\rightarrow_1)$
2. $L(\rightarrow_1^*) \subset L(\rightarrow_1)$

1. Si ha il λ termine $t \equiv ((\lambda x.x)z)((\lambda y.y)w)$
   Allora applicando la strategia di riduzione $\rightarrow$, ottengo

$$\frac{\lambda x.x \rightarrow z}{((\lambda x.x)z)((\lambda y.y)w) \rightarrow z((\lambda y.y)w)}$$

$$\frac{(\lambda y.y)w \rightarrow w}{z((\lambda y.y)w) \rightarrow z(w)}$$

   Dunque riesco a giungere a una forma normale.
   Cosa che non è possibile con la strategia $\rightarrow_1^*$. In quanto $((\lambda x.x)z)((\lambda y.y)w) \rightarrow_1^* ((\lambda x.x)z)((\lambda y.y)w)$ che non è in forma normale $(((\lambda x.x)z)((\lambda y.y)w) \nrightarrow_1^*)$.
   In conclusione risulta vero che $L(\rightarrow_1^*) \nsubseteq L(\rightarrow_1)$.

2. Per provare l'inclusione di $L(\rightarrow_1^*)$ in $L(\rightarrow_1)$ basta che dimostro, per una valutazione che usa entrambe le strategie, il sempre possibile rimpiazzo della regola R_I con la sua regola corrispondente in $\nrightarrow_1$ (A_I + A_{II}) . Più formalmente significa provare che per ogni valutazione di un termine M, che usa la regola R_I, si può sempre ottenere una valutazione che usa solo regole della strategia $\nrightarrow_1$.
   Per farlo procedo per induzione sul numero di volte n che la regola R_I viene utilizzata durante la valutazione di M.

   - (n=0): caso base. La regola R_I non viene mai utilizzata nella valutazione di M, e dunque M è già impicitamente dimostrato con la strategia $\nrightarrow_1$
   - (n $\rightarrow$ n+1): caso induttivo. Per n risulta vero che $L(\rightarrow_1^*) \subset L(\rightarrow)$, devo provare che vale anche per n+1. Inoltre la valutazione di M utilizza almeno una volta la regola R_I.
     Dunque ho $M \rightarrow ... \xrightarrow{R_I}{}^*_1 M^I$ e voglio costruire una derivazione $M \rightarrow ... \xrightarrow{A_I}_1$

$M_1 \xrightarrow{A_{II}}_1 M^I$. Le strategie di valutazione sono deterministiche, portano pertanto allo stesso risultato della sequenza di derivazione, percui $\mathrm{R}_I = \mathrm{A}_I + \mathrm{A}_{II}$ risulta vero. Inoltre per ipotesi induttiva è sempre possibile avere una valutazione con l'utilizzo di solo regole $\rightarrow_1$ (A_I + A_{II} esistono). Perciò risulta corretto rimpiazzare $\rightarrow_1^*$ con $\rightarrow_1$.
In conclusione risulta vero che $\mathrm{L}(\rightarrow_1^*) \subset \mathrm{L}(\rightarrow_1)$.

Capitolo 2

Regole della teoria dei tipi

Lo scopo della teoria dei tipi è offrire un sistema formale in cui derivare, tramite regole e assiomi, giudizi nella forma:

$$A\ type[\Gamma] \qquad A = B\ type[\Gamma] \qquad a \in A\ [\Gamma] \qquad a = b \in A\ [\Gamma]$$

$$+\ ausiliaria \quad \Gamma\ cont$$

L'ultimo giudizio non è necessario, serve esclusivamente per imparare.
Quando si formula una nuova teoria dei tipi è bene impiegare il minor numero possibile di regole strutturali e di formazione di tipi e termini. Tali regole devono essere rivolte all'ottimizzazione e correttezza della teoria. Alcune di queste, come quelle di indebolimento e sostituzione in §2.1.5, sono irrinunciabili, la cui validità è sempre garantita e utilizzate nella derivazione di ogni teoria.

Se la teoria dei tipi è dipendente si ha bisogno di tutti i giudizi. Invece in una teoria dei tipi non dipendente, come quella dei tipi semplici, il giudizio $A = B\ type[\Gamma]$ può venire omesso.

2.1 Regole strutturali

Assioma unico: [ ] cont

Nel calcolo dei sequenti, in logica classica, le derivazioni di giudizio, valide in una teoria dei tipi con solo le regole singoletto, diventano derivazioni di sequenti nella forma $\Gamma \dashv A$ e unico assioma $\varphi \dashv \varphi$.

Di seguito illustro le principali regole di contesto comuni a tutte le teorie dei tipi.

2.1.1 Regole di formazione dei contesti

$$[\,]\ \text{cont} \qquad \text{dove } [\,] = \varnothing$$

$$\text{F-C)}\ \frac{\text{A type}[\Gamma]}{\Gamma,\ \text{x} \in \text{A}}\ \text{x} \in \text{A} \notin \Gamma$$

2.1.2 Regole di assunzione delle variabili

$$\text{var-ass)}\ \frac{\Gamma,\ \mathrm{x} \in \mathrm{A},\ \Delta \qquad \text{cont}}{\mathrm{x} \in [\Gamma, \mathrm{x} \in \mathrm{A},\ \Delta]}$$

2.1.3 Regole strutturali addizionali sull'uguaglianza

L'uguaglianza, in una teoria dei tipi, consiste in una relazione di equivalenza sia fra tipi che fra termini. Sono perciò valide le seguenti regole di uguaglianza tra tipi:

$$ref)\ \frac{A\ type[\Gamma]}{A = A\ type[\Gamma]} \qquad sym)\ \frac{A = B\ type[\Gamma]}{B = A\ type[\Gamma]}$$

$$tra)\ \frac{A = B\ type[\Gamma] \qquad B = C\ type[\Gamma]}{A = C\ type[\Gamma]}$$

E allo stesso modo anche le regole di uguaglianza definizonale/computazionale tra termini:

$$ref)\ \frac{a \in A\ [\Gamma]}{a = a \in A\ [\Gamma]} \qquad sym)\ \frac{a = b \in A\ [\Gamma]}{b = a \in A\ [\Gamma]}$$

$$tra)\ \frac{a = b \in A\ [\Gamma] \qquad b = c \in A\ [\Gamma]}{a = c \in A\ [\Gamma]}$$

2.1.4 Regole di conversione dell'uguaglianza per tipi uguali

L'appartenenza si conserva con l'uguaglianza di termini e tipi. Le regole da aggiungere, in una teoria dei tipi, per garantirlo sono:

$$conv)\ \frac{a \in A\ [\Gamma] \quad A = B\ type[\Gamma]}{a \in B\ [\Gamma]}$$

$$conv - eq)\ \frac{a = b \in A\ [\Gamma] \qquad A = B\ type[\Gamma]}{a = b \in B\ [\Gamma]}$$

2.1.5 Regole di indebolimento e di sostituzione

Indebolimento

$$ind - ty)\ \frac{A\ type[\Gamma] \quad \Gamma, \Delta\ cont}{A\ type[\Gamma, \Delta]} \qquad ind - ty - eq)\ \frac{A = B\ type[\Gamma] \quad \Gamma, \Delta\ cont}{A = B\ type[\Gamma, \Delta]}$$

$$ind - te)\ \frac{a \in A\ [\Gamma] \quad \Gamma, \Delta\ cont}{a \in A\ [\Gamma, \Delta]} \qquad ind - te)\ \frac{a = b \in A\ [\Gamma] \quad \Gamma, \Delta\ cont}{a = b \in A\ [\Gamma, \Delta]}$$

Sostituzione

$$\frac{C(x_1, ..., x_n)\ type[\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1, ..., x_{n-1})]}{sub - typ)\quad \frac{a_1 \in A_1\ [\Gamma]\ ...a_n \in A_n(a_1, ..., a_{n-1})\ [\Gamma]}{C(a_1, ..., a_n)\ type[\Gamma]}} \tag{2.1}$$

$$\frac{C(x_1, ..., x_n)\ type[\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1, ..., x_{n-1})]}{sub - eq - typ)\quad \frac{a_1 = b_1 \in A_1\ [\Gamma]\ ...a_n = b_n \in A_n(a_1, ..., a_{n-1})\ [\Gamma]}{C(a_1, ..., a_n) = C(b_1, ..., b_n)\ type[\Gamma]}} \tag{2.2}$$

$$C(x_1,...,x_n) = D(x_1,...,x_n)\ type[\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1,...,x_{n-1})]$$
$$sub-Eqtyp) \quad \frac{a_1 \in A_1\ [\Gamma]\ ...a_n \in A_n(a_1,...,a_{n-1})\ [\Gamma]}{C(a_1,...,a_n) = D(a_1,...,a_n)\ type[\Gamma]} \tag{2.3}$$

$$C(x_1,...,x_n) = D(x_1,...,x_n)\ type[\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1,...,x_{n-1})]$$
$$sub-eq-Eqtyp) \quad \frac{a_1 = b_1 \in A_1\ [\Gamma]\ ...a_n = b_n \in A_n(a_1,...,a_{n-1})\ [\Gamma]}{C(a_1,...,a_n) = D(a_1,...,a_n)\ type[\Gamma]} \tag{2.4}$$

$$c(x_1,...,x_n) \in C(x_1,...,x_n)\ [\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1,...,x_{n-1})]$$
$$sub-ter) \quad \frac{a_1 in A_1\ type[\Gamma]\ ...a_n \in A_n(a_1,...,a_{n-1})\ [\Gamma]}{c(a_1,...,a_n) \in C(a_1,...,a_n)\ type[\Gamma]} \tag{2.5}$$

$$c(x_1,...,x_n) = d(x_1,...,x_n) \in C(x_1,...,x_n)\ [\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1,...,x_{n-1})]$$
$$sub-eqter) \quad \frac{a_1 \in A_1\ [\Gamma]\ ...a_n \in A_n(a_1,...,a_{n-1})\ [\Gamma]}{c(a_1,...,a_n) = d(a_1,...,a_n) \in C(a_1,...,a_n)\ [\Gamma]} \tag{2.6}$$

$$c(x_1,...,x_n) \in C(x_1,...,x_n)\ [\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1,...,x_{n-1})]$$
$$sub-eq-ter) \quad \frac{a_1 = b_1 \in A_1\ [\Gamma]\ ...a_n = b_n \in A_n(a_1,...,a_{n-1})\ [\Gamma]}{c(a_1,...,a_n) = c(b_1,...,b_n) \in C(a_1,...,a_n)\ [\Gamma]} \tag{2.7}$$

$$c(x_1,...,x_n) = d(x_1,...,x_n) \in C(x_1,...,x_n)\ [\Gamma, x_1 \in A_1, ..., x_n \in A_n(x_1,...,x_{n-1})]$$
$$sub-eq-eqter) \quad \frac{a_1 = b_1 \in A_1\ [\Gamma]\ ...a_n = b_n \in A_n(a_1,...,a_{n-1})\ type[\Gamma]}{c(a_1,...,a_n) = d(b_1,...,b_n) \in C(a_1,...,a_n)\ [\Gamma]} \tag{2.8}$$

2.1.6 Regole proprie e derivabili

In una teoria formale ci sono due tipi di regole:

- **regole proprie del calcolo**, come lo sono le regole strutturali e quelle del singoletto;
- **regole derivabili**, come le regole di sostituzione, utili per abbreviare le derivazioni.

Una regola r $\frac{J_1,...,J_n}{J}$ è ammissibile in un calcolo t sse i giudizi derivabili in $t+r$ sono gli stessi dei giudizi derivabili in t. Ciò comporta che l'aggiunta di una regola rt non cambia i giudizi che ne possono derivare.
Quando un assioma è ammissibili e derivabile questo coincide con un giudizio derivabile.

2.1.7 Nozione di contesto telescopico

Un giudizio, in teoria dei tipi dipendenti si esprime nella forma

$$A(x_1, ..., x_n)[x_1 \in B_1, ..., x_n \in B_n]$$

e prende il nome di **contesto telescopico**. Questi presenta una dipendenza continua, esemplificata nel seguente giudizio

$$A(x_1, x_2, x_3)\ type[x \in C_1, x_2 \in C(x_1), x_3 \in C(x_1 x_2)..)$$

Inoltre si parla di contesti rigidi, ovvero senza possibilità di scambio. Come appare dall'esempio sotto.
$[x \in Nat, y \in Nat, z \in Mat(x, y)]$ cont $\Rightarrow$ **è derivabile**
$[y \in Nat, x \in Nat, z \in Mat(x, y)]$ cont $\Rightarrow$ **è derivabile**
$[y \in Nat, z \in Mat(x, y), x \in Nat] \Rightarrow$ **non è un contesto**.

Percui non esiste lo scambio arbitrario, si deve porre attenzione alle dipendenza delle assunzioni, che provoca una sostituzione rigida.

2.1.8 Esempi di applicazione

Attenzione all'ordine di sostituzione si deve partire sempre da quello con meno dipendenze.

$$\frac{\text{c} \in [\text{C}, \Gamma] \qquad \text{b} \in [\text{B(c)}, \Gamma] \qquad \text{A(x,y) type}[\text{x} \in \text{C}, \text{y} \in \text{B(x)}]}{\text{A(c,b) type}[\Gamma]}$$

$$\frac{\text{c} \in [\text{C},\Gamma] \qquad \text{b} \in [\text{B(c)},\Gamma] \qquad \text{A(x,y) type}[\text{x} \in \text{C}, \text{y} \in \text{B(x)}]}{\text{a(c,b)} \in \text{A(c,b)}\ [\Gamma]}$$

Se si ha un tipo puo' venire usato il giudizio di uguaglianza tra termini e la sostituzione.

$$\frac{\text{A(x) type}[\Gamma, \text{x} \in \text{C}] \qquad \text{c} = \text{e} \in \text{C}[\Gamma]}{\text{A(c)} = \text{A(e) type}[\Gamma]}$$

È dunque fondamentale il concetto di uguaglianza fra tipi. Se considero che ci sia un elemento

$$\frac{\text{a(x)} \in \text{A(x)}\ [\Gamma, \text{x} \in \text{C}] \qquad \text{c} = \text{e} \in \text{C}[\Gamma]}{\text{a(c)} = \text{a(e)} \in \text{A(c)}\ [\Gamma]}$$

Per poter affermare che A(e) = A(c) devo poterlo dedurre. Per farlo mi sono indispensabili le **regole di conversione dell'uguaglianza del tipaggio (§2.1.9)**.

2.1.9 Regole di tipaggio

Regole di Conversione

$$\frac{c \in C[\Gamma] \qquad C = D\ type[\Gamma]}{C \in D\ [\Gamma]}$$

Se due tipi sono uguali allora hanno gli stessi termini: $C = D \Rightarrow (c \in C \Leftrightarrow c \in D)$. L'uguaglianza fra tipi è per questo simmetrica.
Tuttavia non sempre l'unicità del tipaggio di un termine (il $\Leftrightarrow$) è garantito per ogni teoria. Nei casi trattati dal corso sì, in quanto verrà inteso che $C = D\ type[\Gamma]$ sse due tipi hanno gli stessi elementi (come già accade in *set theory*), ma può non essere sempre vero.

Regole di Conversione dell'uguaglianza

$$\frac{c = d \in C[\Gamma] \qquad C = D\ type[\Gamma]}{c = d \in D\ [\Gamma]}$$

Questa regola permette di convertire le uguaglianze nel tipaggio di un termine.

2.2 Il tipo singoletto

Il tipo singoletto risulta essere paradigmatico per gli altri tipi. Per definirlo impiegherò i giudizi nella forma $A\ type[\Gamma]$, $a \in A\ [\Gamma]$ e $a = b \in A\ [\Gamma]$. L'uguaglianza, invece, non verrà coinvolta, in quanto non può essere impiegata per definire un nuovo tipo, è difatti usata solo nelle derivazioni.

Innanzitutto come già visto in §2.1.1 ogni derivazione parte sempre dal contesto vuoto ($\varnothing$).

$$\text{F-C)}\ \frac{\text{A type}[\Gamma]}{\Gamma, \text{x} \in \text{A}}\ \text{x} \in \text{A} \notin \Gamma$$

2.2.1 Regola di Formazione

$$\text{F-S)}\ \frac{[\Gamma]\ \text{cont}}{N_1\ \text{type}[\Gamma]}$$

La regola F-S Permette di derivare vari giudizi e di dire che cosa è un tipo.

Con l'impiego solo delle regole *F-C* e *F-S* si possono derivare $[x_1 \in N_1 \ldots\ x_n \in N_1]$ cont e ottenere, così, contesti di una lista arbitraria di variabili diverse appartenenti a N_1, come si vede dall'esempio seguente.

$$\cfrac{\cfrac{\cfrac{[\,]\ \text{cont}}{[x_1 \in N_1]\ \text{cont}}\ \text{F-C)}\ x_1 \in N_1 \notin \varnothing}{N_1\ \text{type}\ [x_1 \in N_1]}\ \text{F-S)}}{[x_1 \in N_1, x_2 \in N_1]\ \text{cont}}\ \text{F-C)}\ x_2 \in N_1 \notin x_1 \in N_1$$

2.2.2 Regole di Introduzione

$$\text{I-S)}\ \frac{[\Gamma]\ \text{cont}}{* \in N_1\ [\Gamma]\ \text{cont}}$$

Sia N_1 in ogni contesto Γ, partendo da contesto $\varnothing$, la regola I-S permette di formare i termini, per mezzo dell'introduzione di un elemento costante $*$ in N_1.

Un esempio diretto della sua applicazione è

$$\text{I-S)}\ \frac{[\ ]\ \text{cont}}{* \in N_1\ (x_1 \in N_1 ... x_n \in N_1)}$$

2.2.3 Regole di Eliminazione

$$\text{E-S)}\ \frac{\text{t} \in N_1\ [\Gamma] \qquad \text{M(z) type}[\Gamma, \text{z} \in N_1] \qquad \text{c} \in \text{M}(*)[\Gamma]}{El_{N1}(\text{t, c}) \in \text{M}(*)[\Gamma]}$$

El trattasi di costruttore di funzioni e $M[t] = M(z)[\frac{z}{t}]$.

2.2.4 Regole di Conversione

$$\text{C-S)}\ \frac{\text{M(z) type}[\Gamma, \text{z} \in N_1] \qquad \text{c} \in \text{M}(*)[\Gamma]}{El_{N1}(*, \text{c}) = \text{c} \in \text{M}(*)[\Gamma]}$$

La conversione rende possibile l'applicazione della regola di eliminazione introducendo delle uguaglianze.

Le regole *(S), (I-S), (E-S)* e *(C-S)* hanno una spiegazione computazionale, e riguardano la compatibilità tra tipi, ma non da caratterizzare il tipo dei tipi. Inoltre il tipo singoletto non è dipendente.

2.2.5 Eliminatore dipendente

La regola di eliminazione si può equivalentemente scrivere in un altro modo

$$\text{E-S)}_{dip}\ \frac{\text{M(z) type}[\Gamma, \text{z} \in N_1] \qquad \text{c} \in \text{M}(*)[\Gamma]}{El_{N1}(\text{z, c}) \in \text{M(z)}[\Gamma, \text{z} \in N_1]}$$

Le regole *E-S)$_{dip}$* + *la regole di sostituzione* + *F-S* + *I-S* permettono di verificare la validità di *E-S*.

$$\text{sost)}\ \frac{\text{t} \in N_1[\Gamma] \qquad \text{E-S)}_{dip}\ \dfrac{\text{M(z) type}[\Gamma, \text{z} \in N_1] \qquad \text{c} \in \text{M}(*)[\Gamma]}{El_{N1}(\text{z, c}) \in \text{M(z)}[\Gamma]}}{El_{N1}(\text{t, c}) \in \text{M(t)}[\Gamma]}$$

Inoltre vale anche il viceversa, da *E-S* si riesce a ottenere *E-S$_{dip}$*.

2.2.6 Osservazioni sul tipo singoletto

L'eliminatore $\mathrm{El}_{N1}(\mathrm{z}, \mathrm{c})$ rappresenta una funzione definita per ricorsione su N_1, difatti in *C-S* si ha che $\mathrm{El}_{N1}(\mathrm{z}, \mathrm{c})[\frac{z}{*}] = \mathrm{El}_{N1}(*, \mathrm{c})$.
Supposto che se $* \in \mathrm{N}_1[\Gamma]$ in *E-S*, allora per la singola conversione vale che $\mathrm{El}_{N1} = \mathrm{c} \in \mathrm{M}(*)$. Dunque $\mathrm{El}_{N1}(\mathrm{z}, \mathrm{c})$ rappresenta un programma funzionale per ricorsione. Questi è definito su N_1, a partire da $\mathrm{c} \in \mathrm{M}(*)$, perciò $\mathrm{El}_{N1}(*, \mathrm{c}) = \mathrm{c}$.
La regola di eliminazione permette di definire un programma funzionale da N_1 a M(z) esclusivamente con $\mathrm{c} \in \mathrm{M}(*)$, ovvero definendo $*$ come **elemento canonico**. Inoltre non svolge solo il compito di ricorsione, ma anche d'nduzione.

In

$$\frac{\mathrm{t} \in \mathrm{N}_1[\Gamma]}{\mathrm{t} = * \in \mathrm{N}_1[\Gamma]}$$

risulta vera l'uguaglianza definizionale?
No, non è vera. La regola di eliminazione consente di dare un valore al termine canonico, permettendo così di attribuire un valore a tutti i possibili termini del singoletto. Ma in generale questo non vale all'interno della teoria. Difatti l'uguaglianza definizionale è diversa da quella matematica e va intesa come computazionale e non proposizionale (come definito in §1.7.2).

2.3 Sanitary checks rules

Le *Sanitary checks* sono regole strutturali utili per abbreviare le derivazioni. Queste sono derivabili solo se la teoria dei tipi è corretta.

Assumento T, come una teoria dei tipi di riferimento, le *sanitary checks* sono le seguenti:

$$\frac{[\Gamma, \Delta]\ \mathrm{cont}}{\Gamma\ \mathrm{cont}}$$

Se $[\Gamma, \Delta]$ cont è derivabile in T allora anche $[\Gamma]$ cont è derivabile in T.

$$\frac{\mathrm{J}_1,\ldots,\mathrm{J}_n}{\mathrm{J}}$$

Se J_i con i = 1,...,n in T sono derivabili allora anche J è derivabile in T.

$$\frac{\mathrm{A\ type}\ [\Gamma]}{\Gamma\ \mathrm{cont}}$$

Se A type $[\Gamma]$ è derivabile in T allora anche Γ cont è derivabile in T.

$$\frac{\mathrm{A} = \mathrm{B\ type}\ [\Gamma]}{\mathrm{A\ type}[\Gamma]} \quad \frac{\mathrm{A} = \mathrm{B\ type}\ [\Gamma]}{\mathrm{B\ type}[\Gamma]}$$

Se A = B type $[\Gamma]$ è derivabile in T allora anche A type$[\Gamma]$ e B type$[\Gamma]$ sono derivabili in T.

$$\frac{\mathrm{a} \in \mathrm{A}[\Gamma]}{\mathrm{A\ type}[\Gamma]}$$

Se a $\in$ A[Γ] è derivabile in T allora anche A type[Γ] è derivabile in T.

$$\frac{\mathrm{a} = \mathrm{b} \in \mathrm{A}[\Gamma]}{\mathrm{a} \in \mathrm{A}[\Gamma]} \quad \frac{\mathrm{a} = \mathrm{b} \in \mathrm{A}[\Gamma]}{\mathrm{b} \in \mathrm{A}[\Gamma]}$$

Se a = b $\in$ A[Γ] è derivabile in T allora anche a $\in$ A[Γ] e b $\in$ A[Γ] sono derivabili in T.

2.4 Schema generale

Di seguito illustro uno schema generale, valido per ogni teoria dei tipi, di produzione di regole definenti un tipo, i suoi termini e l'uguaglianza.

1. **Regole di Formazione** Identificate con il preambolo F-T, con T teoria dei tipi in esame e t\ elemento canonico.
   Tali regole sono del tipo k e rispettano la forma $\dfrac{[\Gamma]\ \mathrm{cont}}{T\ \mathrm{type}[\Gamma]}$

2. **Regole di Introduzione** identificate con il preambolo I-T, con T teoria dei tipi in esame e t\ elemento canonico.
   Tali regole consistono nella forma introduttiva degli elementi canonici di T e rispettono la forma $\dfrac{[\Gamma]\ \mathrm{cont}}{\mathrm{t}\backslash \in T[\Gamma]}$

3. **Regole di Eliminazione** identificate con il preambolo E-T con T teoria dei tipi in esame e t\ elemento canonico.
   Tali regole sono definenti E_k, a partire dagli elementi di k a valori in un tipo M(z) type[Γ, z $\in$k]. L'ipotesi valida è che siano dati degli elementi in M(z) sui valori canonici di k. Tali regole sono del tipo k e rispettano la forma $\dfrac{\mathrm{t} \in T[\Gamma] \qquad \mathrm{M(z)\ type}[\Gamma, \mathrm{z} \in T] \qquad \mathrm{c} \in \mathrm{M(t\backslash)}[\Gamma]}{El_T(\mathrm{t},\ \mathrm{c}) \in \mathrm{M(t)}[\Gamma]}$

4. **Regole di Conversione** identificate con il preambolo C-T, con T teoria dei tipi in esame e t\ elemento canonico.
   , che stabilscono che gli eliminatori in (3) sono stabiliti per ricorsione a partire dalle ipotesi. Tali regole sono del tipo k e rispettano la forma $\dfrac{\mathrm{M(z)\ type}[\Gamma, \mathrm{z} \in T] \qquad \mathrm{c} \in \mathrm{M(t\backslash)}[\Gamma]}{\mathrm{El}_T(\mathrm{t}\backslash,\ \mathrm{c}) = \mathrm{c} \in \mathrm{M(t\backslash)}[\Gamma]}$

5. **Regole di Uguaglianza** identificate con il preambolo eq-E-T, con T teoria dei tipi in esame e t\ elemento canonico. Tali regole stabiliscono che i costruttori di k in (2) e (3) permettono l'uguaglianza definizionale dei termini da cui dipendono. Tali regole sono del tipo k e rispettano la forma $\dfrac{\mathrm{t} = \mathrm{s} \in \mathrm{N}_1[\Gamma] \qquad \mathrm{M(z)\ type}[\Gamma, \mathrm{z} \in \mathrm{N}_1] \qquad \mathrm{c} = \mathrm{d} \in \mathrm{M(t\backslash)}[\Gamma]}{\mathrm{El}_T(\mathrm{t}\backslash,\ \mathrm{c}) = \mathrm{c} \in \mathrm{M(t\backslash)}[\Gamma]}$

Le regole (5) sono implicite in T, ma non ovvie dal punto di vista computazionale.

2.5 Uguaglianza definizionale

Definizione: se P_1 e P_2 programmi

P_1, P_2: $Nat^m \to Nat$ allora $P_1 = P_2$ sse $\forall\ n_1 \ldots n_m$ e $P_1(n_1 \ldots n_m) = P_2(n_1 \ldots n_m)$ non è decidibile.

Ovvero le funzioni ricorsive per P_1 e P_2 non sono decidibili. A seguito di ciò non esiste un algoritmo in grado di decidere se due programmi P_1 e P_2 sono estensionalmente (proposizionalmente) uguali o meno.

Risulta però vero il concetto di **ugualianza definizionale**/computazionale (in teoria dei tipi intensionali). Dati a $\in$ A[Γ] e b $\in$ A[Γ] derivabili, nella nostra teoria T di *Martin-Löf*, esiste un algoritmo H (a $\in$ A[Γ], b $\in$ A[Γ]) = $\begin{cases} \textbf{sì} \text{ sse a} = \text{b} \in A[\Gamma] \text{ è derivabile in } T \\ \textbf{no} \text{ sse a} = \text{b} \in A[\Gamma] \text{ non ha derivazione in } T \end{cases}$ Il giudizio a = b $\in$ A[Γ] è decidibile, anche con J giuduzio, in teoria dei tipi di *Martin-Löf*, derivabile. Percui con H si scrive: Giudizi di $T \to \{0,1\}$

H[J] = $\begin{cases} \textbf{1} \text{ sse } J \text{ è derivabile in } T \\ \textbf{0} \text{ sse } J \text{ non è derivabile in } T \end{cases}$

H lavora come un *proof assistant* (esempio COQ).

2.5.1 Applicazione dell'uguaglianza definizionale tra termini

Definzione: i termini *untyped* sono

$$\text{t} = \text{x} \mid * \mid \text{El}_{N1}(\text{t}_1, \text{t}_2)$$

Definizione: relazione di riduzione

$$\to_1 \subseteq \text{term x term}$$

$\text{t}_1 \to_1 \text{t}_2 \equiv \text{t}_1$ si riduce in un passo di computazione a t_2. Ecco che esiste una relazione che computa t_1 con t_2.
Le **relazioni di computazione**, dell'uguaglianza definizionale, le ho descritte in §2.6.

Definizione: t termine *untyped* è in forma normale (in NF) sse non esiste termine s tale che $\text{t} \to_1 \text{s}$. Ovvero non è più riducibile a nulla.
Le forme normali sono i valori assumibili dai programmi.

Definzione:**Teoria di Validità**
Dati t $\in$ A[Γ] e s $\in$ A[Γ] in T_1 allora $\to_1 \text{s} \Rightarrow \text{t} = \text{s} \in$ A[Γ] derivabile in T.

termini	termini in forma canonica	termini non in forma canonica o introduttiva
termini in NF	$*$	x, El_{N1}(x, $*$)
termini non in NF	$\varnothing$	$\mathrm{El}_{N1}(*, \mathrm{x})$

Tabella 2.1: Termini a $\rightarrow_1$ di N_1.

I termini non in forma canonica derivano dalle regole di introduzione; invece quelli non in forma canonica vengono introdotte dall'eliminatore.

La chiusura riflessiva, simmetrica e transitiva delle derivazioni è proprio l'uguaglianza definizionale. Tale proprietà non vale esclusivamente per $\rightarrow_1$ ma per qualsiasi combinazione delle riduzioni in §2.6, con variazione però delle forma normali (che non sono altro che i risultati dei nostri programmi, derivanti dai diversi cammini di computazione).
Un esempio significativo di applicazione di strategie di computazione l'ho riportato in §2.7, esercizio 4. Utile per comprendere a cosa serve la relazione $\rightarrow_1$ e che ogni teoria $T \rightarrow_1$ non è deterministica.

Le definizioni seguenti sono definite sulla regole di semantica operazionale.

Definizione **Riducibilità**
Dati i termini t e s allora **t** $\boldsymbol{Red}_{NF}$ **s** sse s è in NF ed esistono $h_1 ... h_n$ (n>=1) tale che $h_1 \equiv$ t, $h_n \equiv$ s e se n>1 $h_i \rightarrow h_{i+1}$ per i = 1 a n-1.

$$\mathbf{t}\ \boldsymbol{Rid}_{NF}\ \mathbf{s} = \begin{cases} \mathbf{t} \equiv \text{s e t è in NF} \\ \text{esiste n>1, esistono } h_1 ... h_n \text{ termini tale che } \mathbf{t} \equiv h_1 \rightarrow_1 h_2 ... h_n \end{cases}$$

Definizione **Teorema di Confluenza**$\rightarrow_1$ per T computabile
Dato t (termine) e s_1 e s_2 (in NF) tale che **t** $\boldsymbol{Red}_{NF}$ $\mathbf{s}_1$ e **t** $\boldsymbol{Red}_{NF}$ $\mathbf{s}_2$ allora $s_1 \equiv s_2$ (coincide a meno di rinomina di variabile vincolante).
Quando t si riduce s_1 e in s_2 c'e' l'unicità della forma normale.

Definizione **Teorema della forma normale (debole)**
Dato t termine della grammatica esiste s termine in NF tale che **t** $\boldsymbol{Red}_{NF}$ **s**. Allora esistono $t \equiv h_1 \rightarrow_1 ... \rightarrow_1 h_n$ con n>1 se t non è già in NF; oppure t ≡ s se t è già in NF.
Questo significa che è sempre possibile rendere un programma convergente. Ma si può dire di più: ∄ programmi che divergono.

Definizione **Teorema della forma normale (forte)**
Per ogni termine t, l'albero dei cammini di riduzione di t è ben formato (ovvero ∄ un cammino di riduzioni $\rightarrow_1$ infinito).
In questo modo ogni strategia deterministica è convergente.
Con quanto appena enunciato sopra possiamo denfinire quanto segue.
Dato t ∈ A[Γ] derivabile in T

$$\mathrm{NF}(t_1) \equiv \begin{cases} \mathbf{t} \text{ se t è in NF} \\ \mathbf{s} \text{ se } \boldsymbol{t\ Red}_{NF}\ \boldsymbol{s} \end{cases}$$

Dunque se t non é in NF per il teorema normale comunque esiste una riduzione in NF.
Sono così in grado di dimostrare che, dati a $\in$ A[Γ] e b $\in$ A[Γ], giudizi derivabili in T_i allora a = b $\in$ A[Γ] sse NF(a) $\equiv$ NF(b) sse

1. a e b sono in NF e quindi a $\equiv$ b
2. a non in NF, b in NF e **a** $\boldsymbol{Red}_{NF}$ **b**
3. a in NF, b non in NF e **b** $\boldsymbol{Red}_{NF}$ **a**
4. né a né b sono in NF esiste s in NF tale che **a** $\boldsymbol{Red}_{NF}$ **s** e **b** $\boldsymbol{Red}_{NF}$ **s**

Per i punti elencanti sopra trova validità la relazione **a** $\boldsymbol{Red}_{NF}$ **NF(a)** $\Rightarrow$ a = NF(a) $\in$ A[Γ] è derivabile (la forma normale è uguale al termine stesso).
Questo rende l'uguaglianza computabile, si è difatti in grado di dimostrare che esiste P programma tale che P(a) = NF(a), per ogni a termine *untyped* in T(incluso T_1).

In conclusione la computabilità dell'uguaglianza (uguaglianza definizionale) tra due termini, si riduce a computare le forme normali del primo termine con quelle del secondo e a verificare se sono identicamente la stessa (a meno di rinomia di variabili).

2.6 Semantica operazionale del singoletto

La relazione $\rightarrow_1$ viene definita all'interno dei termini con l'uso delle seguenti regole di riduzione:

- β_{N1}-red) $\mathrm{El}_{N1}(*, \mathrm{t}) \rightarrow_1 \mathrm{t}$
- red_I) $\dfrac{\mathrm{t}_1 \rightarrow_1 \mathrm{t}_2}{\mathrm{El}_{N1}(\mathrm{t}_1, \mathrm{c}) \rightarrow_1 \mathrm{El}_{N1}(\mathrm{t}_2, \mathrm{c})}$ $\qquad$ red_{II}) $\dfrac{\mathrm{c}_1 \rightarrow_1 \mathrm{c}_2}{\mathrm{El}_{N1}(\mathrm{t}, \mathrm{c}_1) \rightarrow_1 \mathrm{El}_{N1}(\mathrm{t}, \mathrm{c}_2)}$
- red_I e red_{II} possono venire simultate da un'unica regola $\dfrac{\mathrm{t}_1 \rightarrow_1 \mathrm{t}_2 \qquad \mathrm{c}_1 \rightarrow_1 \mathrm{c}_2}{\mathrm{El}_{N1}(\mathrm{t}_1, \mathrm{c}_1) \rightarrow_1 \mathrm{El}_{N1}(\mathrm{t}_2, \mathrm{c}_2)}$

β_{N1}-*red* risulta valida per *C-S*, le regole di riduzione red_I e red_{II} per *eq-E-S*.

2.7 Esercizi

2.7.1 Tipo singoletto

1 Data

$$\text{E-N}_{1prog}) \ \frac{\mathrm{M(w)\ type\ [\Gamma,\ w \in N_1]} \qquad \mathrm{d \in M(*)\ type[\Gamma]}}{\mathrm{El}_{N1}(\mathrm{w},\ \mathrm{d}) \in \mathrm{M(w)\ type[\Gamma,\ w \in N_1]}}$$

dimostrare che in $\boldsymbol{T}_1$ la regola *E-N*$_1$*prog* è derivabile. Al fine di ciò basta mostrare che se i giudizi premessa sono derivabili, allora lo è anche il giudizio di conclusione.

Soluzione

Per una maggiore comprensione delle derivazioni, ho ritenuto opportuno, ove necessario, spezzare l'albero in più parti.

$$\text{E-S}\ \frac{\text{s-checks}\ \dfrac{\dfrac{}{\text{M(w) type}[\Gamma,\ \text{w} \in \text{N}_1]}}{[\Gamma,\ \text{w} \in \text{N}_1]\ \text{cont}}}{\text{w} \in \text{N}_1[\Gamma,\ \text{w} \in \text{N}_1]} \qquad \frac{}{\text{M(z) type}[\Gamma,\ \text{z} \in \text{N}_1,\ \text{w} \in \text{N}_1]} \qquad \frac{}{\text{d} \in \text{M}(*)[\Gamma,\ \text{w} \in \text{N}_1]}}{\text{El}_{N1}(\text{w},\ \text{d}) \in \text{M(w)}[\Gamma,\ \text{w} \in \text{N}_1]}$$

$$\text{ind-ty}\ \frac{\dfrac{}{\text{M(w) type}[\Gamma,\ \text{w} \in \text{N}_1]} \qquad \text{F-c}\ \dfrac{\text{s-checks}\ \dfrac{\dfrac{}{\text{M(w) type}[\Gamma,\ \text{w} \in \text{N}_1]}}{[\Gamma,\ \text{w} \in \text{N}_1]\ \text{cont}}}{\text{F-S}\ \dfrac{}{\text{N}_1\ \text{type}[\Gamma,\ \text{w} \in \text{N}_1]}}}{[\Gamma,\ \text{z} \in \text{N}_1,\ \text{w} \in N_1]\ \text{cont}}\ (\text{z} \in \text{N}_1) \notin \Gamma}{\text{sub-typ}\ \dfrac{\text{M(w) type}[\Gamma,\ \text{z} \in \text{N}_1,\ \text{w} \in N_1]}{\text{M(z) type}[\Gamma,\ \text{z} \in \text{N}_1,\ \text{w} \in \text{N}_1]} \qquad \text{F-c}\ \dfrac{\text{s-checks}\ \dfrac{\dfrac{}{\text{M(w) type}[\Gamma,\ \text{w} \in \text{N}_1]}}{[\Gamma,\ \text{w} \in \text{N}_1]\ \text{cont}}}{\text{F-S}\ \dfrac{}{\text{N}_1\ \text{type}\ [\Gamma,\ \text{w} \in \text{N}_1]}}}{[\Gamma,\ \text{z} \in \text{N}_1,\ \text{w} \in \text{N}_1]\ \text{cont}}\ (\text{z} \in \text{N}_1) \notin \Gamma}{\text{z} \in N_1[\Gamma,\ \text{z} \in N_1,\ \text{w} \in \text{N}_1]}\ \text{var}}$$

Assumo che le premesse di *E-N*$_1$*prog* (*M(w) type [*Γ*,w* $\in$ *N*$_1$*]* e *d* $\in$ *M(*$*$*)[*Γ*]*) siano valide, pèrciò è valido, dalla prova sopra, anche il giudizio di conclusione *El*$_{N1}$*(w, d)* $\in$ *M(w)[*Γ*, w* $\in$ *N*$_1$*]*, di conseguenza derivabile in T_1.

2. Dimostrare che la regola *E-S* è derivabile in una teoria dei tipi T_1, in cui si è rimpiazziata la regola di eliminazione *E-S* con la regola *E-N*$_1$*prog*, aggiungendovi le regole di indebolimento, sostituzione e di *sanitary checks*.

$$\text{E-N}_{1prog})\ \frac{\text{M(w) type}\ [\Gamma,\ \text{w} \in \text{N}_1] \qquad \text{d} \in \text{M}(*)[\Gamma]}{\text{El}_{N1}(\text{w},\ \text{d}) \in \text{M(w)}[\Gamma,\ \text{w} \in \text{N}_1]}$$

$$\text{E-S)}\ \frac{\text{t}\setminus \in \text{N}_1[\textstyle\sum] \qquad \text{M(w) type}\ [\Gamma,\ \text{w} \in \text{N}_1] \qquad \text{d}\setminus \in \text{M}(*)[\textstyle\sum]}{\text{El}_{N1}(\text{t}\setminus,\ \text{d}\setminus) \in \text{M(t}\setminus)[\Gamma]}$$

Soluzione

Idea: parto dalla regola di eliminazione E-S, vi applico la regola di sostituzione sub-typ giungendo così alle premessi di E-N$_{1prog}$

$$\text{sub-typ}\ \frac{\dfrac{}{\text{t}\setminus \in \text{N}_1[\Gamma]} \qquad \text{E-N}_{1prog}\ \dfrac{\dfrac{}{\text{M(w) type}\ [\Gamma,\ \text{w} \in \text{N}_1]} \qquad \dfrac{}{\text{d} \in \text{D}(*)[\Gamma]}}{\text{El}_{N1}(\text{w},\ \text{d}\setminus) \in \text{M(w)}[\Gamma,\ \text{w} \in \text{N}_1]}}{\text{El}_{N1}(\text{t}\setminus,\ \text{d}\setminus) \in \text{M(t}\setminus)[\Gamma]}$$

Assumo che siano valide per costruzione le premesse di *E-N*$_{1prog}$ (come dimostro nell'esercizio 2) e di *E-S*.

3 Sia T_1 la teoria dei tipi definita del tipo singoletto con le regole strutturali, definite in questo capitolo, incluse quelle di sostituzione e indebolimento. Allora stabilire se i seguenti termini sono tipabili come termini del tipo singoletto, secondo T_1 e quali sono uguali definizionalmente.

- $\mathrm{El}_{N1}(*, *)$
- $\mathrm{El}_{N1}(\mathrm{x}, *)$
- $\mathrm{El}_{N1}(*, \mathrm{y})$
- $\mathrm{El}_{N1}(\mathrm{x}, \mathrm{y})$
- $\mathrm{El}_{N1}(\mathrm{El}_{N1}(*, \mathrm{y}), \mathrm{El}_{N1}(\mathrm{x}, *))$

Soluzione

Per una maggiore comprensione delle derivazioni, ho ritenuto opportuno, ove necessario, spezzare l'albero in più parti.

1

$$\dfrac{\text{I-S}\dfrac{\dfrac{[\,]\ \mathrm{cont}}{*\in \mathrm{N1}[\,]}}{\text{E-S}} \qquad \dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{[\,]\ \mathrm{cont}}{\mathrm{N_1\ type}[\,]}}{[\mathrm{z}\in \mathrm{N_1}]\ \mathrm{cont}}(\mathrm{z}\in \mathrm{N_1})\notin[\,]}{\text{I-S}\dfrac{}{\mathrm{N_1\ type}[\mathrm{z}\in \mathrm{N_1}]}} \qquad \text{I-S}\dfrac{[\,]\ \mathrm{cont}}{*\in \mathrm{N_1}[\,]}}{\mathrm{El}_{N1}(*, *)\in N_1[\,]}$$

Applicando la β $N_1 red$ allora $\mathrm{El}_{N1}(*, *) \to_1 *$
$\mathrm{El}_{N1}(*, *)$ è uguale definizionalmente.

2

$$\dfrac{\text{I-S}\dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{[\,]\ \mathrm{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{x}\in \mathrm{N_1\ cont}}(\mathrm{x}\in \mathrm{N_1})\notin[\,]}{*\in \mathrm{N_1}[\mathrm{x}\in \mathrm{N_1}]}}{\text{E-S}} \qquad \dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{[\,]\ \mathrm{cont}}{\mathrm{N_1\ type}\,[\,]}}{\mathrm{x}\in \mathrm{N_1\ cont}}(\mathrm{x}\in \mathrm{N_1})\notin[\,]}{\mathrm{N_1\ type}[\mathrm{x}\in \mathrm{N_1}]}}{\mathrm{x}\in \mathrm{N_1}, \mathrm{z}\in \mathrm{N_1\ cont}}(\mathrm{z}\in \mathrm{N_1})\notin(\mathrm{x}\in \mathrm{N_1})}{\mathrm{N_1\ type}[\mathrm{x}\in \mathrm{N_1}, \mathrm{z}\in \mathrm{N_1}]} \qquad \text{var}\dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{[\,]\ \mathrm{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{x}\in \mathrm{N_1\ cont}}(\mathrm{x}\in \mathrm{N_1})\notin[\,]}{\mathrm{x}\in \mathrm{N_1}[\mathrm{x}\in \mathrm{N_1}]}}{\mathrm{El}_{N1}(\mathrm{x}, *)\in \mathrm{N_1}[\mathrm{x}\in \mathrm{N_1}]}$$

Applicando la β $N_1 red$ allora $\mathrm{El}_{N1}(\mathrm{x}, *) \twoheadrightarrow_1$
$\mathrm{El}_{N1}(\mathrm{x}, *)$ non è uguale definizionalmente.

3

$$\dfrac{\text{I-S}\dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{[\,]\ \mathrm{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{y}\in \mathrm{N_1\ cont}}(\mathrm{y}\in \mathrm{N_1})\notin[\,]}{*\in \mathrm{N_1}[\mathrm{y}\in \mathrm{N_1}]}}{\text{E-S}} \qquad \dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{[\,]\ \mathrm{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{y}\in \mathrm{N_1\ cont}}(\mathrm{y}\in \mathrm{N_1})\notin[\,]}{\mathrm{N_1\ type}[\mathrm{y}\in \mathrm{N_1}]}}{\mathrm{y}\in \mathrm{N_1}, \mathrm{z}\in \mathrm{N_1\ cont}}(\mathrm{z}\in \mathrm{N_1})\notin \mathrm{y}\in \mathrm{N_1}}{\mathrm{N_1\ type}[\mathrm{y}\in \mathrm{N_1}, \mathrm{z}\in \mathrm{N_1}]} \qquad \text{var}\dfrac{\text{F-c}\dfrac{\text{F-S}\dfrac{[\,]\ \mathrm{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{y}\in \mathrm{N_1\ cont}}(\mathrm{y}\in \mathrm{N_1})\notin[\,]}{\mathrm{y}\in \mathrm{N_1}[\mathrm{y}\in \mathrm{N_1}]}}{\mathrm{El}_{N1}(*, \mathrm{y})\in \mathrm{N_1}[\mathrm{y}\in \mathrm{N_1}]}$$

Applicando la β $N_1 red$ allora $\mathrm{El}_{N1}(*, \mathrm{y}) \to_1 \mathrm{y}$
$\mathrm{El}_{N1}(*, \mathrm{y})$ è uguale definizionalmente.

4

$$\text{E-S}\ \frac{\text{var}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{[\,]\ \text{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{x \in N_1\ cont}}\ (\mathrm{x} \in \mathrm{N_1}) \notin [\,]}{\mathrm{N_1\ type[x \in N_1]}}}{\mathrm{x \in N_1, y \in N_1\ cont}}\ (\mathrm{y} \in \mathrm{N_1}) \notin \mathrm{x} \in \mathrm{N_1}}{\mathrm{x \in N_1[x \in N_1, y \in N_1]}} \quad \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{[\,]\ \text{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{x \in N_1\ cont}}\ (\mathrm{x} \in \mathrm{N_1}) \notin [\,]}{\mathrm{N_1\ type[x \in N_1]}}}{\mathrm{x \in N_1, y \in N_1\ cont}}\ (\mathrm{y} \in \mathrm{N_1}) \notin (\mathrm{x} \in \mathrm{N_1})}{\mathrm{N_1\ type[x \in N_1, y \in N_1]}}}{\mathrm{x \in N_1, y \in N_1, z \in N_1\ cont}}\ (\mathrm{z} \in \mathrm{N_1}) \notin (\mathrm{x} \in \mathrm{N_1}, \mathrm{y} \in \mathrm{N_1}) \quad \dfrac{\text{var}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{[\,]\ \text{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{x \in N_1\ cont}}\ (\mathrm{x} \in \mathrm{N_1}) \notin [\,]}{\mathrm{N_1\ type[x \in N_1]}}}{\mathrm{x \in N_1, y \in N_1\ cont}}\ (\mathrm{y} \in \mathrm{N_1}) \notin \mathrm{x} \in \mathrm{N_1}}{\mathrm{y \in N_1[x \in N_1, y \in N_1]}}}{\mathrm{El}_{N1}(\mathrm{x}, \mathrm{y}) \in \mathrm{N_1[x \in N_1, y \in N_1]}}$$

Applicando la β $N_1 red$ allora $\mathrm{El}_{N1}(\mathrm{x}, \mathrm{y}) \twoheadrightarrow_1$
$\mathrm{El}_{N1}(*, \mathrm{y})$ non è uguale definizionalmente.

5

$$\text{E-S}\ \frac{\mathrm{El}_{N1}(*, \mathrm{y}) \in \mathrm{N_1[y \in N_1, x \in N_1]} \quad \mathrm{N_1\ type[y \in N_1, x \in N_1, z \in N_1]} \quad \mathrm{El}_{N1}(\mathrm{x}, *) \in \mathrm{N_1[y \in N_1, x \in N_1]}}{\mathrm{El}_{N1}(\mathrm{El}_{N1}(*, \mathrm{y}), \mathrm{El}_{N1}(\mathrm{x}, *)) \in \mathrm{N_1[y \in N_1, x \in N_1]}}$$

$$\text{ind-te}\ \frac{\mathrm{El}_{N1}(*, \mathrm{y}) \in \mathrm{N_1[y \in N_1]} \quad \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{[\,]\ \text{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{y \in N_1\ cont}}\ (\mathrm{y} \in \mathrm{N_1}) \notin [\,]}{\mathrm{N_1\ type[y \in N_1]}}}{\mathrm{y \in N_1, x \in N_1\ cont}}\ (\mathrm{x} \in \mathrm{N_1}) \notin (\mathrm{y} \in \mathrm{N_1})}{\mathrm{El}_{N1}(*, \mathrm{y}) \in \mathrm{N_1[y \in N_1, x \in N_1]}}$$

$$\text{F-S}\ \frac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{[\,]\ \text{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{y \in N_1\ cont}}\ (\mathrm{y} \in \mathrm{N_1}) \notin [\,]}{\mathrm{N_1\ type[y \in N_1]}}}{\mathrm{y \in N_1, x \in N_1\ cont}}\ (\mathrm{x} \in \mathrm{N_1}) \notin (\mathrm{y} \in \mathrm{N_1})}{\mathrm{N_1\ type[y \in N_1, x \in N_1]}}}{\mathrm{y \in N_1, x \in N_1, z \in N_1\ cont}}\ (\mathrm{z} \in \mathrm{N_1}) \notin (\mathrm{y} \in \mathrm{N_1}, \mathrm{x} \in \mathrm{N_1})}{\mathrm{N_1\ type[y \in N_1, x \in N_1, z \in N_1]}}$$

$$\text{ex-te}\ \frac{\text{ind-te}\ \dfrac{\mathrm{El}_{N1}(\mathrm{x}, *) \in \mathrm{N_1[x \in N_1]} \quad \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{[\,]\ \text{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{x \in N_1\ cont}}\ (\mathrm{x} \in \mathrm{N_1}) \notin [\,]}{\mathrm{N_1\ type[x \in N_1]}}}{\mathrm{x \in N_1, y \in N_1\ cont}}\ (\mathrm{y} \in \mathrm{N_1}) \notin \mathrm{x} \in \mathrm{N_1}}{\mathrm{El}_{N1}(\mathrm{x}, *) \in \mathrm{N_1[x \in N_1, y \in N_1]}} \quad \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{\text{F-c}\ \dfrac{\text{F-S}\ \dfrac{[\,]\ \text{cont}}{\mathrm{N_1\ type}[\,]}}{\mathrm{y \in N_1\ count}}\ (\mathrm{y} \in \mathrm{N_1}) \notin \mathrm{y} \in [\,]}{\mathrm{N_1\ type[y \in N_1]}}}{\mathrm{y \in N_1, x \in N_1\ cont}}\ (\mathrm{x} \in \mathrm{N_1}) \notin \mathrm{y} \in \mathrm{N_1}}{\mathrm{El}_{N1}(\mathrm{x}, *) \in \mathrm{N_1[y \in N_1, x \in N_1]}}$$

Per i giudizi conclusione $\mathrm{El}_{N1}(*, \mathrm{y}) \in \mathrm{N}_1[\mathrm{y} \in \mathrm{N}_1]$ e $\mathrm{El}_{N1}(\mathrm{x}, *) \in \mathrm{N}_1[\mathrm{x} \in \mathrm{N}_1]$ ho già dimostrato sopra (in 3 e 2) la loro tipabilità per il tipo singoletto.
Applicando la *red*$_I$ e β N_1*red* allora $\mathrm{El}_{N1}(\mathrm{El}_{N1}(*, \mathrm{y}), \mathrm{El}_{N1}(\mathrm{x}, *)) \to_1 \mathrm{El}_{N1}(\mathrm{y}, \mathrm{El}_{N1}(\mathrm{x}, *))$.
Più nel dettaglio la riduzione è la seguente

$$\mathrm{red}_I \dfrac{\beta\ \mathrm{N_1red}\ \dfrac{}{\mathrm{El}_{N1}(*, \mathrm{y}) \to_1 \mathrm{y}}}{\mathrm{El}_{N1}(\mathrm{El}_{N1}(*, \mathrm{y}), \mathrm{El}_{N1}(\mathrm{x}, *)) \to_1 \mathrm{El}_{N1}(\mathrm{y}, \mathrm{El}_{N1}(\mathrm{x}, *))}$$

$\mathrm{El}_{N1}(\mathrm{El}_{N1}(*, \mathrm{y}), \mathrm{El}_{N1}(\mathrm{x}, *))$ è uguale definizionalmente.

4 Dati i termini definiti dalla seguente grammatica relativa ai termini del tipo singoletto

$$\mathbf{t \equiv v \mid * \mid El_{N1}(t_1, t_2)}$$

con v $\in$ {x,y,w,z} $\bigcup$ {$\mathbf{x}_i \mid \mathbf{i} \in$ Nat}, ovvero considerando come variabili le ultime lettere dell'alfabeto inglese e poi tutte le variabili ottenute ponendo alla variabili x un indice che varia nei numeri naturali.
Sia $\to_1$ una relazione binaria tra questi termini *untyped* definita a partire dalle seguenti regole

$$\beta_{N1})\ \mathbf{red} \qquad \mathbf{El}_{Nat}(\mathbf{0,c}) \to_1 \mathbf{c}$$

$$\mathbf{red}_I) \dfrac{\mathbf{t_1 \to t_2}}{\mathbf{El}_{N1}(\mathbf{t_1, c}) \to_1 \mathbf{El}_{N1}(\mathbf{t_2, c})} \qquad \mathbf{red}_{II}) \dfrac{\mathbf{c_1 \to c_2}}{\mathbf{El}_{N1}(\mathbf{t, c_1}) \to_1 \mathbf{El}_{N1}(\mathbf{t, c_2})}$$

- **Costruire l'albero dei cammini (ovvero sequenze) di passi di riduzione possibili fino a un termine in forma normale, ovvero non ulteriormente riducibile rispetto alla relazione $\to_1$ del termine**

$$\mathbf{El}N_1(\mathbf{El}_{N1}(*, *), \mathbf{El}_{N1}(*, \mathbf{x}))$$

- **Produrre un infinità di termini del tipo singoletto che non sono riducibili secondo la relazione di un passo di riduzione$\to_1$.**
  Dati due di questi termini , si riesce a dire che sono definionalmente uguali secondo le regole del tipo singoletto?

Soluzione

Idea: uso un albero di derivazione per mostrare ogni passo derivazione di ogni cammino.
Se w = $El_{N1}(El_{N1}(*, *), El_{N1}(*, x))$ combino il lambda termine w con l'applicazione della strategia deterministica di riduzione ($\to_1$), con la quale il termine si riduce eventualmente a forma normale (implicando la definizione di riducibilità).
β-red:
$El_{N1}(*, *) \to_1 *$
$El_{N1}(*, x) \to_1 x$

$$\mathrm{red}_I \dfrac{\beta\text{-}\mathrm{N_1}\ \mathrm{red}\ \dfrac{\mathrm{x}}{\mathrm{El}_{N1}(*, \mathrm{x})} \quad \beta\text{-}\mathrm{N_1}\ \mathrm{red}\ \dfrac{\dfrac{\mathrm{x}}{\mathrm{El}_{N1}(*, \mathrm{x})}}{\mathrm{El}_{NI}(*, \mathrm{El}_{N1}(*, \mathrm{x}))}\ \mathrm{red}_{II}}{\mathrm{El}_{N1}(\mathrm{El}_{N1}(*, *), \mathrm{El}_{N1}(*, \mathrm{x}))}\ \mathrm{red}_{II} \qquad \dfrac{\dfrac{\mathrm{x}}{\mathrm{El}_{N1}(*, \mathrm{x})}\ \beta\text{-}\mathrm{N_1}\ \mathrm{red}}{\mathrm{El}_{N1}(\mathrm{El}_{N1}(*,*), \mathrm{x})}\ \mathrm{red}_I$$

$\Rightarrow$ (w, (red$_I I$, red$_I$, β_{N1red})) rappresenta un programma.

Termine t non più riducibile significa che è un termine *untyped* che è in forma normale perchè non esiste alcun altro termine s tale che t $\rightarrow_1$ s. Dunque l'infinità di termini singoletto, non più riducibili rispetterano la definizione data sopra

$$\mathrm{t} \equiv \begin{cases} v \\ * \\ El_{N1}(t_1, t_2) \end{cases}$$

Dati due termini t^I e t^{II} termini *untyped* non riesco a dire che sono definizionalmente uguali perchè già e in forma normale. Difatti per il teorema della forma normale forte vale $\rightarrow_0$.

Capitolo 3

Naturali, Liste e Somma disgiunta

3.1 Tipo dei numeri Naturali

3.1.1 Regole di Formazione

$$\text{F-Nat)}\ \frac{\Gamma\ \text{cont}}{\text{Nat type}[\Gamma]}$$

3.1.2 Regole di Introduzione

$$\text{I}_1\text{-Nat)}\ \frac{\Gamma\ \text{cont}}{0 \in \text{Nat}[\Gamma]} \qquad \text{I}_2\text{-Nat)}\ \frac{\text{m} \in \text{Nat}[\Gamma]}{\text{succ(m)} \in \text{Nat}[\Gamma]}$$

3.1.3 Regole di Eliminazione

$$\text{E-Nat)}\ \frac{\text{t} \in \text{Nat}[\Gamma] \qquad \text{M(z) type}[\Gamma,\ \text{z} \in \text{Nat}] \qquad \text{c} \in \text{M(0)}[\Gamma] \qquad \text{e(x,y)} \in \text{M(succ(x))}[\Gamma,\ \text{x} \in \text{Nat},\ \text{y} \in \text{M(x)}]}{\text{El}_{Nat}\text{(t,c,e)} \in \text{M(t)}[\Gamma]}$$

3.1.4 Regole di Conversione

$$\text{C}_1\text{-Nat)}\ \frac{\text{M(z) type}[\Gamma,\ \text{z} \in \text{Nat}] \qquad \text{c} \in \text{M(0)}[\Gamma] \qquad \text{e(x,y)} \in \text{M(succ(x))}[\Gamma,\ \text{x} \in \text{Nat},\ \text{y} \in \text{M(x)}]}{\text{El}_{Nat}\text{(0,c,e)} = \text{c} \in \text{M(0)}[\Gamma]}$$

$$\text{C}_2\text{-Nat)}\ \frac{\text{m} \in \text{Nat}[\Gamma] \qquad \text{M(z) type}[\Gamma,\ \text{z} \in \text{Nat}] \qquad \text{c} \in \text{M(0)}[\Gamma] \qquad \text{e(x,y)} \in \text{M(succ(x))}[\Gamma,\ \text{x} \in \text{Nat},\ \text{y} \in \text{M(x)}]}{\text{El}_{Nat}\text{(succ(m),c,e)} = \text{e(m, El}_{Nat}\text{(m,c,e))} \in \text{M(succ(m))}[\Gamma]}$$

3.1.5 Regole di Uguaglianza

$$\text{eq-Nat)}\ \frac{\text{t}_1 = \text{t}_2 \in \text{Nat}[\Gamma]}{\text{succ(t}_1\text{)} = \text{succ(t}_2\text{)} \in \text{Nat}[\Gamma]}$$

3.1.6 Introduzione ed Eliminatore dipendente

Le regole di formazione dei tipi e dei loro termini sono formulate in modo da rendere la regole si sostituzione per tipi e termini ammissibili.
Ad esempio la regola di introduzione del successore di un numero naturale si può formulare come un esplicito programma funzionale visto come termine dipendente.

$$\text{I}_2\text{-Nat}_{prog}) \ \frac{\Gamma \text{ cont}}{\text{succ(x)} \in \text{Nat}[\Gamma,\ \text{x} \in \text{Nat}]}$$

Il medesimo discorso vale per la regola di eliminazione

$$\text{E-Nat}_{dip}) \ \frac{\text{M(z) type}[\Gamma,\ \text{z} \in \text{Nat}] \qquad \text{c} \in \text{M(0)}[\Gamma] \qquad \text{e(x,y)} \in \text{M(succ(x))}[\Gamma,\ \text{x} \in \text{Nat},\ \text{y} \in \text{M(x)}]}{\text{El}_{Nat}\text{(w,c,e)} \in \text{M(t)}[\Gamma,\ \text{w} \in \text{Nat}]}$$

E-Nat è equivalente a *E-Nat*$_{dip}$. Difatti la teoria T_{N1Nat}, in cui c'è *E-Nat*, è equivalente a $T\backslash$ senza *E-Nat*, ma con *E-Nat*$_{dip}$, le regole di sosituzione e di *sanitary check*.

3.1.7 Primitiva ricorsiva

Definizione

$\text{Nat}^n \times \text{Nat} \to \text{Nat}$
Dati g_0: $\text{Nat}^m \to \text{Nat}$ e g_1: $\text{Nat}^m \times \text{Nat} \times \text{Nat} \to \text{Nat}$
$\text{n}_1 ... \text{n}_m \in \text{Nat}$ allora

$\text{rec}(\text{n}_1 ... \text{n}_m,\ 0) \equiv \text{g}_0(\text{n}_1 ... \text{n}_m)$
$\text{rec}(\text{n}_1 ... \text{n}_m,\ \text{k+1}) \equiv \text{g}_0(\text{n}_1 ... \text{n}_m,\ \text{k},\ \text{rec}(\text{n}_1 ... \text{n}_m,\ \text{k}))$

3.1.8 Semantica operazionale dei numeri naturali

La relazione $\to_1$ viene definita all'interno dei termini con l'uso delle seguenti regole di riduzione:

- β_{1Nat}-red) $\text{El}_{Nat}(0,\ \text{c},\ \text{e}) \to_1 \text{c}$
- β_{2Nat}-red) $\text{El}_{Nat}(\text{succ(m)},\ \text{c},\ \text{e}) \to_1 \text{e(m,El}_{Nat}\text{(m, c, e))}$
- $\text{red}_I) \ \dfrac{\text{t}_1 \to_1 \text{t}_2}{\text{El}_{N1}(\text{t}_1,\ \text{c},\ \text{e}) \to_1 \text{El}_{N1}(\text{t}_2,\ \text{c},\ \text{e})} \qquad \text{red}_{II}) \ \dfrac{\text{c}_1 \to_1 \text{c}_2}{\text{El}_{N1}(\text{t},\ \text{c}_1,\ \text{e}) \to_1 \text{El}_{N1}(\text{t},\ \text{c}_2,\ \text{e})}$
- Novità dei numeri naturali rispetto al tipo singoletto N-red) $\dfrac{\text{t}_1 \to_1 \text{t}_2}{\text{succ}(\text{t}_1) \to_1 \text{succ}(\text{t}_2)}$
- + riduzione $\to_1$ rispetto a N_1

3.1.9 Addizione per ricorsione

Di seguito riporto un esercizio, svolto in aula, con lo scopo di comprendere come l'uguaglianza definizionale, tra numeri naturali, non coincide con l'uguaglianza aritmetica in matematica.

La somma, tra numeri naturali, viene definita usando l'eliminatore dipendente *E-Nat*$_{dip}$(§??). In questo modo si riesce a definire per la nostra teoria T_{1Nat}

$$\begin{gathered} \text{w} + \text{z} \in \text{Nat}\ [\text{w} \in \text{Nat}, \text{z} \in \text{Nat}] \\ \text{come} \\ \text{El}_{Nat}(\text{z}, \text{w}, (\text{x,y}).\text{succ}(\text{y}))[\text{w} \in \text{Nat}, \text{z} \in \text{Nat}] \end{gathered}$$

Usando la nozione di primitava ricorsiva e decidendo di ricorrere su z

$$\begin{gathered} \text{w} + 0 \equiv \text{w} \\ \text{w} + \text{succ}(\text{z}) \equiv \text{succ}(\text{w}+\text{z}) \equiv \text{succ}(\text{y}) \end{gathered}$$

Ecco che l'albero di derivazione assume la seguente forma:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\text{Nat type}\,[\,]}\,[\,]\ \text{cont}}{\text{w} \in \text{Nat cont}}\,\text{F-c}}{\text{Nat type}[\text{w} \in \text{Nat}]}\,\text{F-Nat}}{\text{w} \in \text{Nat}, \text{z} \in \text{Nat cont}}\,\text{F-c}\ (\text{z} \in \text{Nat}) \notin \text{w} \in \text{Nat}}{\text{Nat type}[\text{w} \in \text{Nat}, \text{z} \in \text{Nat}]}\,\text{F-Nat} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\text{Nat type}\,[\,]}\,[\,]\ \text{cont}}{\text{w} \in \text{Nat cont}}\,\text{F-c}\ (\text{w} \in \text{Nat}) \notin [\,]}{\text{w} \in \text{Nat}[\text{w} \in \text{Nat}]}\,\text{var} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\text{Nat type}\,[\,]}\,[\,]\ \text{cont}}{\text{w} \in \text{Nat cont}}\,\text{F-c}\ (\text{w} \in \text{Nat}) \notin [\,]}{\text{Nat type}[\text{w} \in \text{Nat}]}\,\text{F-Nat}}{\text{w} \in \text{Nat}, \text{z} \in \text{Nat cont}}\,\text{F-c}\ (\text{z} \in \text{Nat}) \notin \text{w} \in \text{Nat}}{\text{succ}(\text{y}) \in [\text{w} \in \text{Nat}, \text{z} \in \text{Nat}, \text{y} \in \text{Nat}]}\,\text{I}_2\text{-Nat}_{prog}}{\text{El}_{Nat}(\text{z}, \text{w}, (\text{x,y}).\text{succ}(\text{y}))[\text{w} \in \text{Nat}, \text{z} \in \text{Nat}]}\,\text{E-Nat}_{dip}$$

- Prova induttiva (minimo dei controlli da fare per verificare l'esattezza della derivazione)
  - Caso base: cosa accade se poniamo al posto di z lo 0?
    $\text{El}_{Nat}(0, \text{w}, (\text{x,y}).\text{succ}(\text{y})) \to_1 \text{w}$ per β_{1Nat}-*red* ($\equiv \text{w} + 0 = \text{w}$)
  - Passo induttivo: ricorro su z (esempio succ(0) = 1)
    $\text{El}_{Nat}(\text{succ}(0), \text{w}, (\text{x,y}).\text{succ}(\text{y}))$ per β_{1Nat}-*red* $\to_1 \text{succ}(\text{El}_{Nat}(0, \text{w}, (\text{x,y}).\text{succ}(\text{y}))) \to_1 \text{succ}(\text{w})$
    Dunque $\text{w} + 1 = \text{succ}(\text{w}) \in \text{Nat}$

  $\Rightarrow$ Il programma fa effettivamente quello che dovrebbe.

Osservazioni sull'addizione

$\text{w} +_1 \text{z} \equiv \text{El}_{Nat}(\text{z}, \text{w}, (\text{x,y}), \text{succ}(\text{y})) \neq$ come NF da z.
Se sostituisco w con 0, allora $0 +_1 \text{z} \equiv \text{w} +_1 \text{z}[\frac{w}{0}] \equiv \text{El}_{Nat}(\text{z}, 0, (\text{x,y}), \text{succ}(\text{y}))$ è in NF (dunque non riesco più a ridurla ulteriormente).
Ecco che $0 +_1 \text{z}$ è un valore in NF $\neq$ da z, da cui è impossibile dimostrare che $0 +_1 \text{z} = \text{z} \in \text{Nat}[\text{z} \in \text{Nat}]$.
Questo non accade per $\text{w} +_1 0 = \text{w}$ (§3.1.9).
Perciò, se noi scriviamo la somma riccorrendo sul secondo membro, riusciamo a dire che *primo-membro* $+_1$ *0* = *primo-membro*, ma non che *0* $+_1$ *secondo-membro* = *secondo - membro*. In quanto non esiste alcuna sottostrategia deterministica, per il secondo caso, per cui il programma si ferma.

In conclusione, l'uguaglianza definizionale di termini con variabili è diversa dall'uguaglianza aritmentica. Eccezione fatta per le espressioni chiuse, senza variabili, perchè il termine chiuso si riduce a un'unica NF, che si dimostra essere un numero arabico.

3.2 Tipo delle liste di un tipo

Il tipo delle liste di elementi costruisce un costruttore delle liste ed è definito dalle regole seguenti.

3.2.1 Regole di Formazione

$$\text{F-cost)}\ \frac{\text{A type } [\Gamma]}{\text{List(A)type } [\Gamma]}$$

3.2.2 Regole di Introduzione

$$\text{I}_1\text{-list)}\ \frac{\text{list(A) type } [\Gamma]}{\text{nil} \in \text{List(A)}[\Gamma]} \qquad \text{I}_2\text{-list)}\ \frac{\text{s} \in \text{List(A)}[\Gamma] \qquad \text{a} \in \text{A}[\Gamma]}{\text{cons(s,a)} \in \text{List(A)}[\Gamma]}$$

3.2.3 Regole di Eliminazione

$$\text{E-List)}\ \frac{\begin{array}{c}\text{M(z) type}[\Gamma,\ \text{z} \in \text{List(A)}] \\ \text{t} \in \text{List(A)}[\Gamma] \qquad \text{e(x,w,y)} \in \text{M(cons(x,w))}[\Gamma,\ \text{x} \in \text{List(A)},\ \text{w} \in \text{A},\ \text{y} \in \text{M(x)}] \\ \text{c} \in \text{M(nil)}[\Gamma]\end{array}}{\text{El}_{list}\text{(t,c,e)} \in \text{M(t)}[\Gamma]}$$

3.2.4 Regole di Conservazione

$$\text{C}_1\text{-list)}\ \frac{\begin{array}{c}\text{M(z) type}[\Gamma,\ \text{z} \in \text{List(A)}] \\ \text{c} \in \text{M(nil)}[\Gamma]\end{array} \qquad \text{e(x,w,y)} \in \text{M(cons(x,w))}[\Gamma,\ \text{x} \in \text{List(A)},\ \text{w} \in \text{A},\ \text{y} \in \text{M(x)}]}{\text{El}_{list}\text{(nil,c,e)} = \text{c} \in \text{M(nil)}[\Gamma]}$$

$$\text{C}_2\text{-list)}\ \frac{\text{M(z) type}[\Gamma,\ \text{z} \in \text{List(A)}] \qquad \begin{array}{c}\text{s} \in \text{List(A)}[\Gamma] \\ \text{a} \in \text{A}[\Gamma] \\ \text{c} \in \text{M(nil)}[\Gamma]\end{array} \qquad \text{e(x,w,y)} \in \text{M(cons(x,w))}[\Gamma,\ \text{x} \in \text{List(A)},\ \text{w} \in \text{A},\ \text{y} \in \text{M(x)}]}{\text{El}_{list}\text{(cons(s,a),c,e)} = \text{e(s,a, E}_{list}\text{(s,c,e)} \in \text{M(cons(s,a))}[\Gamma]}$$

3.2.5 Regole di Uguaglianza

$$\text{eq-I}_1\text{-List)}\ \frac{\text{A}_1 = \text{A}_2 \in \text{type}[\Gamma]}{\text{List(A}_1\text{)} = \text{List(A}_2\text{) type}(\Gamma)}$$

$$\text{eq-I}_2\text{-List)}\ \frac{\text{s}_1 = \text{s}_2 \in \text{List(A)}[\Gamma] \qquad \text{a}_1 = \text{a}_2 \in \text{A}[\Gamma]}{\text{cons(s}_1\text{,a}_1\text{)} = \text{cons(s}_2\text{,a}_2\text{)} \in \text{List(A)}(\Gamma)}$$

$$\text{E-eq-List)}\ \frac{\begin{array}{c}\text{M(z) type}[\Gamma,\ \text{z} \in \text{List(A)}] \\ \text{t}_1 = \text{t}_2 \in \text{list(A)}[\Gamma] \qquad \text{e}_1\text{(x,w,y)} = \text{e}_2\text{(x,w,y)} \in \text{M(cons(x,w))}[\Gamma,\ \text{x} \in \text{List(A)},\ \text{w} \in \text{A},\ \text{y} \in \text{M(x)}] \\ \text{c}_1 = \text{c}_2 \in \text{M(nil)}[\Gamma]\end{array}}{\text{El}_{list}\text{(t}_1\text{, c}_1\text{, e}_1\text{)} = \text{El}_{list}\text{(t}_2\text{, c}_2\text{, e}_2 \in \text{M(t}_1\text{)}[\Gamma]}$$

3.2.6 Eliminatore dipendente

L'eliminatore ha anche la forma dipendente, conveniente da usare soprattutto per motivi pratici.

$$\text{E-List}_{dip})\ \frac{\text{M(z) type}[\Gamma,\ z \in \text{List(A)}] \qquad \text{c} \in \text{M(nil)}[\Gamma] \qquad \text{e(x,w,y)} \in \text{M(cons(x,w))}[\Gamma,\ \text{x} \in \text{List(A)},\ \text{w} \in \text{A},\ \text{y} \in \text{M(x)}]}{\text{El}_{list}\text{(z,c,e)} \in \text{M(z)}[\Gamma,\ \text{z} \in \text{List(A)}]}$$

Quando si scrive un programma é bene scrivere la sostituzione espressamente, come riporto sotto.

$$\text{E-List}_{dip})\ \frac{\text{D type}[\Gamma,\ \text{z} \in \text{List(A)}] \qquad \text{c} \in \text{D}(\frac{z}{nil})[\Gamma] \qquad \text{e(x,w,y)} \in \text{M}(\frac{z}{cons(x,w)})[\Gamma,\ \text{x} \in \text{List(A)},\ \text{w} \in \text{A},\ \text{y} \in \text{D}(\frac{z}{x})]}{\text{El}_{list}\text{(z,c,e)} \in \text{M(z)}[\Gamma,\ \text{z} \in \text{List(A)}]}$$

3.2.7 Semantica operazionale del tipo lista

La relazione $\to_1$ viene definita all'interno dei termini con l'uso delle seguenti regole di riduzione:

- β_{1list}-red) $\text{El}_{list}(\text{nil, c, e}) \to_1 \text{c}$
- β_{2list}-red) $\text{El}_{list}(\text{cons(s,a), c, e}) \to_1 \text{e(s,a,El}_{list}\text{(s, c, e))}$
- $\text{red}_I)\ \dfrac{\text{t}_1 \to_1 \text{t}_2}{\text{El}_{list}(\text{t}_1,\ \text{c, e}) \to_1 \text{El}_{list}(\text{t}_2,\ \text{c, e})}$ $\qquad \text{red}_{II})\ \dfrac{\text{c}_1 \to_1 \text{c}_2}{\text{El}_{list}(\text{t},\ \text{c}_1,\ \text{e}) \to_1 \text{El}_{list}(\text{t},\ \text{c}_2,\ \text{e})}$
- Novità delle liste rispetto al tipo singoletto $\quad \text{L-red}_I)\ \dfrac{\text{s}_1 \to_1 \text{s}_2}{\text{cons}(\text{s}_1\text{,a}) \to_1 \text{cons}(\text{s}_2\text{,a})}$
- $\text{L-red}_{II})\ \dfrac{\text{a}_1 \to_1 \text{a}_2}{\text{cons(s,a}_1) \to_1 \text{cons(s,a}_2)}$
- $+$ riduzione $\to_1$ rispetto a N_1

3.3 Esercizi

3.3.1 Tipo dei numeri Naturali

1 Dimostrare che le regole enunciate in §3.1, I_2-Nat_{prog} ed E-Nat_{prog}, sono ammissibili nel sistema di teoria dei tipi dei numeri naturali.

Soluzione

$$\text{ind-te}\ \frac{\text{I}_2\text{-Nat}\ \dfrac{\text{s-checks}\ \dfrac{\text{F-Nat}\ \dfrac{\overline{\Gamma\ count}}{\text{Nat type}[\Gamma]}}{\text{x} \in \text{Nat}[\Gamma]}}{\text{succ(x)} \in \text{Nat}[\Gamma]} \qquad \text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{\overline{\Gamma\ count}}{\text{Nat type}[\Gamma]}}{\Gamma,\ \text{x} \in \text{Nat cont}}\ (\text{x} \in \text{Nat}) \notin \Gamma}{\text{succ(x)} \in \text{Nat}[\Gamma,\ \text{x} \in \text{Nat}]}$$

Assumo che le premesse di I_2-Nat_{prog} (Γ *count*) sia valida, perciò è valido, dalla prova sopra, anche il giudizio di conclusione *succ(x)* $\in$ *Nat*$[\Gamma,\ x \in Nat]$, di conseguenza derivabile in $\mathcal{T}$.

$$\text{ind-ty}\ \frac{\text{sub-ter}\ \dfrac{\text{s-checks}\ \dfrac{\text{s-checks}\ \dfrac{\Gamma\ \text{cont}}{\text{Nat type}[\Gamma]}}{\text{w} \in \text{Nat type}[\Gamma]} \qquad \text{E-S}\ \dfrac{\overline{\begin{matrix}\text{e(x,y)} \in \text{M(succ(x))}[\\ \Gamma,\ \text{x} \in \text{Nat},\\ \text{y} \in \text{Nat},\ \text{y} \in \text{M(x)}]\end{matrix}} \qquad \dfrac{}{\text{c} \in \text{M(0)}[\Gamma]} \qquad \dfrac{}{\begin{matrix}\text{M(z) type Nat}\ [\\ \Gamma,\ \text{z} \in \text{Nat}]\end{matrix}} \qquad \text{s-checks}\ \dfrac{\Gamma\ \text{cont}}{\text{t} \in \text{Nat}[\Gamma]}}{\text{El}_{Nat}\text{(t,c,e)} \in \text{M(t)}[\Gamma]}}{\text{El}_{Nat}\text{(w,c,e)} \in \text{M(t)}[\Gamma]} \qquad \text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{\Gamma\ \text{cont}}{\text{Nat type}[\Gamma]}}{\Gamma,\ \text{w} \in \text{Nat cont}}\ (\text{w} \in \text{N}$$

Assumo che le premesse di I_E-Nat_{prog} (*M(z) type[Γ, z ∈ Nat], c ∈ M(0)[Γ], e(x,y) ∈ M(succ(x))[Γ, x ∈ Nat, y ∈ M(x)]*) sia valida, pèrciò è valido, dalla prova sopra, anche il giudizio di conclusione *El_{Nat}(w,c,e) ∈ M(t)[Γ, w ∈ Nat]*, di conseguenza derivabile in *T*.

2 Definire w + 2 ∈ Nat[w ∈ Nat], ove 2 è l'abbreviazione del termine ottenuto applicando 2 ≡ succ(succ(0)).

Soluzione

La ricorsione la faccio w, usando lo schema di ricorsione primitva, vale che w + 2 ≡ El_{Nat}(w, 2, (x,y).succ(y)) ∈ Nat[w ∈ Nat]

$$\text{E-Nat}_{dip}\ \dfrac{\text{F-Nat}\ \dfrac{\text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ \text{cont}}{\text{Nat type}[\,]}}{\text{w} \in \text{Nat cont}}\ (\text{w} \in \text{Nat}) \notin [\,]}{\text{Nat type}[\text{w} \in \text{Nat}]} \qquad \begin{array}{c}\text{I1-Nat}\ \dfrac{[\,]\ \text{cont}}{0 \in \text{Nat}[\,]} \\ \text{I2-Nat}\ \dfrac{1 \in \text{Nat}[\,]}{2 \in \text{Nat}[\,]}\end{array} \qquad \text{I2-Nat}_{prog}\ \dfrac{\text{F-Nat}\ \dfrac{\text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ \text{cont}}{\text{Nat type}[\,]}}{\text{x} \in \text{Nat cont}}\ (\text{x} \in \text{Nat}) \notin [\,]}{\text{succ(y)} \in \text{Nat}[\text{x} \in \text{Nat}, \text{y} \in \text{Nat}]}}{El_{Nat}(\text{w, 2, (x,y).succ(y))} \in \text{Nat}[\text{w} \in \text{Nat}]}$$

Dimostrazione di correttezza di El_{Nat}(w, 2, (x,y).succ(y)) ∈ Nat[w ∈ Nat]

- El_{Nat}(0, 2, (x,y).succ(y)) $\rightarrow_1$ 2 per β_{1Nat}-*red*
- El_{Nat}(succ(m), 2, (x,y).succ(y)) $\rightarrow_1$ succ(El_{Nat}(m, 2, (x,y).succ(y))) per β_{2Nat}-*red* ⇒ per m = 0 ≡ succ(El_{Nat}(0, 2, (x,y).succ(y))) $\rightarrow_1$ succ(2) ∈ Nat = 3 (dal punto precedente).

3 Definire l'operazione di addizione usando le regole del tipo dei numeri naturali.

$$\text{x} + \text{y} \in \text{Nat}[\text{x} \in \text{Nat}, \text{y} \in \text{Nat}]$$

in modo tale che valga x + 0 = x ∈ Nat[x ∈ Nat]

Soluzione

La ricorsione la faccio y, usando lo schema di ricorsione primitva, vale che x + y ≡ El_{Nat}(y, x, (w,z).succ(z)) ∈ Nat[x ∈ Nat, y ∈ Nat]

$$\text{E-Nat}_{dip}\ \dfrac{\text{F-Nat}\ \dfrac{\text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ \text{cont}}{\text{Nat type}[\,]}}{\text{x} \in \text{Nat cont}}\ (\text{x} \in \text{Nat}) \notin [\,]}{\text{Nat type}[\text{x} \in \text{Nat}]}\ \text{F-c}\ \dfrac{}{\text{x} \in \text{Nat, y} \in \text{Nat cont}}\ (\text{y} \in \text{Nat}) \notin \text{x} \in \text{Nat}}{\text{Nat type}[\text{x} \in \text{Nat}, \text{y} \in \text{Nat}]} \qquad \text{F-Nat}\ \dfrac{\text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ \text{cont}}{\text{Nat type}[\,]}}{\text{x} \in \text{Nat cont}}\ (\text{x} \in \text{Nat}) \notin [\,]}{\text{var}\ \dfrac{}{\text{x} \in \text{Nat}[\text{x} \in \text{Nat}]}} \qquad \text{I}_2\text{-Nat}_{prog}\ \dfrac{\text{F-Nat}\ \dfrac{\text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ \text{cont}}{\text{Nat type}[\,]}}{\text{x} \in \text{Nat cont}}\ (\text{x} \in \text{Nat}}{\text{Nat type}[\text{x} \in \text{Nat}]}\ \text{F-c}\ \dfrac{}{\text{x} \in \text{Nat, w} \in \text{Nat cont}}\ (\text{w} \notin}{\text{succ(z)} \in \text{Nat}[\text{x} \in \text{Nat, w} \in \text{Nat, z} \in \text{N}}}{El_{Nat}(\text{y, x, (w,z).succ(z))} \in \text{Nat}[\text{x} \in \text{Nat, y} \in \text{Nat}]}$$

Dimostrazione di correttezza di El_{Nat}(y, x, (w,z).succ(z)) ∈ Nat[x ∈ Nat, y ∈ Nat]

- El_{Nat}(0, x, (w,z).succ(z)) $\rightarrow_1$ x per β_{1Nat}-*red*
- El_{Nat}(succ(y), x, (w,z).succ(z)) $\rightarrow_1$ succ(El_{Nat}(y, x, (w,z).succ(z))) per β_{2Nat}-*red* ⇒ per y=0 ≡ succ(El_{Nat}(0, x, (w,z).succ(z))) $\rightarrow_1$ succ(x) ∈ Nat = x + 1 (dal punto precedente).

4 Definire l'operazione di addizione usando le regole del tipo dei numeri naturali.

$$x + y \in Nat[x \in Nat, y \in Nat$$

in modo tale che valga $0 + y = x \in Nat[x \in Nat]$

Soluzione

La ricorsione la faccio x, percui, usando lo schema di ricorsione primitva, vale che $x + y \equiv El_{Nat}(x, y, (w,z).succ(z)) \in Nat[x \in Nat, y \in Nat]$

$$\text{ex-te}\ \frac{El_{Nat}(x, y, (w,z).succ(z)) \in Nat[y \in Nat, x \in Nat] \qquad \text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ cont}{Nat\ type\ [\,]}}{x \in Nat\ cont}\ \dfrac{\text{F-Nat}\ \dfrac{x \in Nat\ cont}{Nat\ type[x \in Nat]}}{x \in Nat,\ y \in Nat\ cont}\ (y \in Nat) \notin x \in Nat}{El_{Nat}(x, y, (w,z).succ(z)) \in Nat[x \in Nat, y \in Nat]}\ (x \in Nat) \notin [\,]$$

$$\text{E-Nat}_{dip}\ \frac{\text{F-Nat}\ \dfrac{\text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ cont}{Nat\ type[\,]}}{y \in Nat\ cont}\ (y \in Nat) \notin [\,]}{Nat\ type[y \in Nat]}}{y \in Nat,\ x \in Nat\ cont}\ (x \in Nat) \notin y \in Nat \qquad \text{var}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ cont}{Nat\ type\ [\,]}}{y \in Nat[y \in Nat]}\ (y \in Nat) \notin [\,] \qquad \text{I}_2\text{-Nat}_{prog}\ \dfrac{\text{F-Nat}\ \dfrac{\text{F-c}\ \dfrac{\text{F-Nat}\ \dfrac{[\,]\ cont}{Nat\ type\ [\,]}}{y \in Nat\ cont}\ (y \in Nat) \notin [\,]}{Nat\ type[y \in Nat]}}{y \in Nat,\ w \in Nat\ cont}\ (w \in Nat) \notin y \in Nat}{succ(z) \in [y \in Nat,\ w \in Nat,\ z \in Nat]}}{El_{Nat}(x, y, (w,z).succ(z)) \in Nat[y \in Nat, x \in Nat]}$$

Dimostrazione di correttezza di $El_{Nat}(x, y, (w,z).succ(z)) \in Nat[y \in Nat, x \in Nat]$

- $El_{Nat}(0, y, (w,z).succ(z)) \rightarrow_1 y$ per β_{1Nat}*-red*
- $El_{Nat}(succ(x), y, (w,z).succ(z)) \rightarrow_1 succ(El_{Nat}(x, y, (w,z).succ(z)))$ per β_{2Nat}*-red* $\Rightarrow$ per $x{=}0 \equiv succ(El_{Nat}(0, y, (w,z).succ(z))) \rightarrow_1 succ(y) \in Nat = y + 1$ (dal punto precedente).

3.3.2 Tipo delle liste

1 Dati i tipi singoletto e delle liste è possibile definire un tipo dei numeri naturali *Nat*?

Soluzione

Posso vedere una lista come una collezione di n singoletti. Ecco che posso definire il tipo dei numeri naturali su questa lista, per ricorsione primitiva, nel modo sottostante:

$$0 = nil$$
$$1 = cons(nil, *)$$
$$n = cons(cons(n\text{-}1), *)$$

2 Dato un tipo A semplice, ovvero non dipendente, per esempio A = N_1, se si vuole definire un'operazione

$$\mathbf{append_1(x`,y`) \in List(A)[x` \in List(A),\ y` \in A]}$$

tale che l'elemento y` venga posto alla fine della lista x`, allora basta definire $append_1$ nel modo seguente

$$\mathbf{append_1(x`,y`) = cons(x`,y`) \in List(A)[x` \in List(A),\ y` \in A]}$$

Ma se si vuole definire un'operazione

$$\mathbf{append_2(x`,y`) \in List(A)[x` \in List(A),\ y` \in A]}$$

tale che y` venga posto all'inizio della lista x` ,allora come occorre procedere?.

Soluzione (*Procedimento preso dalle note*)

Per riuscire a porre y` all'inizio della lista x`, devo usare la regola di eliminazione sulla lista x`. Per farlo, devo prima di tutto definire equazionalmente la definizione ricorsiva di $append_2$

$$\begin{aligned} append_2(nil,\ y`) &= cons(nil,\ y`) = y` \\ append_2(cons(s,\ x`),\ y`) &= cons(append_2(s,\ y`),\ x`) \end{aligned}$$

Ora posso applicare la regola dell'eliminazione nel modo seguente:

- Premesse:
  - M(z) type[Γ, z $\in$ List(A)] $\equiv$ List(A) type[x` $\in$ List(A), y` $\in$ List(A)]
  - t $\in$ List(A)[Γ] $\equiv$ x` $\in$ List(A)[x` $\in$ List(A), y` $\in$ List(A)]
  - c $\in$ M(nil)[Γ] $\equiv$ y` $\in$ List(A)[x` $\in$ List(A), y` $\in$ List(A)]
  - e(x,w,y) $\in$ M(cons(x,w))[Γ, x $\in$ List(A), w $\in$ A, y $\in$ M(x)] $\equiv$ cons(z,y) $\in$ List(A)[x` $\in$ List(A), y` $\in$ List(A), z $\in$ List(A), y$\in$ A]
- Giudizio di conclusione:
  - El_{list}(t,c,e) $\in$ M(t)[Γ] $\equiv$ El_{list}(x`, y`,(x,y,z).cons(z,y)) $\in$ List(A)[x` $\in$ List(A), y` $\in$ List(A)]

3 Definire l'operazione *append* di accostamento di una lista a un'altra di tipo A type[ ] usando le regole del tipo delle liste

$$\mathbf{append(x,y) \in List(A)[x \in List(A),\ x \in List(A)]}$$

in modo tale che valga append(x,nil) = x $\in$ List(A)[x $\in$ List(A)]

Soluzione

Per riuscire a poter concatenare la lista x alla lista y, devo usare la regola di eliminazione sulla lista x. Per farlo, devo prima di tutto definire equazionalmente la definizione ricorsiva di append

$$\text{append(x, nil)} = \text{cons(x, nil)} = \text{x}$$
$$\text{append(x, cons(y, a))} = \text{cons(x, append(y, a))}$$

Ora posso applicare la regola dell'eliminazione nel modo seguente:

- Premesse:
  - M(z) type[Γ, z ∈ List(A)] ≡ List(A) type[x ∈ List(A), y ∈ List(A)]
  - t ∈ List(A)[Γ] ≡ y ∈ List(A)[x ∈ List(A), y ∈ List(A)]
  - c ∈ M(nil)[Γ] ≡ x ∈ List(A)[x ∈ List(A), y ∈ List(A)]
  - e(x,w,y) ∈ M(cons(x,w))[Γ, x ∈ List(A), w ∈ A, y ∈ M(x)] ≡ cons(z,w) ∈ List(A)[x ∈ List(A) y ∈ List(A), z ∈ List(A), w∈ A]
- Giudizio di conclusione:
  - El_{list}(t,c,e) ∈ M(t)[Γ] ≡ El_{list}(y, x,(u,v,z).cons(z,v)) ∈ List(A)[x ∈ List(A), y ∈ List(A)]

$$\text{E-list}\ \frac{\mathbf{1} \qquad \mathbf{2} \qquad \mathbf{3} \qquad \mathbf{4}}{\text{El}_{list}(\text{y, x,(u,v,z).cons(z,v)}) \in \text{List(A)[x} \in \text{List(A), y} \in \text{List(A)]}}$$

1

$$\text{s-checks}\ \frac{[\,]\ \text{cont}}{\text{A type}[\,]} \qquad \text{ind-ty}\ \frac{\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}[\,]} \qquad \text{F-cost}\ \dfrac{\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}[\,]}}{\text{List(A) type}\ [\,]} \qquad \text{F-c}\ \dfrac{\text{List(A) type}\ [\,]}{\text{x} \in \text{List(A) cont}}\ (\text{x} \in \text{List(A)}) \notin [\,]}{\text{A type[x} \in \text{List(A)]}} \qquad \text{F-cost}\ \frac{\text{List(A) type[x} \in \text{List(A)]}}{\text{F-c}\ \dfrac{\text{x} \in \text{List(A), y} \in \text{List(A) cont}}{\text{A type[x} \in \text{List(A), y} \in \text{List(A)]}}}\ (\text{y} \in \text{List(A)}) \notin \text{x} \in \text{List(A)}$$

2

$$\text{s-checks}\ \frac{[\,]\ \text{cont}}{\text{A type}[\,]} \qquad \text{ind-ty}\ \frac{\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}[\,]} \qquad \text{F-cost}\ \dfrac{\text{List(A) type}\ [\,]}{\text{F-c}\ \dfrac{\text{x} \in \text{List(A) cont}}{\text{A type[x} \in \text{List(A)]}}}\ (\text{x} \in \text{List(A)}) \notin [\,]}{\text{List(A) type[x} \in \text{List(A)]}} \qquad \text{F-c}\ \frac{\text{x} \in \text{List(A), y} \in \text{List(A) cont}}{\text{var}\ \dfrac{\text{y} \in \text{List(A)[x} \in \text{List(A), y} \in \text{List(A)]}}{}}\ (\text{y} \in \text{List(A)}) \notin \text{x} \in \text{List(A)}$$

3

$$\text{ind-ty}\ \dfrac{\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}[\,]} \qquad \text{F-cost}\ \dfrac{\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}[\,]}}{\text{List(A) type}\ [\,]} \quad \text{F-c}\ \dfrac{}{\text{y} \in \text{List(A) cont}}\ (\text{y} \in \text{List(A)}) \notin [\,]}{\text{F-cost}\ \dfrac{}{\text{A type}[\text{y} \in \text{List(A)}]} \quad \text{F-c}\ \dfrac{}{\text{List(A) type}[\text{y} \in \text{List(A)}]} \quad \text{var}\ \dfrac{}{\text{y} \in \text{List(A)}, \text{x} \in \text{List(A) cont}}\ (\text{x} \in \text{List(A)}) \notin \text{y} \in \text{List(A)}}$$

$$\text{var}\ \dfrac{}{\text{x} \in \text{List(A)}[\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}]}$$

4

$$\text{I}_2\text{-list}\ \dfrac{\text{var}\ \dfrac{\mathbf{4`}}{\text{z} \in \text{List(A)}[\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}, \text{z} \in \text{List(A)}, \text{w} \in \text{A}]} \qquad \text{var}\ \dfrac{\mathbf{4`}}{\text{w} \in \text{A}[\text{x} \in \text{List(A)}\ \text{y} \in \text{List(A)}, \text{z} \in \text{List(A)}, \text{w} \in \text{A}]}}{\text{cons(z,w)} \in \text{List(A)}[\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}, \text{z} \in \text{List(A)}, \text{w} \in \text{A}]}$$

$$\text{var}\ \dfrac{}{\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}, \text{z} \in \text{List(A)}, \text{w} \in \text{A cont}}$$

4`

$$\text{ind-ty}\ \dfrac{\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}[\,]} \qquad \text{F-cost}\ \dfrac{\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}\ [\,]}}{\text{List(A) type}\ [\,]} \quad \text{F-c}\ \dfrac{}{\text{x} \in \text{List(A) cont}}\ (\text{x} \in \text{List(A)}) \notin [\,]}{\text{F-cost}\ \dfrac{}{\text{A type}[\text{x} \in \text{List(A)}]} \quad \text{F-c}\ \dfrac{}{\text{List(A) type}[\text{x} \in \text{List(A)}]} \quad \text{s-checks}\ \dfrac{}{\text{x} \in \text{List(A)}, \text{y} \in \text{List(A) cont}}\ (\text{y} \in \text{List(A)}) \notin \text{x} \in \text{List(A)}}$$

$$\text{s-checks}\ \dfrac{[\,]\ \text{cont}}{\text{A type}[\,]} \qquad \text{F-cost}\ \dfrac{}{\text{A type}[\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}]} \quad \text{F-c}\ \dfrac{}{\text{List(A) type}[\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}]} \quad \text{ind-ty}\ \dfrac{}{\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}, \text{z} \in \text{List(A) cont}}\ (\text{z} \in \text{List(A)}) \notin (\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}))$$

$$\text{F-c}\ \dfrac{\text{A type}[\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}, \text{z} \in \text{List(A)}]}{\text{x} \in \text{List(A)}\ \text{y} \in \text{List(A)}, \text{z} \in \text{List(A)}, \text{w} \in \text{A cont}}\ (\text{w} \in \text{A}) \notin (\text{x} \in \text{List(A)}, \text{y} \in \text{List(A)}, \text{z} \in \text{List(A)})$$

Dimostrazione di correttezza di El_{list}(y, x,(u,v,z).cons(z,v)) ∈ List(A)[x ∈ List(A), y ∈ List(A)]

- El_{list}(nil, x,(u,v,z).cons(z,v)) $\to_1$ x per β_{1list}-*red*
- El_{list}(cons(y,a), x,(u,v,z).cons(z,v)) $\to_1$ cons(El_{list}(y, x, (u,v,z).cons(z,v))) per β_{2list}-*red* $\Rightarrow$ per y = nil $\equiv$ cons(El_{list}(nil, x, (u,v,z).cons(z,v))) $\to_1$ cons(nil, x) ∈ List(A) = x (dal punto precedente).

3.4 Tipo della somma disgiunta

Il tipo somma disgiunta è un costruttore di tipo. Questi non è dipendente se da solo, lo diventa solo quando agisce su un tipo dipendente.

Anche con il tipo somma disgiunta si parla di tipo induttivo (accade già per N_1, Nat, List(A)).
Le regole del tipo della somma disgiunta sono le seguenti.

3.4.1 Regole di Formazione

$$\text{F-+)}\ \frac{\text{B type } [\Gamma] \qquad \text{C type } [\Gamma]}{\text{B + C type } [\Gamma]}$$

3.4.2 Regole di Introduzione

$$\text{I}_1\text{-+)}\ \frac{\text{b} \in \text{B}[\Gamma] \qquad \text{B + C type}[\Gamma]}{\text{inl(b)} \in \text{B + C}[\Gamma]} \qquad \text{I}_2\text{-+)}\ \frac{\text{c} \in \text{C}[\Gamma] \qquad \text{B + C type}[\Gamma]}{\text{inr(c)} \in \text{B + C}[\Gamma]}$$

3.4.3 Regole di Eliminaione

$$\text{E-+)}\ \frac{\begin{array}{c}\text{M(z) type}[\Gamma,\ \text{z} \in \text{B + C}] \\ \text{t} \in \text{B + C}[\Gamma]\end{array} \qquad e_B(x_1) \in \text{M(inl}(x_1))[\Gamma,\ x_1 \in \text{B}] \qquad e_C(x_2) \in \text{M(inr}(x_2))[\Gamma,\ x_2 \in \text{C}]}{\text{El}_+(\text{t},e_B,e_C) \in \text{M(t)}[\Gamma]}$$

3.4.4 Regole di Conservazione

$$\text{C}_1\text{-+)}\ \frac{\begin{array}{c}\text{M(z) type}[\Gamma,\ \text{z} \in \text{B + C}] \\ \text{b} \in \text{B}[\Gamma]\end{array} \qquad e_B(x_1) \in \text{M(inl}(x_1))[\Gamma,\ x_1 \in \text{B}] \qquad e_C(x_2) \in \text{M(inr}(x_2))[\Gamma,\ x_2 \in \text{C}]}{\text{El}_+(\text{inl(b)},e_B,e_C) \in \text{M(t)}[\Gamma] = e_B(\text{b}) \in \text{M(inl(b))}[\Gamma]}$$

$$\text{C}_1\text{-+)}\ \frac{\begin{array}{c}\text{M(z) type}[\Gamma,\ \text{z} \in \text{B + C}] \\ \text{c} \in \text{C}[\Gamma]\end{array} \qquad e_B(x_1) \in \text{M(inl}(x_1))[\Gamma,\ x_1 \in \text{B}] \qquad e_C(x_2) \in \text{M(inr}(x_2))[\Gamma,\ x_2 \in \text{C}]}{\text{El}_+(\text{inr(c)},e_B,e_C) \in \text{M(t)}[\Gamma] = e_C(\text{c}) \in \text{M(inr(c))}[\Gamma]}$$

3.4.5 Regole di Uguaglianza

$$\text{eq-F-+)}\ \frac{\text{B}_1 = \text{B}_2 \in \text{type}[\Gamma] \qquad \text{C}_1 = \text{C}_2 \in \text{type}[\Gamma]}{\text{B}_1 + \text{C}_1 = \text{B}_2 + \text{C}_2 \text{ type}(\Gamma)}$$

3.4.6 Eliminatore dipendente

L'eliminatore ha, anche nel caso della somma disgiunta, la forma dipendente.

$$\text{E}_{dip}\text{-+)}\ \frac{\text{M(z) type}[\Gamma,\ \text{z} \in \text{B + C}] \qquad e_B(x_1) \in \text{M(inl}(x_1))[\Gamma,\ x_1 \in \text{B}] \qquad e_C(x_2) \in \text{M(inr}(x_2))[\Gamma,\ x_2 \in \text{C}]}{\text{El}_+(\text{z},e_B,e_C) \in \text{M(z)}[\text{z} \in \text{B + C}]}$$