



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 - 30/07/2020
Eleonora Signor, 1237581

Indice

1	Introduzione	9
1.1	La triplice faccia della teoria dei tipi	9
1.2	Come nasce la teoria dei tipi	9
1.3	Il Paradosso di Russell	10
1.4	Idee principali nelle teorie di tipo moderne	11
1.4.1	Richiamo della teoria del λ -calcolo di <i>Church</i>	11
1.5	Che cosa è un tipo?	12
1.6	Esempi di tipi	14
1.6.1	I tipi dipendenti	14
1.7	Regole paradigmatiche per caratterizzare la teoria dei tipi	15
1.7.1	Simbolo \in	15
1.7.2	Uguaglianza definizionale vs uguaglianza proposizionale	16
1.7.3	Generazione di contesti	16
1.8	Esercizi	16
2	Regole della teoria dei tipi	21
2.1	Regole strutturali	21
2.1.1	Regole di formazione dei contesti	21
2.1.2	Regole di assunzione delle variabili	22
2.1.3	Regole strutturali addizionali sull'uguaglianza	22
2.1.4	Regole di conversione dell'uguaglianza per tipi uguali	22
2.1.5	Regole di indebolimento e di sostituzione	22
2.1.6	Regole proprie e derivabili	23
2.1.7	Nozione di contesto telescopico	24
2.1.8	Esempi di applicazione	24
2.1.9	Regole di tipaggio	25
2.2	Il tipo singoletto	25
2.2.1	Regola di Formazione	25
2.2.2	Regole di Introduzione	26
2.2.3	Regole di Eliminazione	26
2.2.4	Regole di Conversione	26
2.2.5	Eliminatore dipendente	26
2.2.6	Osservazioni sul tipo singoletto	27
2.3	Sanitary checks rules	27
2.4	Schema generale	28
2.5	Uguaglianza definizionale	29
2.5.1	Applicazione dell'uguaglianza definizionale tra termini	29
2.6	Semantica operativa del singoletto	31

2.7	Esercizi	31
3	Tipo dei numeri Naturali	37
3.1	Regole di Formazione	37
3.2	Regole di Introduzione	37
3.3	Regole di Eliminazione	37
3.4	Regole di Conversione	37
3.5	Regole di Uguaglianza	37
3.6	Introduzione ed Eliminatore dipendente	38
3.7	Semantica operativa dei numeri Naturali	38
3.8	Primitiva ricorsiva	38
3.9	Addizione per ricorsione	39
3.9.1	Osservazioni sull'addizione	39
3.10	Esercizi	40
4	Tipo delle liste di un tipo	45
4.1	Regole di Formazione	45
4.2	Regole di Introduzione	45
4.3	Regole di Eliminazione	45
4.4	Regole di Conservazione	45
4.5	Regole di Uguaglianza	46
4.6	Eliminatore dipendente	46
4.7	Semantica operativa del tipo lista	46
4.8	Esercizi	46
5	Tipo della somma disgiunta	53
5.1	Regole di Formazione	53
5.2	Regole di Introduzione	53
5.3	Regole di Eliminazione	53
5.4	Regole di Conservazione	53
5.5	Regole di Uguaglianza	54
5.6	Eliminatore dipendente	54
5.7	Semantica operativa della somma disgiunta	54
5.8	Osservazioni dal punto di vista logico	54
5.8.1	La regola di Formazione	54
5.9	Esercizi	55
6	Tipo dell'uguaglianza proposizionale	57
6.1	Regole di Formazione	57
6.2	Regole di Introduzione	57
6.3	Regole di Eliminazione	57
6.4	Regole di Conservazione	57
6.5	Regole di Uguaglianza	58
6.6	Eliminatore dipendente	58
6.7	Lemma dell'Id	58
6.8	Esercizio di dimostrazione per induzione su addizione con zero	59
6.9	Esercizi	59

7	Tipo somma indicata forte	67
7.1	Regole di Formazione	67
7.2	Regole di Introduzione	67
7.3	Regole di Eliminazione	67
7.4	Regole di Conservazione	68
7.5	Regole di Uguaglianza	68
7.6	Eliminatore dipendente	68
7.7	Semantica operativa della somma indicata forte	68
7.8	Esercizi	69
7.9	Rappresentazione del tipo prodotto cartesiano	71
7.9.1	Regole di Formazione	71
7.9.2	Regole di Introduzione	71
7.9.3	Regole di Proiezione	71
7.9.4	Regole di Uguaglianza delle proiezioni	71
7.9.5	Semantica operativa del prodotto cartesiano	72
7.9.6	Correttezza del prodotto cartesiano	72
7.9.7	Esempi	73
7.10	Usi del tipo somma indicata per rappresentare l'assioma di se- parazione e di quantificazione universale	75
7.10.1	L'assioma di separazione	75
7.10.2	Proposizionale	76
7.10.3	Conclusioni sugli usi del tipo della somma indicata di- sgiunta	77
7.11	Congiunzione come tipo prodotto cartesiano	78
8	Tipo delle funzioni	81
8.1	Regole di Formazione	81
8.2	Regole di Introduzione	81
8.3	Regole di Eliminazione	81
8.4	Regole di Conservazione	81
8.5	Regole di Uguaglianza	82
8.6	Semantica operativa del tipo funzione	82
8.7	Osservazioni dal punto di vista logico	82
8.7.1	La regola di Introduzione	82
8.7.2	La regola di Eliminazione (Modus Ponens)	83
8.8	Tipo prodotto dipendente	83
8.8.1	Regole di Formazione	83
8.9	Regole di Introduzione	83
8.10	Regole di Eliminazione	83
8.11	Regole di Conservazione	83
8.12	Regole di Uguaglianza	84
8.13	Osservazioni dal punto di vista logico	84
8.13.1	La regola di Formazione	84
9	Tipo vuoto	87
9.1	Regole di Formazione	87
9.2	Regole di Introduzione	87
9.3	Regole di Eliminazione	87
9.4	Regole di Conservazione	87
9.5	Regole di Uguaglianza	87

9.6	Eliminatore dipendente	88
9.7	Semantica operativa del tipo funzione	88
9.8	Osservazioni dal punto di vista logico	88
10	La logica della teoria dei tipi di <i>Martin-Löf</i>	89
11	Principi di induzione per tipi induttivi	93
11.1	Proprietà per i tipi induttivi	93
11.2	Principio di induzione numeri Naturali	94
11.3	Principio di induzione su N_0	94
11.4	Principio di induzione su N_1	94
11.5	Principio di induzione su $List(A)$	94
11.6	Principio di induzione su $B+C$	95
11.7	Principio di induzione su $\sum_{x \in B} C(x)$	95
11.8	Principio di induzione su $Id(A,a,b)$	95
11.9	Esercizi	95
A	Tipi principali e regole	97
A.1	Il tipo singoletto	97
A.1.1	Regola di Formazione	97
A.1.2	Regole di Introduzione	97
A.1.3	Regole di Eliminazione	97
A.1.4	Regole di Conversione	97
A.1.5	Eliminatore dipendente	97
A.2	Tipo dei numeri Naturali	98
A.2.1	Regole di Formazione	98
A.2.2	Regole di Introduzione	98
A.2.3	Regole di Eliminazione	98
A.2.4	Regole di Conversione	98
A.2.5	Regole di Uguaglianza	98
A.2.6	Introduzione ed Eliminatore dipendente	98
A.3	Tipo delle liste di un tipo	99
A.3.1	Regole di Introduzione	99
A.3.2	Regole di Eliminazione	99
A.3.3	Regole di Conservazione	99
A.3.4	Regole di Uguaglianza	99
A.3.5	Eliminatore dipendente	99
A.4	Tipo della somma disgiunta	100
A.4.1	Regole di Formazione	100
A.4.2	Regole di Introduzione	100
A.4.3	Regole di Eliminazione	100
A.4.4	Regole di Conservazione	100
A.4.5	Regole di Uguaglianza	100
A.4.6	Eliminatore dipendente	100
A.5	Tipo dell'uguaglianza proposizionale	101
A.5.1	Regole di Formazione	101
A.5.2	Regole di Introduzione	101
A.5.3	Regole di Eliminazione	101
A.5.4	Regole di Conservazione	101

A.5.5	Regole di Uguaglianza	101
A.5.6	Eliminatore dipendente	101
A.6	Tipo somma indicata forte	102
A.6.1	Regole di Formazione	102
A.6.2	Regole di Introduzione	102
A.6.3	Regole di Eliminazione	102
A.6.4	Regole di Conservazione	102
A.6.5	Regole di Uguaglianza	102
A.6.6	Eliminatore dipendente	102
A.7	Tipo prodotto cartesiano	103
A.7.1	Regole di Formazione	103
A.7.2	Regole di Introduzione	103
A.7.3	Regole di Proiezione	103
A.7.4	Regole di Uguaglianza delle proiezioni	103
A.8	Tipo delle funzioni	104
A.8.1	Regole di Formazione	104
A.8.2	Regole di Introduzione	104
A.8.3	Regole di Eliminazione	104
A.8.4	Regole di Conservazione	104
A.8.5	Regole di Uguaglianza	104
A.9	Tipo vuoto	105
A.9.1	Regole di Formazione	105
A.9.2	Regole di Eliminazione	105
A.9.3	Regole di Uguaglianza	105
A.9.4	Eliminatore dipendente	105

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-theory* per dimostrarne la non falsità o non contraddittorietà;
- **Informatica:** per la correttezza dei programmi, da una semantica operativa 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 connotazioni 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 conclusione 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 contraddittorio.

Di seguito espongo un sistema contraddittorio 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 $\text{VAR} \ni \{x, y, z, w, \dots\}$

dove x, y, z sono da intendersi come insiemi e $x \in y =$ "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 $\phi(x)$ vale che $\exists z \forall y (y \in z \Leftrightarrow \phi(y)) [\equiv \exists z z = \{x \mid \phi(x)\}]$

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

Dimostrazione:

$$\phi(x) = x \notin x \quad (\equiv \neg (x \in x))$$

Per l'assioma di comprensione $\exists z \ z = \{x \mid x \notin x\}$ ($\exists z \ \forall y \ (y \in z \Leftrightarrow y \notin y)$)

Ponendo $y=z$ ottengo che $z \in z \Leftrightarrow z \notin z$, risulta una **contraddizione**.

L'assioma di comprensione è contraddittorio 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:

$$\exists z \ \forall y \ (y \in a \rightarrow (y \in z \Leftrightarrow \phi(y))) \equiv z = \{x \in a \mid \phi(x)\}$$

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:

$$t := x \mid b_1(b_2) \mid \lambda x.t$$

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

Regole di computazione di base

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

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. $t \rightarrow b_1 \rightarrow b_2 \dots \rightarrow b_m \rightarrow t'$.

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. Perciò diventa $\Lambda \equiv (\lambda x.x(x))(\lambda x.x(x))$ che seguendo la computazione si riduce a:

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

Dunque esiste una catena di $(t_i)_{i \in \mathbb{N}}$ di termini $t_i \rightarrow t_{i+1}$. 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 $\lambda x.x \in A \rightarrow B(x \in A)$.

Il λ -calcolo 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 sintassi in un **linguaggio logico** a rappresenta una proposizione di A , inteso come insieme di tipo delle sue dimostrazioni. Perciò un *proof-term* affermatore 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 sintassi: **$A(x) \text{ prop}[x \in D]$** .

Dalla logica si sa che i predicati $\phi(x)$ hanno x senza un dominio specifico, perchè la sintassi non determina che cosa è in x . Al seguito di tutto questo i predicati hanno una variabile che deve essere tipata come **$\phi(x) \text{ prop}[x \in D]$** .

Dunque (definizione di predicato)

$$\exists z \quad z = \{x \in a \mid \phi(x)\} \quad \equiv \quad \phi(x) \text{ prop}[x \in a]$$

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
$A(x) \text{ prop}[x \in D]$	$A(x) \text{ set}[x \in D]$	$A(x) \text{ prop}[x \in D]$	$A(x) \text{ datatype}[x \in D]$

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
$B \times C$ (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 cartesiano (come in <i>set theory</i>)
$B + C$ (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: $A \rightarrow B \equiv \{f \mid f: B \rightarrow C\}$	$B \rightarrow C$, implicazione della proposizione B e della proposizione C	insieme delle funzioni dal <i>datatype</i> B al <i>datatype</i> C

Tabella 1.3: Famiglia di tipi.

1.6.1 I tipi dipendenti

$A(x)\text{type}[x \in B]$
tipo prodotto dipendente
$\prod_{x \in B} C(x)$
tipo somma dipendente disgiunta indicata
$\sum_{x \in B} C(x)$

$A(x)\text{set}[x \in B]$	$A(x)\text{prop}[x \in B]$	$A(x)\text{datatype}[x \in B]$
$\{f : B \rightarrow \prod_{x \in B} C(x)\}$	$\forall x \in B \quad C(x)$	tipo prodotto indicato come in <i>set theory</i> (non esiste un <i>data-type</i> specifico)
$\prod_{x \in B} C(x) = \{b, c \mid b \in B \quad c \in C(b)\}$		
$\bigcup_{x \in B} C(x)$	$\exists x \in B \quad C(x)$	non è primitivo deriva sempre dalla <i>set theory</i>
$\prod_{x \in B} C(x) = \{b, c \mid b \in B \quad c \in C(b)\}$		

Tabella 1.4: Tipi dipendenti.

1.7. REGOLE PARADIGMATICHE PER CARATTERIZZARE LA TEORIA DEI TIPI¹⁵

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

Esiste anche il **tipo uguaglianza**:

- intensionale: $\text{Id}(B, c, d)$;
- estensionale: $\text{Eq}(B, c, d)$.

Introdotta 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):

- **$A \text{ type}[\Gamma]$** : A è un tipo, possibilmente indicato da variabili nel contesto Γ , dipendente da Γ stesso. Rappresenta il giudizio di tipo.
- **$A = B \text{ type}[\Gamma]$** : il tipo A dipendente da Γ è uguale al tipo B dipendente da Γ . Rappresenta il giudizio di uguaglianza di tipo.
- **$a \in A [\Gamma]$** : a è un elemento del tipo A , possibilmente indicato, 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.
- **$a = b \in A [\Gamma]$** : 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:

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

- 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 $1 \equiv \{\emptyset\}$, poichè $0 \equiv \emptyset$.
- 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 $a = b \in A[\Gamma]$ 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 $\text{Eq}(A, a, b)$ 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) 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

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

- 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

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

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

$$(\lambda x. (z (\lambda z. ((\textcolor{red}{x} \textcolor{red}{y} \textcolor{red}{z}) \textcolor{red}{x})) \textcolor{red}{z} \textcolor{red}{x})) \textcolor{red}{x} (\lambda x. ((\lambda y. \textcolor{red}{y} \textcolor{red}{y}) (\lambda z. \textcolor{red}{z} \textcolor{red}{z})))$$

- 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 $n \geq 1$ di termini s_1, \dots, s_n tale che $s_1 \equiv t$ e $s_i \rightarrow_1 s_{i+1}$ per $i = 1, \dots, n-1$ 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 $i \in \text{Nat}$ tale che $s_1 \equiv t$ e $s_i \rightarrow_1 s_{i+1}$. Per ogni $i \in \text{Nat}$ (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 termini diversi del λ -calcolo che sono convergenti è:

$$(\lambda x. x(x))(\lambda y. y(y)) \rightarrow_1 (\lambda y. y(y))(\lambda y. y(y)) \rightarrow_1 \dots \rightarrow_1 (\lambda y. y(y))(\lambda y. y(y)) \rightarrow_1 \dots$$

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

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

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. $L(\rightarrow_1^*) \not\subseteq L(\rightarrow_1)$
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 $((\lambda x.x)z)((\lambda y.y)w) \not\rightarrow_1^*$.

In conclusione risulta vero che $L(\rightarrow_1^*) \not\subseteq L(\rightarrow_1)$.

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 \rightarrow_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 \rightarrow_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 implicitamente dimostrato con la strategia \rightarrow_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 $M \rightarrow \dots \xrightarrow{R_I^*} M^I$ e voglio costruire una derivazione $M \rightarrow \dots \xrightarrow{A_I} M^I$

$M_I \xrightarrow{A_{II}}_1 M^I$. Le strategie di valutazione sono deterministiche, portano pertanto allo stesso risultato della sequenza di derivazione, per cui $R_I = A_I + A_{II}$ 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 $L(\rightarrow_1^*) \subset L(\rightarrow_1)$.

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:

$$A \text{ type}[\Gamma] \quad A = B \text{ type}[\Gamma] \quad a \in A [\Gamma] \quad a = b \in A [\Gamma] \\ + \text{ ausiliaria } \Gamma \text{ cont}$$

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 $A = B \text{ type}[\Gamma]$ può venire omissa.

2.1 Regole strutturali

Assioma unico: $[\] \text{ 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 $\Gamma \multimap A$ e unico assioma $\varphi \multimap \varphi$.

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} \quad \text{dove } [\] = \emptyset$$

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

2.1.2 Regole di assunzione delle variabili

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

2.1.3 Regole strutturali aggiuntive 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:

$$\begin{aligned} \text{ref)} \quad & \frac{A \text{ type}[\Gamma]}{A = A \text{ type}[\Gamma]} & \text{sym)} \quad & \frac{A = B \text{ type}[\Gamma]}{B = A \text{ type}[\Gamma]} \\ \text{tra)} \quad & \frac{A = B \text{ type}[\Gamma] \quad B = C \text{ type}[\Gamma]}{A = C \text{ type}[\Gamma]} \end{aligned}$$

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

$$\begin{aligned} \text{ref)} \quad & \frac{a \in A [\Gamma]}{a = a \in A [\Gamma]} & \text{sym)} \quad & \frac{a = b \in A [\Gamma]}{b = a \in A [\Gamma]} \\ \text{tra)} \quad & \frac{a = b \in A [\Gamma] \quad b = c \in A [\Gamma]}{a = c \in A [\Gamma]} \end{aligned}$$

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:

$$\begin{aligned} \text{conv)} \quad & \frac{a \in A [\Gamma] \quad A = B \text{ type}[\Gamma]}{a \in B [\Gamma]} \\ \text{conv - eq)} \quad & \frac{a = b \in A [\Gamma] \quad A = B \text{ type}[\Gamma]}{a = b \in B [\Gamma]} \end{aligned}$$

2.1.5 Regole di indebolimento e di sostituzione

Indebolimento

$$\begin{aligned} \text{ind - ty)} \quad & \frac{A \text{ type}[\Gamma] \quad \Gamma, \Delta \text{ cont}}{A \text{ type}[\Gamma, \Delta]} & \text{ind - ty - eq)} \quad & \frac{A = B \text{ type}[\Gamma] \quad \Gamma, \Delta \text{ cont}}{A = B \text{ type}[\Gamma, \Delta]} \\ \text{ind - te)} \quad & \frac{a \in A [\Gamma] \quad \Gamma, \Delta \text{ cont}}{a \in A [\Gamma, \Delta]} & \text{ind - te)} \quad & \frac{a = b \in A [\Gamma] \quad \Gamma, \Delta \text{ cont}}{a = b \in A [\Gamma, \Delta]} \end{aligned}$$

Sostituzione

$$\begin{aligned} & C(x_1, \dots, x_n) \text{ type}[\Gamma, x_1 \in A_1, \dots, x_n \in A_n(x_1, \dots, x_{n-1})] \\ \text{sub - typ)} \quad & \frac{a_1 \in A_1 [\Gamma] \dots a_n \in A_n(a_1, \dots, a_{n-1}) [\Gamma]}{C(a_1, \dots, a_n) \text{ type}[\Gamma]} \end{aligned} \quad (2.1)$$

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

$$\begin{array}{c}
C(x_1, \dots, x_n) = D(x_1, \dots, x_n) \text{ type}[\Gamma, x_1 \in A_1, \dots, x_n \in A_n(x_1, \dots, x_{n-1})] \\
\text{sub} - Eqttyp) \quad \frac{a_1 \in A_1 [\Gamma] \dots a_n \in A_n(a_1, \dots, a_{n-1}) [\Gamma]}{C(a_1, \dots, a_n) = D(a_1, \dots, a_n) \text{ type}[\Gamma]}
\end{array} \quad (2.3)$$

$$\begin{array}{c}
C(x_1, \dots, x_n) = D(x_1, \dots, x_n) \text{ type}[\Gamma, x_1 \in A_1, \dots, x_n \in A_n(x_1, \dots, x_{n-1})] \\
\text{sub} - eq - Eqttyp) \quad \frac{a_1 = b_1 \in A_1 [\Gamma] \dots a_n = b_n \in A_n(a_1, \dots, a_{n-1}) [\Gamma]}{C(a_1, \dots, a_n) = D(a_1, \dots, a_n) \text{ type}[\Gamma]}
\end{array} \quad (2.4)$$

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

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

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

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

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 $r \frac{J_1, \dots, J_n}{J}$ 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 ammissibile 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

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

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

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

Inoltre si parla di contesti rigidi, ovvero senza possibilità di scambio. Come appare dall'esempio sotto.

$$\begin{aligned} [x \in Nat, y \in Nat, z \in Mat(x, y)] \text{ cont} &\Rightarrow \text{è derivabile} \\ [y \in Nat, x \in Nat, z \in Mat(x, y)] \text{ cont} &\Rightarrow \text{è derivabile} \\ [y \in Nat, z \in Mat(x, y), x \in Nat] &\Rightarrow \text{non è un contesto.} \end{aligned}$$

Perciò non esiste lo scambio arbitrario, si deve porre attenzione alle dipendenze 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.

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

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

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

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

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

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

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

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

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 $C = D \text{ type}[\Gamma]$ sse due tipi hanno gli stessi elementi (come già accade in *set theory*), ma può non essere sempre vero.

Regole di Conversione dell'uguaglianza

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

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 $A \text{ type}[\Gamma]$, $a \in A[\Gamma]$ 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 ($[]$).

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

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 $[x_1 \in N_1 \dots x_n \in N_1] \text{ cont}$ e ottenere, così, contesti di una lista arbitraria di variabili diverse appartenenti a N_1 , come si vede dall'esempio seguente.

$$\begin{array}{c} \text{F-c)} \frac{[\] \text{ cont}}{[x_1 \in N_1] \text{ cont}} x_1 \in N_1 \notin [\] \\ \text{F-S)} \frac{[x_1 \in N_1] \text{ cont}}{N_1 \text{ type} [x_1 \in N_1]} \\ \text{F-c)} \frac{N_1 \text{ type} [x_1 \in N_1]}{[x_1 \in N_1, x_2 \in N_1] \text{ cont}} x_2 \in N_1 \notin x_1 \in N_1 \end{array}$$

2.2.2 Regole di Introduzione

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

Sia N_1 in ogni contesto Γ , partendo da contesto \emptyset , 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

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

2.2.3 Regole di Eliminazione

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

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

2.2.4 Regole di Conversione

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

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

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

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

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

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

2.2.6 Osservazioni sul tipo singoletto

L'eliminatore $\text{El}_{N_1}(z, c)$ rappresenta una funzione definita per ricorsione su N_1 , difatti in $C\text{-}S$ si ha che $\text{El}_{N_1}(z, c)[\frac{z}{*}] = \text{El}_{N_1}(*, c)$.

Supposto che se $* \in N_1[\Gamma]$ in $E\text{-}S$, allora per la singola conversione vale che $\text{El}_{N_1} = c \in M(*)$. Dunque $\text{El}_{N_1}(z, c)$ rappresenta un programma funzionale per ricorsione. Questi è definito su N_1 , a partire da $c \in M(*)$, perciò $\text{El}_{N_1}(*, c) = c$.

La regola di eliminazione permette di definire un programma funzionale da N_1 a $M(z)$ esclusivamente con $c \in M(*)$, ovvero definendo $*$ come **elemento canonico**. Inoltre non svolge solo il compito di ricorsione, ma anche d'nduzione.

In

$$\frac{t \in N_1[\Gamma]}{t = * \in N_1[\Gamma]}$$

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.

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

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

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

$$\frac{J_1, \dots, J_n}{J}$$

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

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

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

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

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

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

Se $a \in A[\Gamma]$ è derivabile in T allora anche $A \text{ type}[\Gamma]$ è derivabile in T .

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

Se $a = b \in A[\Gamma]$ è derivabile in T allora anche $a \in A[\Gamma]$ e $b \in A[\Gamma]$ 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 tv elemento canonico.

Tali regole sono del tipo k e rispettano la forma $\frac{[\Gamma] \text{ cont}}{T \text{ type}[\Gamma]}$

2. **Regole di Introduzione** identificate con il preambolo I- T , con T teoria dei tipi in esame e tv elemento canonico.

Tali regole consistono nella forma introduttiva degli elementi canonici di T e rispettano la forma $\frac{[\Gamma] \text{ cont}}{\text{tv} \in T[\Gamma]}$

3. **Regole di Eliminazione** identificate con il preambolo E- T con T teoria dei tipi in esame e tv elemento canonico.

Tali regole sono definenti E_k , a partire dagli elementi di k a valori in un tipo $M(z) \text{ type}[\Gamma, z \in k]$. 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 $\frac{t \in T[\Gamma] \quad M(z) \text{ type}[\Gamma, z \in T] \quad c \in M(\text{tv})[\Gamma]}{El_T(t, c) \in M(t)[\Gamma]}$

4. **Regole di Conversione** identificate con il preambolo C- T , con T teoria dei tipi in esame e tv elemento canonico.

, che stabiliscono che gli eliminatori in (3) sono stabiliti per ricorsione a partire dalle ipotesi. Tali regole sono del tipo k e rispettano la forma $\frac{M(z) \text{ type}[\Gamma, z \in T] \quad c \in M(\text{tv})[\Gamma]}{El_T(\text{tv}, c) = c \in M(\text{tv})[\Gamma]}$

5. **Regole di Uguaglianza** identificate con il preambolo eq-E- T , con T teoria dei tipi in esame e tv 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

$$\frac{t = s \in T[\Gamma] \quad M(z) \text{ type}[\Gamma, z \in T] \quad c = d \in M(\text{tv})[\Gamma]}{El_T(t, c) = El_T(s, d) \in M(t)[\Gamma]}$$

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

$P_1, P_2: \text{Nat}^m \rightarrow \text{Nat}$ allora $P_1 = P_2$ sse $\forall n_1 \dots n_m$ e $P_1(n_1 \dots n_m) = P_2(n_1 \dots n_m)$ 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 $a \in A[\Gamma]$ e $b \in A[\Gamma]$ derivabili, nella nostra teoria T di *Martin-Löf*, esiste un algoritmo H ($a \in A[\Gamma]$, $b \in A[\Gamma]$) =

$$\begin{cases} \mathbf{sì} \text{ sse } a = b \in A[\Gamma] \text{ è derivabile in } T \\ \mathbf{no} \text{ sse } a = b \in A[\Gamma] \text{ non ha derivazione in } T \end{cases}$$

Il giudizio $a = b \in A[\Gamma]$ è decidibile, anche con J giudizio, in teoria dei tipi di *Martin-Löf*, derivabile. Percui con H si scrive: Giudizi di $T \rightarrow \{0,1\}$

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

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

2.5.1 Applicazione dell'uguaglianza definizionale tra termini

Definizione: i termini *untyped* sono

$$t = x \mid * \mid \text{El}_{N1}(t_1, t_2)$$

Definizione: relazione di riduzione

$$\rightarrow_1 \subseteq \text{term} \times \text{term}$$

$t_1 \rightarrow_1 t_2 \equiv t_1$ 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 $t \rightarrow_1 s$. Ovvero non è più riducibile a nulla.

Le forme normali sono i valori assumibili dai programmi.

Definizione: **Teoria di Validità**

Dati $t \in A[\Gamma]$ e $s \in A[\Gamma]$ in T_1 allora $\rightarrow_1 s \Rightarrow t = s \in A[\Gamma]$ derivabile in T .

termini	termini in forma canonica	termini non in forma canonica o introduttiva
termini in NF	*	$x, \text{El}_{N_1}(x, *)$
termini non in NF	\emptyset	$\text{El}_{N_1}(*, x)$

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 forme 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 operativa.

Definizione Riducibilità

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

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

Definizione Teorema di Confluenza \rightarrow_1 per T computabile

Dato t (termine) e s_1 e s_2 (in NF) tale che $t \text{ Red}_{NF} s_1$ e $t \text{ Red}_{NF} s_2$ 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'è l'unicità della forma normale.

Definizione Teorema della forma normale (debole)

Dato t termine della grammatica esiste s termine in NF tale che $t \text{ Red}_{NF} s$. Allora esistono $t \equiv h_1 \rightarrow_1 \dots \rightarrow_1 h_n$ con $n > 1$ se t non è già in NF; oppure $t \equiv s$ se t è già in NF.

Questo significa che è sempre possibile rendere un programma convergente. Ma si può dire di più: \nexists 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 \nexists un cammino di riduzioni \rightarrow_1 infinito).

In questo modo ogni strategia deterministica è convergente.

Con quanto appena enunciato sopra possiamo definire quanto segue.

Dato $t \in A[\Gamma]$ derivabile in T

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

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 $a \in A[\Gamma]$ e $b \in A[\Gamma]$, giudizi derivabili in T_i allora $a = b \in A[\Gamma]$ 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 $\mathbf{a} \text{ Red}_{NF} \mathbf{b}$
3. a in NF, b non in NF e $\mathbf{b} \text{ Red}_{NF} \mathbf{a}$
4. né a né b sono in NF esiste s in NF tale che $\mathbf{a} \text{ Red}_{NF} \mathbf{s}$ e $\mathbf{b} \text{ Red}_{NF} \mathbf{s}$

Per i punti elencanti sopra trova validità la relazione $\mathbf{a} \text{ Red}_{NF} \mathbf{NF(a)} \Rightarrow a = NF(a) \in A[\Gamma]$ 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 operativa del singoletto

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

- $\beta_{N1}\text{-red}$) $El_{N1}(*, t) \rightarrow_1 t$
- red_I) $\frac{t_1 \rightarrow_1 t_2}{El_{N1}(t_1, c) \rightarrow_1 El_{N1}(t_2, c)}$ red_{II}) $\frac{c_1 \rightarrow_1 c_2}{El_{N1}(t, c_1) \rightarrow_1 El_{N1}(t, c_2)}$
- red_I e red_{II} possono venire simultate da un'unica regola $\frac{t_1 \rightarrow_1 t_2 \quad c_1 \rightarrow_1 c_2}{El_{N1}(t_1, c_1) \rightarrow_1 El_{N1}(t_2, c_2)}$

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

2.7 Esercizi

1) Data

$$E\text{-}N_{1\text{prog}}) \frac{M(w) \text{ type } [\Gamma, w \in N_1] \quad d \in M(*) \text{ type } [\Gamma]}{El_{N1}(w, d) \in M(w) \text{ type } [\Gamma, w \in N_1]}$$

dimostrare che in T_1 la regola *E-N₁prog* 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.

In $E-N_{1prog}$ il contesto è $\Sigma \equiv \Gamma, w \in N_1$ che permette l'applicazione di E-S.

$$\begin{array}{c}
\text{s-checks} \frac{\overline{M(w) \text{ type}[\Gamma, w \in N_1]}}{[\Gamma, w \in N_1] \text{ cont}} \\
\text{var} \frac{[\Gamma, w \in N_1] \text{ cont}}{w \in N_1[\Gamma, w \in N_1]} \\
\text{E-S} \frac{\overline{1} \quad \overline{M(z) \text{ type}[\Gamma, z \in N_1, w \in N_1]} \quad \overline{d \in M(*)[\Gamma, w \in N_1]}}{\text{El}_{N_1}(w, d) \in M(w)[\Gamma, w \in N_1]}
\end{array}$$

1

$$\begin{array}{c}
\text{ind-ty} \frac{\overline{M(w) \text{ type}[\Gamma, w \in N_1]}}{\text{sub-typ} \frac{\overline{M(w) \text{ type}[\Gamma, z \in N_1, w \in N_1]} \quad \overline{M(z) \text{ type}[\Gamma, z \in N_1, w \in N_1]}}{M(z) \text{ type}[\Gamma, z \in N_1, w \in N_1]}} \\
\text{s-checks} \frac{\overline{M(w) \text{ type}[\Gamma, w \in N_1]}}{[\Gamma, w \in N_1] \text{ cont}} \\
\text{F-S} \frac{[\Gamma, w \in N_1] \text{ cont}}{N_1 \text{ type}[\Gamma, w \in N_1]} \\
\text{F-c} \frac{N_1 \text{ type}[\Gamma, w \in N_1]}{[\Gamma, z \in N_1, w \in N_1] \text{ cont}} \quad (z \in N_1) \notin \Gamma \\
\text{F-c} \frac{[\Gamma, z \in N_1, w \in N_1] \text{ cont}}{z \in N_1[\Gamma, z \in N_1, w \in N_1]} \quad (z \in N_1) \notin \Gamma \\
\text{var} \frac{z \in N_1[\Gamma, z \in N_1, w \in N_1]}{z \in N_1[\Gamma, z \in N_1, w \in N_1]}
\end{array}$$

Assumo che le premesse di $E-N_{1prog}$ ($M(w) \text{ type} [\Gamma, w \in N_1]$ e $d \in M(*)[\Gamma]$) siano valide, perciò è valido, dalla prova sopra, anche il giudizio di conclusione $\text{El}_{N_1}(w, d) \in M(w)[\Gamma, w \in N_1]$, 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 rimpiazzata la regola di eliminazione $E-S$ con la regola $E-N_{1prog}$, aggiungendovi le regole di indebolimento, sostituzione e di *sanitary checks*.

$$\begin{array}{c}
\text{E-}N_{1prog} \frac{\overline{M(w) \text{ type} [\Gamma, w \in N_1]} \quad \overline{d \in M(*)[\Gamma]}}{\text{El}_{N_1}(w, d) \in M(w)[\Gamma, w \in N_1]} \\
\text{E-S} \frac{t\lambda \in N_1[\Sigma] \quad \overline{M(w) \text{ type} [\Gamma, w \in N_1]} \quad \overline{d\lambda \in M(*)[\Sigma]}}{\text{El}_{N_1}(t\lambda, d\lambda) \in M(t\lambda)[\Gamma]}
\end{array}$$

Soluzione

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

$$\text{sub-typ} \frac{\overline{t\lambda \in N_1[\Gamma]} \quad \text{E-}N_{1prog} \frac{\overline{M(w) \text{ type} [\Gamma, w \in N_1]} \quad \overline{d \in D(*)[\Gamma]}}{\text{El}_{N_1}(w, d\lambda) \in M(w)[\Gamma, w \in N_1]}}{\text{El}_{N_1}(t\lambda, d\lambda) \in M(t\lambda)[\Gamma]}$$

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

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

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

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

2

$$\begin{array}{c}
\text{F-S } \frac{[] \text{ cont}}{N_1 \text{ type } []} \\
\text{F-c } \frac{\text{F-S } \frac{[] \text{ cont}}{N_1 \text{ type } []}}{x \in N_1 \text{ cont}} \quad (x \in N_1) \notin [] \\
\text{I-S } \frac{\text{F-c } \frac{[] \text{ cont}}{N_1 \text{ type } []}}{x \in N_1 \text{ cont}} \\
\text{E-S } \frac{\text{F-c } \frac{[] \text{ cont}}{N_1 \text{ type } []}}{* \in N_1[x \in N_1]} \\
\hline
\text{El}_{N_1}(x, *) \in N_1[x \in N_1]
\end{array}$$

Applicando la β *N₁red* allora $El_{N_1}(x, *) \not\rightarrow_1 El_{N_1}(x, *)$ non è uguale definizionalmente.

$$\frac{\text{F-S } \frac{[] \text{ cont}}{N_1 \text{ type}[]} \quad \text{F-c } \frac{\frac{\text{F-S } \frac{[] \text{ cont}}{N_1 \text{ type}[]} (y \in N_1) \not\in []}{y \in N_1 \text{ cont}} \quad \text{F-c } \frac{N_1 \text{ type}[y \in N_1]}{y \in N_1, z \in N_1 \text{ cont}} (z \in N_1) \not\in y \in N_1 \quad \text{F-c } \frac{[] \text{ cont}}{N_1 \text{ type}[]} (y \in N_1) \not\in []}{\text{F-S } \frac{N_1 \text{ type}[y \in N_1, z \in N_1]}{N_1 \text{ type}[y \in N_1, z \in N_1]}} \quad \text{var } \frac{\frac{\text{F-c } \frac{[] \text{ cont}}{N_1 \text{ type}[]} (y \in N_1) \not\in []}{y \in N_1 \text{ cont}}}{y \in N_1[y \in N_1]} \quad \text{I-S } \frac{* \in N_1[y \in N_1]}{* \in N_1[y \in N_1]} \quad \text{E-S } \frac{}{\text{El}_{N_1}(*, y) \in N_1[y \in N_1]}$$

Applicando la β N_1 red allora $\text{El}_{N_1}(*, y) \rightarrow_1 y$
 $\text{El}_{N_1}(*, y)$ è uguale definizionalmente.

4

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

Applicando la β N_1 red allora $\text{El}_{N_1}(x, y) \rightarrow_1$
 $\text{El}_{N_1}(*, y)$ non è uguale definizionalmente.

5

$$\begin{array}{c}
\begin{array}{c}
\mathbf{5}_A \\
\text{E-S } \frac{\text{El}_{N_1}(*, y) \in N_1[y \in N_1, x \in N_1]}{\text{El}_{N_1}(\text{El}_{N_1}(*, y), \text{El}_{N_1}(x, *)) \in N_1[y \in N_1, x \in N_1]}
\end{array}
\quad
\begin{array}{c}
\mathbf{5}_B \\
\frac{N_1 \text{ type}[y \in N_1, x \in N_1, z \in N_1]}{\text{El}_{N_1}(\text{El}_{N_1}(*, y), \text{El}_{N_1}(x, *)) \in N_1[y \in N_1, x \in N_1]}
\end{array}
\quad
\begin{array}{c}
\mathbf{5}_C \\
\frac{\text{El}_{N_1}(x, *) \in N_1[y \in N_1, x \in N_1]}{\text{El}_{N_1}(\text{El}_{N_1}(*, y), \text{El}_{N_1}(x, *)) \in N_1[y \in N_1, x \in N_1]}
\end{array}
\end{array}$$

 $\mathbf{5}_A$

$$\begin{array}{c}
\begin{array}{c}
\text{F-S } \frac{[] \text{ cont}}{N_1 \text{ type}[]} \\
\text{F-c } \frac{N_1 \text{ type}[]}{y \in N_1 \text{ cont}} (y \in N_1) \notin [] \\
\text{F-S } \frac{N_1 \text{ type}[y \in N_1]}{y \in N_1, x \in N_1 \text{ cont}} (x \in N_1) \notin (y \in N_1) \\
\text{F-c } \frac{y \in N_1, x \in N_1 \text{ cont}}{\text{El}_{N_1}(*, y) \in N_1[y \in N_1]} \\
\text{ind-te } \frac{\text{El}_{N_1}(*, y) \in N_1[y \in N_1]}{\text{El}_{N_1}(*, y) \in N_1[y \in N_1, x \in N_1]}
\end{array}
\end{array}$$

 $\mathbf{5}_B$

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

5_C

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

Per i giudizi conclusione $\text{El}_{N_1}(*, y) \in N_1[y \in N_1]$ e $\text{El}_{N_1}(x, *) \in N_1[x \in N_1]$ ho già dimostrato sopra (in 3 e 2) la loro tipabilità per il tipo singoletto. Applicando la red_I e $\beta_{N_1 \text{red}}$ allora $\text{El}_{N_1}(\text{El}_{N_1}(*, y), \text{El}_{N_1}(x, *)) \rightarrow_1 \text{El}_{N_1}(y, \text{El}_{N_1}(x, *))$.

Più nel dettaglio la riduzione è la seguente

$$\text{red}_I \frac{\beta_{N_1 \text{red}} \frac{\text{El}_{N_1}(*, y) \rightarrow_1 y}{\text{El}_{N_1}(\text{El}_{N_1}(*, y), \text{El}_{N_1}(x, *)) \rightarrow_1 \text{El}_{N_1}(y, \text{El}_{N_1}(x, *))}}{\text{El}_{N_1}(\text{El}_{N_1}(*, y), \text{El}_{N_1}(x, *)) \rightarrow_1 \text{El}_{N_1}(y, \text{El}_{N_1}(x, *))}$$

$\text{El}_{N_1}(\text{El}_{N_1}(*, y), \text{El}_{N_1}(x, *))$ è uguale definizionalmente.

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

$$t \equiv v \mid * \mid \text{El}_{N_1}(t_1, t_2)$$

con $v \in \{x, y, w, z\} \cup \{x_i \mid i \in \text{Nat}\}$, 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 \rightarrow_1 una relazione binaria tra questi termini *untyped* definita a partire dalle seguenti regole

$$\begin{array}{c}
\beta_{N_1} \text{red} \quad \text{El}_{Nat}(0, c) \rightarrow_1 c \\
\text{red}_I) \frac{t_1 \rightarrow t_2}{\text{El}_{N_1}(t_1, c) \rightarrow_1 \text{El}_{N_1}(t_2, c)} \quad \text{red}_{II}) \frac{c_1 \rightarrow c_2}{\text{El}_{N_1}(t, c_1) \rightarrow_1 \text{El}_{N_1}(t, c_2)}
\end{array}$$

- 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 \rightarrow_1 del termine

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

- Produrre un infinità di termini del tipo singoletto che non sono riducibili secondo la relazione di un passo di riduzione \rightarrow_1 .
Dati due di questi termini, si riesce a dire che sono definizionalmente 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 (\rightarrow_1), con la quale il termine si riduce eventualmente a forma normale (implicando la definizione di riducibilità).

β -red:

$$El_{N1}(*, *) \rightarrow_1 *$$

$$El_{N1}(*, x) \rightarrow_1 x$$

$$\frac{\beta\text{-}N_1 \text{ red } \frac{x}{El_{N1}(*, x)} \quad \frac{x}{El_{N1}(*, x)} \beta\text{-}N_1 \text{ red}}{\beta\text{-}N_1 \text{ red } \frac{El_{NI}(*, El_{N1}(*, x))}{El_{N1}(El_{N1}(*, *), El_{N1}(*, x))} \text{ red}_{II}} \text{ red}_I \quad \frac{\frac{x}{El_{N1}(*, x)} \beta\text{-}N_1 \text{ red}}{El_{N1}(El_{N1}(*, *), x) \text{ red}_I} \text{ red}_{II}$$

$\Rightarrow (w, (red_{II}I, red_I, \beta_{N1red}))$ 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 rispetteranno la definizione data sopra

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

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

Capitolo 3

Tipo dei numeri Naturali

3.1 Regole di Formazione

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

3.2 Regole di Introduzione

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

3.3 Regole di Eliminazione

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

3.4 Regole di Conversione

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

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

3.5 Regole di Uguaglianza

$$\text{eq-Nat)} \frac{t_1 = t_2 \in \text{Nat}[\Gamma]}{\text{succ}(t_1) = \text{succ}(t_2) \in \text{Nat}[\Gamma]}$$

3.6 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.

$$I_2\text{-Nat}_{prog}) \frac{\Gamma \text{ cont}}{\text{succ}(x) \in \text{Nat}[\Gamma, x \in \text{Nat}]}$$

Il medesimo discorso vale per la regola di eliminazione

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

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

3.7 Semantica operativa dei numeri Naturali

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

- $\beta_{1Nat\text{-red}}) El_{Nat}(0, c, e) \rightarrow_1 c$
- $\beta_{2Nat\text{-red}}) El_{Nat}(\text{succ}(m), c, e) \rightarrow_1 e(m, El_{Nat}(m, c, e))$
- $\text{Nat-red}_1) \frac{t_1 \rightarrow_1 t_2}{El_{Nat}(t_1, c, e) \rightarrow_1 El_{Nat}(t_2, c, e)} \quad \text{Nat-red}_2) \frac{c_1 \rightarrow_1 c_2}{El_{Nat}(t, c_1, e) \rightarrow_1 El_{Nat}(t, c_2, e)}$
- Novità dei numeri naturali rispetto al tipo singoletto $\text{Nat-red}) \frac{t_1 \rightarrow_1 t_2}{\text{succ}(t_1) \rightarrow_1 \text{succ}(t_2)}$

3.8 Primitiva ricorsiva

Definizione

$\text{Nat}^n \times \text{Nat} \rightarrow \text{Nat}$

Dati $g_0: \text{Nat}^m \rightarrow \text{Nat}$ e $g_1: \text{Nat}^m \times \text{Nat} \times \text{Nat} \rightarrow \text{Nat}$

$n_1 \dots n_m \in \text{Nat}$ allora

$\text{rec}(n_1 \dots n_m, 0) \equiv g_0(n_1 \dots n_m)$

$\text{rec}(n_1 \dots n_m, k+1) \equiv g_0(n_1 \dots n_m, k, \text{rec}(n_1 \dots n_m, k))$

3.9 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 $E\text{-Nat}_{dip}$ (§3.6). In questo modo si riesce a definire per la nostra teoria T_{1Nat}

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

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

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

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

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

- 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?
 $\text{El}_{Nat}(0, w, (x,y).\text{succ}(y)) \rightarrow_1 w$ per $\beta_{1Nat\text{-red}} (\equiv w + 0 = w)$
- Passo induttivo: ricorro su z (esempio $\text{succ}(0) = 1$)
 $\text{El}_{Nat}(\text{succ}(0), w, (x,y).\text{succ}(y))$ per $\beta_{1Nat\text{-red}} \rightarrow_1 \text{succ}(\text{El}_{Nat}(0, w, (x,y).\text{succ}(y))) \rightarrow_1 \text{succ}(w)$
Dunque $w + 1 = \text{succ}(w) \in \text{Nat}$

\Rightarrow Il programma fa effettivamente quello che dovrebbe.

3.9.1 Osservazioni sull'addizione

$w +_1 z \equiv \text{El}_{Nat}(z, w, (x,y), \text{succ}(y)) \neq$ come NF da z .

Se sostituisco w con 0, allora $0 +_1 z \equiv w +_1 z[\frac{w}{0}] \equiv \text{El}_{Nat}(z, 0, (x,y), \text{succ}(y))$ è in NF (dunque non riesco più a ridurla ulteriormente).

Ecco che $0 +_1 z$ è un valore in NF \neq da z , da cui è impossibile dimostrare che $0 +_1 z = z \in \text{Nat}[z \in \text{Nat}]$.

Questo non accade per $w +_1 0 = w$ (§3.9).

Perciò, se noi scriviamo la somma ricorrendo 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.

3.10 Esercizi

2_A

$$\begin{array}{c}
\alpha - eq \frac{\overline{M(z) \text{ type}[\Gamma, z \in \text{Nat}]}}{\overline{M(x) \text{ type}[\Gamma, x \in \text{Nat}]}} \\
\text{F-c} \frac{\overline{M(x) \text{ type}[\Gamma, x \in \text{Nat}]}}{\Gamma, x \in \text{Nat}, y \in M(x) \text{ cont}} (y \in M(x)) \notin (\Gamma, x \in \text{Nat}) \\
\text{F-Nat} \frac{\overline{\text{Nat type}[\Gamma, x \in \text{Nat}, y \in M(x)]}}{\Gamma, x \in \text{Nat}, y \in M(x), z \in \text{Nat cont}} (z \in \text{Nat}) \notin (\Gamma, x \in \text{Nat}, y \in M(x))
\end{array}$$

 2_B

$$\begin{array}{c}
\text{F-Nat} \frac{\Gamma \text{ cont}}{\text{Nat type}[\Gamma]} \\
\text{F-c} \frac{\Gamma \text{ cont}}{\Gamma, z \in \text{Nat cont}} (z \in \text{Nat}) \notin \Gamma \\
\text{F-Nat} \frac{\text{Nat type}[\Gamma, z \in \text{Nat}]}{\Gamma, z \in \text{Nat, x} \in \text{Nat cont}} (x \in \text{Nat}) \notin (\Gamma, z \in \text{Nat}) \\
\text{F-c} \frac{\text{Nat type}[\Gamma, z \in \text{Nat}]}{\Gamma, z \in \text{Nat, x} \in \text{Nat cont}} (x \in \text{Nat}) \notin (\Gamma, z \in \text{Nat}) \\
\text{var} \frac{\Gamma, z \in \text{Nat, x} \in \text{Nat cont}}{x \in \text{Nat}[\Gamma, z \in \text{Nat, x} \in \text{Nat}]}
\end{array}$$

 2_C

$$\text{ind-ty} \frac{\overline{M(z) \text{ type}[\Gamma, z \in \text{Nat}]} \quad \text{F-c} \frac{(*)}{\Gamma, z \in \text{Nat, x} \in \text{Nat cont}} (x \in \text{Nat}) \notin (\Gamma, z \in \text{Nat})}{M(z) \text{ type}[\Gamma, z \in \text{Nat, x} \in \text{Nat}]}$$

Le derivazioni contrassegnate con (*) indicano che le ho già svolte all'interno dell'esercizio e che le ho omesse per evitare ripetizioni.

Assumo che le premesse di $E\text{-Nat}_{dip} (M(z) \text{ type}[\Gamma, z \in \text{Nat}], c \in M(0)/\Gamma, e(x,y) \in M(\text{succ}(x))/\Gamma, x \in \text{Nat}, y \in M(x))$ siano valide e che $\Gamma \text{ cont}$ sia assioma ($[\] \text{ cont} = [\Gamma] \text{ cont}$ con $\Gamma = \emptyset$), perciò è valido, dalla prova sopra, anche il giudizio di conclusione $El_{Nat}(z, c, e) \in M(z)/\Gamma, w \in \text{Nat}$, di conseguenza derivabile in T .

2) Definire $w + 2 \in \text{Nat}[w \in \text{Nat}]$, ove 2 è l'abbreviazione del termine ottenuto applicando $2 \equiv \text{succ}(\text{succ}(0))$.

Soluzione

La ricorsione la faccio w , usando lo schema di ricorsione primitiva, vale che $w + 2 \equiv El_{Nat}(w, 2, (x,y).\text{succ}(y)) \in \text{Nat}[w \in \text{Nat}]$

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

Dimostrazione di correttezza di $El_{Nat}(w, 2, (x,y).\text{succ}(y)) \in \text{Nat}[w \in \text{Nat}]$

- $El_{Nat}(0, 2, (x,y).\text{succ}(y)) \rightarrow_1 2$ per $\beta_{1Nat-red}$

- $\text{El}_{Nat}(\text{succ}(m), 2, (x,y).\text{succ}(y)) \rightarrow_1 \text{succ}(\text{El}_{Nat}(m, 2, (x,y).\text{succ}(y)))$ per $\beta_{2Nat-red} \Rightarrow$ per $m = 0 \equiv \text{succ}(\text{El}_{Nat}(0, 2, (x,y).\text{succ}(y))) \rightarrow_1 \text{succ}(2) \in \text{Nat} = 3$ (dal punto precedente).

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

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

in modo tale che valga $x + 0 = x \in \text{Nat}[x \in \text{Nat}]$

Soluzione

La ricorsione la faccio sulla y , usando lo schema di ricorsione primitiva, vale che $x + y \equiv \text{El}_{Nat}(y, x, (w,z).\text{succ}(z)) \in \text{Nat}[x \in \text{Nat}, y \in \text{Nat}]$

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

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

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

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

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

in modo tale che valga $0 + y = y \in \text{Nat}[y \in \text{Nat}]$

Soluzione

La ricorsione la faccio sulla x , per cui, usando lo schema di ricorsione primitiva, vale che $x + y \equiv \text{El}_{Nat}(x, y, (w,z).\text{succ}(z)) \in \text{Nat}[x \in \text{Nat}, y \in \text{Nat}]$

$$\begin{array}{c} \text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad (x \in \text{Nat}) \notin [] \\ \text{F-c} \frac{}{x \in \text{Nat cont}} \\ \text{F-Nat} \frac{\text{Nat type}[x \in \text{Nat}]}{x \in \text{Nat, } y \in \text{Nat cont}} \quad (y \in \text{Nat}) \notin x \in \text{Nat} \\ \text{F-c} \frac{}{x \in \text{Nat, } y \in \text{Nat cont}} \\ \text{ex-te} \frac{\text{El}_{Nat}(x, y, (w,z).\text{succ}(z)) \in \text{Nat}[y \in \text{Nat}, x \in \text{Nat}]}{\text{El}_{Nat}(x, y, (w,z).\text{succ}(z)) \in \text{Nat}[x \in \text{Nat}, y \in \text{Nat}]} \end{array}$$

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

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

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

5) Un concetto importante del λ -calcolo è α -equivalenza (α -eq). Questa consiste in una forma base di equivalenza, definibile in termini *lambda*, dove una qualsiasi variabile legata può essere convertita in un'altra. Per esempio, $\lambda x.x$ e $\lambda y.y$ *lambda* sono termini α -eq che rappresentano la medesima funzione.

Nella dispensa ho utilizzato α -eq per provare la validità delle conclusioni, quando ne sussistevano le premesse per farlo.

Di seguito dimostro la validità di α -eq derivando la sua stessa regola.

$$\frac{M(x) \text{ type}[x \in A]}{M(w) \text{ type}[w \in A]}$$

Soluzione

Assumo per ipotesi che A sia un tipo non dipendente ($A \equiv \text{Nat}$), vale però per qualsiasi teoria dei tipi.

$$\begin{array}{c}
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type}[]} \\
\text{F-c} \frac{\text{Nat type}[]}{w \in \text{Nat cont}} (w \in \text{Nat}) \notin [] \\
\text{var} \frac{\text{Nat type}[w \in \text{Nat}]}{w \in \text{Nat}[w \in \text{Nat}]} \\
\text{sub-typ} \frac{\text{Nat type}[w \in \text{Nat}]}{\text{Nat type}[w \in \text{Nat}]} \\
\text{ex-ty} \frac{\text{M}(x) \text{ type}[x \in \text{Nat, } w \in \text{Nat}]}{\text{M}(x) \text{ type}[w \in \text{Nat, } x \in \text{Nat}]} \\
\text{M}(w) \text{ type}[w \in \text{Nat}]
\end{array}$$

1

$$\begin{array}{c}
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type}[]} \\
\text{F-c} \frac{\text{Nat type}[]}{x \in \text{Nat cont}} (x \in \text{Nat}) \notin [] \\
\text{F-Nat} \frac{\text{Nat type}[x \in \text{Nat}]}{x \in \text{Nat, } w \in \text{Nat cont}} (w \in \text{Nat}) \notin x \in \text{Nat} \\
\text{F-c} \frac{\text{Nat type}[x \in \text{Nat}]}{x \in \text{Nat, } w \in \text{Nat cont}} \\
\text{ind-ty} \frac{\text{M}(x) \text{ type}[x \in \text{Nat}]}{\text{M}(x) \text{ type}[x \in \text{Nat, } w \in \text{Nat}]}
\end{array}$$

2

$$\begin{array}{c}
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type}[]} \\
\text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type}[]}}{w \in \text{Nat cont}} (w \in \text{Nat}) \notin [] \\
\text{F-Nat} \frac{\text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type}[]}}{w \in \text{Nat cont}}}{\text{Nat type}[w \in \text{Nat}]} \\
\text{F-c} \frac{\text{F-Nat} \frac{\text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type}[]}}{w \in \text{Nat cont}}}{\text{Nat type}[w \in \text{Nat}]}}{w \in \text{Nat}, x \in \text{Nat cont}} (x \in \text{Nat}) \notin w \in \text{Nat}
\end{array}$$

Capitolo 4

Tipo delle liste di un tipo

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

4.1 Regole di Formazione

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

4.2 Regole di Introduzione

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

4.3 Regole di Eliminazione

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

4.4 Regole di Conservazione

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

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

4.5 Regole di Uguaglianza

$$\begin{array}{c}
 \text{eq-I}_1\text{-List) } \frac{A_1 = A_2 \in \text{type}[\Gamma]}{\text{List}(A_1) = \text{List}(A_2) \text{ type}(\Gamma)} \\
 \\
 \text{eq-I}_2\text{-List) } \frac{s_1 = s_2 \in \text{List}(A)[\Gamma] \quad a_1 = a_2 \in A[\Gamma]}{\text{cons}(s_1, a_1) = \text{cons}(s_2, a_2) \in \text{List}(A)(\Gamma)} \\
 \\
 \text{E-eq-List) } \frac{\begin{array}{c} M(z) \text{ type}[\Gamma, z \in \text{List}(A)] \\ t_1 = t_2 \in \text{list}(A)[\Gamma] \\ c_1 = c_2 \in M(\text{nil})[\Gamma] \end{array} \quad \begin{array}{c} e_1(x, w, y) = e_2(x, w, y) \in M(\text{cons}(x, w))[\Gamma, x \in \text{List}(A), w \in A, y \in M(x)] \end{array}}{\text{El}_{list}(t_1, c_1, e_1) = \text{El}_{list}(t_2, c_2, e_2) \in M(t_1)[\Gamma]}
 \end{array}$$

4.6 Eliminatore dipendente

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

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

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

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

4.7 Semantica operativa del tipo lista

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

- $\beta_{1list}\text{-red}) \text{El}_{list}(\text{nil}, c, e) \rightarrow_1 c$
- $\beta_{2list}\text{-red}) \text{El}_{list}(\text{cons}(s, a), c, e) \rightarrow_1 e(s, a, \text{El}_{list}(s, c, e))$
- $\text{List-red}_1) \frac{t_1 \rightarrow_1 t_2}{\text{El}_{list}(t_1, c, e) \rightarrow_1 \text{El}_{list}(t_2, c, e)} \quad \text{List-red}_2) \frac{c_1 \rightarrow_1 c_2}{\text{El}_{list}(t, c_1, e) \rightarrow_1 \text{El}_{list}(t, c_2, e)}$
- Novità delle liste rispetto al tipo singoletto $\text{List-red}_I) \frac{s_1 \rightarrow_1 s_2}{\text{cons}(s_1, a) \rightarrow_1 \text{cons}(s_2, a)}$
- $\text{List-red}_{II}) \frac{a_1 \rightarrow_1 a_2}{\text{cons}(s, a_1) \rightarrow_1 \text{cons}(s, a_2)}$

4.8 Esercizi

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:

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

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

$$\text{append}_1(x', y') \in \text{List}(A)[x' \in \text{List}(A), y' \in A]$$

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

$$\text{append}_1(x', y') = \text{cons}(x', y') \in \text{List}(A)[x' \in \text{List}(A), y' \in A]$$

Ma se si vuole definire un'operazione

$$\text{append}_2(x', y') \in \text{List}(A)[x' \in \text{List}(A), y' \in A]$$

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 dare equazionalmente la definizione ricorsiva di append_2

$$\begin{aligned} \text{append}_2(\text{nil}, y') &= \text{cons}(\text{nil}, y') = y' \\ \text{append}_2(\text{cons}(s, x'), y') &= \text{cons}(\text{append}_2(s, y'), x') \end{aligned}$$

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

• Premesse:

- $M(z) \text{ type}[\Gamma, z \in \text{List}(A)] \equiv \text{List}(A) \text{ type}[x' \in \text{List}(A), y' \in A]$
- $t \in \text{List}(A)[\Gamma] \equiv x' \in \text{List}(A)[x' \in \text{List}(A), y' \in A]$
- $c \in M(\text{nil})[\Gamma] \equiv y' \in A[x' \in \text{List}(A), y' \in A]$
- $e(x, w, y) \in M(\text{cons}(x, w))[\Gamma, x \in \text{List}(A), w \in A, y \in M(x)] \equiv \text{cons}(z, y) \in \text{List}(A)[x' \in \text{List}(A), y' \in A, z \in \text{List}(A), y \in A]$

• Giudizio di conclusione:

- $\text{El}_{list}(t, c, e) \in M(t)[\Gamma] \equiv \text{El}_{list}(x', y', (x, y, z). \text{cons}(z, y)) \in \text{List}(A)[x' \in \text{List}(A), y' \in A]$

$$\text{E-list} \frac{\begin{array}{cccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \end{array}}{\text{El}_{list}(x', y', (x, y, z). \text{cons}(z, y)) \in \text{List}(A)[x' \in \text{List}(A), y' \in A]}$$

1

$$\begin{array}{c}
\text{ind-ty} \frac{\frac{\frac{[] \text{ cont}}{A \text{ type}[]}}{\text{ind-ty} \frac{A \text{ type}[x' \in \text{List}(A)]}{F-c \frac{A \text{ type}[x' \in \text{List}(A), y' \in A \text{ cont}}{x' \in \text{List}(A)}}} \quad \frac{\frac{\overline{A \text{ type}[]}}{F-c \frac{\text{List}(A) \text{ type}[]}{x' \in \text{List}(A) \text{ cont}}} (x' \in \text{List}(A)) \notin []}{F-c \frac{A \text{ type}[x' \in \text{List}(A)]}{x' \in \text{List}(A), y' \in A \text{ cont}}} (y' \in A) \notin x' \in \text{List}(A)} \\
\frac{\overline{A \text{ type}[]}}{F-c \frac{A \text{ type}[x' \in \text{List}(A), y' \in A]}{\text{List}(A) \text{ type}[x' \in \text{List}(A), y' \in A]}}
\end{array}$$

2

$$\begin{array}{c}
\text{ind-ty} \frac{\frac{\overline{A \text{ type}[]}}{F-c \frac{A \text{ type}[y' \in A]}{\text{List}(A) \text{ type}[y' \in A]} (y' \in A) \notin []} \quad \frac{\overline{A \text{ type}[]}}{F-c \frac{\text{List}(A) \text{ type}[]}{x' \in \text{List}(A) \text{ cont}}} (x' \in \text{List}(A)) \notin []}{\frac{\overline{A \text{ type}[]}}{F-c \frac{A \text{ type}[x' \in \text{List}(A)]}{x' \in \text{List}(A), y' \in A \text{ cont}}} (y' \in A) \notin x' \in \text{List}(A)} \\
\frac{\frac{\overline{A \text{ type}[]}}{F-c \frac{A \text{ type}[y' \in A]}{\text{List}(A) \text{ type}[y' \in A]} (y' \in A) \notin []} \quad \frac{\overline{A \text{ type}[]}}{F-c \frac{\text{List}(A) \text{ type}[]}{x' \in \text{List}(A) \text{ cont}}} (x' \in \text{List}(A)) \notin []}{\frac{\overline{A \text{ type}[]}}{F-c \frac{A \text{ type}[x' \in \text{List}(A)]}{x' \in \text{List}(A), y' \in A \text{ cont}}} (y' \in A) \notin x' \in \text{List}(A)} \\
\frac{\text{var} \frac{x' \in \text{List}(A)[y' \in A, x' \in \text{List}(A)]}{x' \in \text{List}(A)[x' \in \text{List}(A), y' \in A]}}{\text{ex-ty} \frac{x' \in \text{List}(A)[x' \in \text{List}(A), y' \in A]}{x' \in \text{List}(A)[x' \in \text{List}(A), y' \in A]}}
\end{array}$$

3

$$\begin{array}{c}
\text{ind-ty} \frac{\frac{\overline{A \text{ type}[]}}{F-c \frac{\text{List}(A) \text{ type}[]}{x' \in \text{List}(A) \text{ cont}}} (x' \in \text{List}(A)) \notin []}{\frac{\overline{A \text{ type}[]}}{F-c \frac{A \text{ type}[x' \in \text{List}(A)]}{x' \in \text{List}(A), y' \in A \text{ cont}}} (y' \in A) \notin x' \in \text{List}(A)} \\
\frac{\text{var} \frac{x' \in \text{List}(A)[x' \in \text{List}(A), y' \in A]}{y' \in A[x' \in \text{List}(A), y' \in A]}}{\text{ex-ty} \frac{x' \in \text{List}(A)[x' \in \text{List}(A), y' \in A]}{x' \in \text{List}(A)[x' \in \text{List}(A), y' \in A]}}
\end{array}$$

4

$$\begin{array}{c}
\text{I}_2\text{-list} \frac{\frac{\text{var} \frac{x' \in \text{List}(A), y' \in A, z \in \text{List}(A), y \in A}{z \in \text{List}(A)[x' \in \text{List}(A), y' \in A, z \in \text{List}(A), y \in A]} \quad \frac{\text{var} \frac{x' \in \text{List}(A), y' \in A, z \in \text{List}(A), y \in A}{y \in A[x' \in \text{List}(A), y' \in A, z \in \text{List}(A), y \in A]}}{\text{cons}(z, y) \in \text{List}(A)[x' \in \text{List}(A), y' \in A, z \in \text{List}(A), y \in A]}
\end{array}$$

4'

$$\begin{array}{c}
\text{ind-ty } \frac{\overline{\text{A type}[]}}{\text{F-c } \frac{\text{A type}[x' \in \text{List}(A)]}{x' \in \text{List}(A), y' \in A \text{ cont}}} (y' \in A) \not\subset x' \in \text{List}(A) \\
\\
\text{ind-ty } \frac{\overline{\text{A type}[]}}{\text{F-c } \frac{\text{A type}[x' \in \text{List}(A), y' \in A]}{x' \in \text{List}(A), y' \in A, z \in \text{List}(A) \text{ cont}}} (z \in \text{List}(A)) \not\subset (x' \in \text{List}(A), y' \in A) \\
\\
\text{ind-ty } \frac{\overline{\text{A type}[]}}{\text{F-c } \frac{\text{A type}[x' \in \text{List}(A), y' \in A, z \in \text{List}(A)]}{x' \in \text{List}(A), y' \in A, z \in \text{List}(A), v \in A}} (v \in A) \not\subset (x' \in \text{List}(A), y' \in A, z \in \text{List}(A))
\end{array}$$

Dimostrazione di correttezza di $El_{list}(x^{\wedge}, y^{\wedge}, (x, y, z).cons(z, y)) \in List(A)[x^{\wedge} \in List(A), y^{\wedge} \in A]$

- $\text{El}_{list}(\text{nil}, y^1, (x, y, z).cons(z, y)) \rightarrow_1 y^1$ per $\beta_{1list-red}$
- $\text{El}_{list}(cons(s, x^1), y^1, (x, y, z).cons(z, y)) \rightarrow_1 cons(\text{El}_{list}(x^1, y^1, (x, y, z).cons(z, y)))$
per $\beta_{2list-red} \Rightarrow$ per $x^1 = \text{nil} \equiv cons(\text{El}_{list}(\text{nil}, y^1, (x, y, z).cons(z, y))) \rightarrow_1$
 $cons(\text{nil}, y^1) \in \text{List}(A) = y^1$ (dal punto precedente).

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

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

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

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

$$\begin{aligned}\text{append}(\text{x}, \text{nil}) &= \text{cons}(\text{x}, \text{nil}) = \text{x} \\ \text{append}(\text{x}, \text{cons}(\text{y}, \text{a})) &= \text{cons}(\text{x}, \text{append}(\text{y}, \text{a}))\end{aligned}$$

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

- Premesse:
 - $M(z) \text{ type}[\Gamma, z \in \text{List}(A)] \equiv \text{List}(A) \text{ type}[x \in \text{List}(A), y \in \text{List}(A)]$
 - $t \in \text{List}(A)[\Gamma] \equiv y \in \text{List}(A)[x \in \text{List}(A), y \in \text{List}(A)]$
 - $c \in M(\text{nil})[\Gamma] \equiv x \in \text{List}(A)[x \in \text{List}(A), y \in \text{List}(A)]$
 - $e(x, w, y) \in M(\text{cons}(x, w))[\Gamma, x \in \text{List}(A), w \in A, y \in M(x)] \equiv \text{cons}(z, w) \in \text{List}(A)[x \in \text{List}(A), y \in \text{List}(A), z \in \text{List}(A), w \in A]$
- Giudizio di conclusione:
 - $\text{El}_{list}(t, c, e) \in M(t)[\Gamma] \equiv \text{El}_{list}(y, x, (u, v, z). \text{cons}(z, v)) \in \text{List}(A)[x \in \text{List}(A), y \in \text{List}(A)]$

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

1

$$\begin{array}{c} \text{ind-ty} \frac{\frac{\text{A type}[]}{\text{F-c} \frac{\text{List}(A) \text{ type} []}{x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin []}{\text{F-c} \frac{\text{List}(A) \text{ type}[x \in \text{List}(A)]}{x \in \text{List}(A), y \in \text{List}(A) \text{ cont}}} (y \in \text{List}(A)) \notin x \in \text{List}(A)} \\ \text{ind-ty} \frac{\frac{\text{A type}[]}{\text{F-c} \frac{\text{List}(A) \text{ type}[x \in \text{List}(A), y \in \text{List}(A)]}{x \in \text{List}(A), y \in \text{List}(A) \text{ cont}}} (y \in \text{List}(A)) \notin x \in \text{List}(A)}{\text{F-c} \frac{\text{List}(A) \text{ type}[x \in \text{List}(A), y \in \text{List}(A)]}{\text{List}(A) \text{ type}[x \in \text{List}(A), y \in \text{List}(A)]}} \end{array}$$

2

$$\begin{array}{c} \text{ind-ty} \frac{\frac{\text{A type}[]}{\text{F-c} \frac{\text{List}(A) \text{ type} []}{x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin []}{\text{F-c} \frac{\text{List}(A) \text{ type}[x \in \text{List}(A)]}{x \in \text{List}(A), y \in \text{List}(A) \text{ cont}}} (y \in \text{List}(A)) \notin x \in \text{List}(A)} \\ \text{var} \frac{\text{F-c} \frac{\text{List}(A) \text{ type}[x \in \text{List}(A)]}{x \in \text{List}(A), y \in \text{List}(A) \text{ cont}}} (y \in \text{List}(A)) \notin x \in \text{List}(A)}{y \in \text{List}(A)[x \in \text{List}(A), y \in \text{List}(A)]} \end{array}$$

3

$$\begin{array}{c} \text{ind-ty} \frac{\frac{\text{A type}[]}{\text{F-c} \frac{\text{List}(A) \text{ type} []}{y \in \text{List}(A) \text{ cont}}} (y \in \text{List}(A)) \notin []}{\text{F-c} \frac{\text{List}(A) \text{ type}[y \in \text{List}(A)]}{y \in \text{List}(A), x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin y \in \text{List}(A)} \\ \text{var} \frac{\text{F-c} \frac{\text{List}(A) \text{ type}[y \in \text{List}(A)]}{y \in \text{List}(A), x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin y \in \text{List}(A)}{x \in \text{List}(A)[y \in \text{List}(A), x \in \text{List}(A)]} \\ \text{ex-ty} \frac{\text{var} \frac{\text{F-c} \frac{\text{List}(A) \text{ type}[y \in \text{List}(A)]}{y \in \text{List}(A), x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin y \in \text{List}(A)}{x \in \text{List}(A)[x \in \text{List}(A), y \in \text{List}(A)]} \end{array}$$

4

$$\begin{array}{c} \text{4'} \quad \text{4'} \\ \text{var} \frac{\text{var} \frac{\text{F-c} \frac{\text{List}(A) \text{ type}[y \in \text{List}(A)]}{y \in \text{List}(A), x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin y \in \text{List}(A)}{x \in \text{List}(A)[y \in \text{List}(A), x \in \text{List}(A)]} \quad \text{var} \frac{\text{F-c} \frac{\text{List}(A) \text{ type}[y \in \text{List}(A)]}{y \in \text{List}(A), x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin y \in \text{List}(A)}{x \in \text{List}(A)[y \in \text{List}(A), x \in \text{List}(A)]} \\ \text{I}_2\text{-list} \frac{\text{var} \frac{\text{F-c} \frac{\text{List}(A) \text{ type}[y \in \text{List}(A)]}{y \in \text{List}(A), x \in \text{List}(A) \text{ cont}}} (x \in \text{List}(A)) \notin y \in \text{List}(A)}{x \in \text{List}(A)[x \in \text{List}(A), y \in \text{List}(A)]} \end{array}$$

4[^]

$$\begin{array}{c}
\text{ind-ty} \frac{\overline{A \text{ type} []}}{\text{F-c} \frac{A \text{ type}[x \in \text{List}(A), y \in \text{List}(A), z \in \text{List}(A)]}{x \in \text{List}(A) \ y \in \text{List}(A), z \in \text{List}(A), w \in A \text{ cont}} (w \in A) \notin (x \in \text{List}(A), y \in \text{List}(A), z \in \text{List}(A))} \\
\text{ind-ty} \frac{\overline{A \text{ type}[]}}{\text{F-c} \frac{\overline{A \text{ type}[x \in \text{List}(A), y \in \text{List}(A)]}}{\text{F-c} \frac{\overline{\text{List}(A) \text{ type}[x \in \text{List}(A), y \in \text{List}(A)]}}{x \in \text{List}(A), y \in \text{List}(A), z \in \text{List}(A) \text{ cont}} (z \in \text{List}(A)) \notin (x \in \text{List}(A), y \in \text{List}(A))} \\
\text{ind-ty} \frac{\overline{A \text{ type}[]}}{\text{F-c} \frac{\overline{A \text{ type}[x \in \text{List}(A)]}}{\text{F-c} \frac{\overline{\text{List}(A) \text{ type}[x \in \text{List}(A)]}}{x \in \text{List}(A), y \in \text{List}(A) \text{ cont}} (y \in \text{List}(A)) \notin x \in \text{List}(A)} \\
\text{ind-ty} \frac{\overline{A \text{ type}[]}}{\text{F-c} \frac{\overline{\text{List}(A) \text{ type}[]}}{\text{F-c} \frac{\overline{A \text{ type}[]}}{x \in \text{List}(A) \text{ cont}} (x \in \text{List}(A)) \notin []}
\end{array}$$

Dimostrazione di correttezza di $\text{El}_{list}(y, x, (u, v, z).cons(z, v)) \in \text{List}(A) [x \in \text{List}(A), y \in \text{List}(A)]$

- $\text{El}_{list}(\text{nil}, x, (u, v, z).cons(z, v)) \rightarrow_1 x$ per $\beta_{list-red}$
- $\text{El}_{list}(cons(y, a), x, (u, v, z).cons(z, v)) \rightarrow_1 cons(\text{El}_{list}(y, x, (u, v, z).cons(z, v)))$
per $\beta_{2list-red} \Rightarrow$ per $y = \text{nil} \equiv cons(\text{El}_{list}(\text{nil}, x, (u, v, z).cons(z, v))) \rightarrow_1$
 $cons(x, \text{nil}) \in \text{List}(A) = x$ (dal punto precedente).

Capitolo 5

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 , $\text{List}(A)$). Le regole del tipo della somma disgiunta sono le seguenti.

5.1 Regole di Formazione

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

5.2 Regole di Introduzione

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

5.3 Regole di Eliminazione

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

5.4 Regole di Conservazione

$$\begin{array}{l} \text{C}_{1\text{-+}}) \frac{\begin{array}{c} M(z) \text{ type}[\Gamma, z \in B + C] \\ b \in B[\Gamma] \end{array} \quad e_B(x_1) \in M(\text{inl}(x_1))[\Gamma, x_1 \in B] \quad e_C(x_2) \in M(\text{inr}(x_2))[\Gamma, x_2 \in C]}{\text{El}_+(\text{inl}(b), e_B, e_C) = e_B(b) \in M(\text{inl}(b))[\Gamma]} \\ \\ \text{C}_{1\text{-+}}) \frac{\begin{array}{c} M(z) \text{ type}[\Gamma, z \in B + C] \\ c \in C[\Gamma] \end{array} \quad e_B(x_1) \in M(\text{inl}(x_1))[\Gamma, x_1 \in B] \quad e_C(x_2) \in M(\text{inr}(x_2))[\Gamma, x_2 \in C]}{\text{El}_+(\text{inr}(c), e_B, e_C) = e_C(c) \in M(\text{inr}(c))[\Gamma]} \end{array}$$

5.5 Regole di Uguaglianza

$$\text{eq-F-+)} \frac{B_1 = B_2 \in \text{type}[\Gamma] \quad C_1 = C_2 \in \text{type}[\Gamma]}{B_1 + C_1 = B_2 + C_2 \text{ type}(\Gamma)}$$

5.6 Eliminatore dipendente

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

$$E_{dip-+)} \frac{M(z) \text{ type}[\Gamma, z \in B + C] \quad e_B(x_1) \in M(\text{inl}(x_1))[\Gamma, x_1 \in B] \quad e_C(x_2) \in M(\text{inr}(x_2))[\Gamma, x_2 \in C]}{El_+(z, e_B, e_C) \in M(z)[\Gamma, z \in B + C]}$$

Con questi non si definisce solo che si ricorre su $B + C$, ma si dà anche un principio di induzione.

5.7 Semantica operativa della somma disgiunta

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

- $\beta_{1+}\text{-red}$) $El_+(\text{inl}(b), e_B, e_C) \rightarrow_1 e_B(b)$
- $\beta_{2+}\text{-red}$) $El_+(\text{inl}(b), e_B, e_C) \rightarrow_1 e_C(c)$
- $+\text{-red}$) $\frac{t_1 \rightarrow_1 t_2}{El_+(t_1, e_B, e_C) \rightarrow_1 El_+(t_2, e_B, e_C)}$
- Novità della somma disgiunta rispetto al tipo singoletto $+\text{-red}_I$) $\frac{b_1 \rightarrow_1 b_2}{\text{inl}(b_1) \rightarrow_1 \text{inl}(b_2)}$
- $+\text{-red}_{II}$) $\frac{c_1 \rightarrow_1 c_2}{\text{inr}(c_1) \rightarrow_1 \text{inr}(c_2)}$

5.8 Osservazioni dal punto di vista logico

Le regole di *Eliminazione/Introduzione* partono dalle regole di congiunzione nella logica (nel caso in cui sia B che C siano proposizioni), inoltre nel caso dell'eliminazione si va a eliminare verso altre proposizioni.

Tale concetto è stato voluto da *Martin-Löf*, per riuscire a interpretare la logica con la teoria dei tipi.

5.8.1 La regola di Formazione

Se si hanno le preposizioni $\beta \text{ prop}[\Gamma]$ e $\gamma \text{ prop}[\Gamma]$ e si interpreta la somma come disgiunzione, allora la regola di *Formazione* diventa

$$F-+)} \frac{\beta \text{ prop} [\Gamma] \quad \gamma \text{ prop} [\Gamma]}{B \vee C \text{ prop} [\Gamma]}$$

Questi ha influenza sia sulla regola ($E\text{-dip}+$) che ($I+$).

La regola di Eliminazione

$$E_{dipl+}) \frac{\xi \text{ prop}[\Gamma] \quad \xi \text{ è vero}[\Gamma, \text{supponiamo } \beta \text{ vero}] \quad \xi \text{ è vero}[\Gamma, \gamma \text{ è vero}]}{\cancel{El+}(z, e_B, e_C) \in \xi \text{ vero}[\Gamma, z \in \beta \vee \gamma \text{ vero}]}$$

che, nel calcolo dei sequenti, equivale alla or a sinistra

$$\vee\text{-S}) \frac{\beta \vdash_{\Gamma} \xi \quad \gamma \vdash_{\Gamma} \xi}{\beta \vee \gamma \vdash_{\Gamma} \xi}$$

Allo stesso modo le regole di Introduzione si possono vedere in modo molto semplice.

Le regole di Introduzione

$$I_{1-+}) \frac{b \in \beta[\Gamma] \quad \beta + \gamma \text{ prop}[\Gamma]}{\text{inl}(b) \in \beta \vee \gamma}$$

$b \in \beta \equiv \beta \text{ vero}$

$\beta \vee \gamma \equiv \beta \vee \gamma \text{ è vero}$

Questa, nel calcolo dei sequenti, equivale alla or a destra

Δ contesto, allora

$$\vee\text{-D}) \frac{\Delta \vdash_{\Gamma} \beta}{\Delta \vdash_{\Gamma} \beta \vee \gamma}$$

Lo stesso si può fare sulla seconda regola di Introduzione.

$$I_{2-+}) \frac{c \in \gamma[\Gamma] \quad \beta + \gamma \text{ prop}[\Gamma]}{\text{inr}(c) \in \beta \vee \gamma}$$

$c \in \gamma \equiv \gamma \text{ vero}$

$\beta \vee \gamma \equiv \beta \vee \gamma \text{ è vero}$

Questa, nel calcolo dei sequenti, equivale alla regola di riduzione naturale nella logica

Δ contesto, allora

$$I_{2-+}) \frac{\Delta \vdash_{\Gamma} \gamma}{\Delta \vdash_{\Gamma} \beta \vee \gamma}$$

5.9 Esercizi

1) Si scrivano le regole del tipo booleano come tipo semplice e si provi che è rappresentabile come $N_1 + N_2$.

Soluzione

Definisco il tipo Bool nel modo seguente $\text{Bool} = \text{true} \parallel \text{false}$

$\text{inl}(\ast) \equiv \text{true}$

$\text{inr}(\ast) \equiv \text{false}$

• **Regole del tipo Bool:**

– *Regole di Formazione*

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

– *Regole di Introduzione*

$$\text{I}_1\text{-Bool)} \frac{\Gamma \text{ cont}}{\text{true} \in \text{Bool}[\Gamma]} \quad \text{I}_2\text{-Bool)} \frac{\Gamma \text{ cont}}{\text{false} \in \text{Bool}[\Gamma]}$$

– *Regole di Eliminazione*

$$\text{E-Bool)} \frac{\text{M}(z) \text{ type}[\Gamma, z \in \text{Bool}] \quad t \in \text{Bool}[\Gamma] \quad e_B \in \text{M}(\text{true})[\Gamma] \quad e_C \in \text{M}(\text{false})[\Gamma]}{\text{El}_{\text{Bool}}(t, e_B, e_C) \in \text{M}(t)[\Gamma]}$$

– *Regole di Conversione*

$$\text{C}_1\text{-Bool)} \frac{\text{M}(z) \text{ type}[\Gamma, z \in \text{Bool}] \quad e_B \in \text{M}(\text{true})[\Gamma] \quad e_C \in \text{M}(\text{false})[\Gamma]}{\text{El}_{\text{Bool}}(\text{true}, e_B, e_C) = e_C \in \text{M}(\text{true})[\Gamma]}$$

$$\text{C}_1\text{-Bool)} \frac{\text{M}(z) \text{ type}[\Gamma, z \in \text{Bool}] \quad e_B \in \text{M}(\text{true})[\Gamma] \quad e_C \in \text{M}(\text{false})[\Gamma]}{\text{El}_{\text{Bool}}(\text{false}, e_B, e_C) = e_C \in \text{M}(\text{false})[\Gamma]}$$

– *Regole di Uguaglianza*

$$\text{eq}_1\text{-E-Bool)} \frac{\text{M}(z) \text{ type} [\Gamma, z \in \text{Bool}] \quad e_B = s \in \text{M}(\text{true})[\Gamma] \quad e_C = t \in \text{M}(\text{false})[\Gamma]}{\text{El}_{\text{Bool}}(\text{true}, e_B, e_C) = \text{El}_{\text{Bool}}(\text{true}, s, t) \in \text{M}(\text{true})[\Gamma]}$$

$$\text{eq}_2\text{-E-Bool)} \frac{\text{M}(z) \text{ type} [\Gamma, z \in \text{Bool}] \quad e_B = s \in \text{M}(\text{true})[\Gamma] \quad e_C = t \in \text{M}(\text{false})[\Gamma]}{\text{El}_{\text{Bool}}(\text{false}, e_B, e_C) = \text{El}_{\text{Bool}}(\text{false}, s, t) \in \text{M}(\text{false})[\Gamma]}$$

$$\text{eq}_1\text{-Bool)} \frac{t = t\lambda \in \text{Bool}[\Gamma] \quad e_B \in \text{M}(\text{true})[\Gamma] \quad e_C \in \text{M}(\text{false})[\Gamma]}{\text{M}(t) = \text{M}(t\lambda) \in \text{M}(\text{true})[\text{Gamma}]}$$

$$\text{eq}_2\text{-Bool)} \frac{t = t\lambda \in \text{Bool}[\Gamma] \quad e_B \in \text{M}(\text{true})[\Gamma] \quad e_C \in \text{M}(\text{false})[\Gamma]}{\text{M}(t) = \text{M}(t\lambda) \in \text{M}(\text{false})[\Gamma]}$$

• **Semantica operativa del tipo Bool:**

– $\beta_{1\text{Bool-red}}$ $\text{El}_{\text{list}}(\text{true}, e_B, e_C) \rightarrow_1 e_B(x)$

– $\beta_{2\text{Bool-red}}$ $\text{El}_{\text{list}}(\text{true}, e_B, e_C) \rightarrow_1 e_C(x)$

– IF-true) if true then M else N \rightarrow_1 M

– IF-false) if false then M else N \rightarrow_1 N

– Bool-red) $\frac{t_1 \rightarrow_1 t_2}{\text{El}_{\text{list}}(t_1, e_B, e_C) \rightarrow_1 \text{El}_{\text{list}}(t_2, e_B, e_C)}$

– IF) $\frac{M_1 \rightarrow_1 M_1'}{\text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rightarrow_1 \text{if } M_1' \text{ then } M_2 \text{ else } M_3}$

Capitolo 6

Tipo dell'uguaglianza proposizionale

Più modi permettono di descrivere il tipo dell'uguaglianza, uno di questi è il tipo dell'uguaglianza proposizionale (o identità proposizionale) di *Martin-Löf*. È un primo esempio di tipo dipendente, e se opportunamente abbinato al costruttore può costruire altri tipi dipendenti.

Questo è dipendente, in modo primitivo ed è definito dalle regole seguenti.

6.1 Regole di Formazione

$$\text{F-Id)} \frac{A \text{ type } [\Gamma] \quad a \in A[\Gamma] \quad b \in A[\Gamma]}{\text{Id}(A, a, b) \text{ type}[\Gamma]}$$

6.2 Regole di Introduzione

$$\text{I-Id)} \frac{a \in A[\Gamma]}{\text{id}(a) \in \text{Id}(A, a, a)[\Gamma]}$$

6.3 Regole di Eliminazione

$$\text{E-Id)} \frac{\begin{array}{c} M(z_1, z_2, z_3) \text{ type}[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)] \\ a \in A[\Gamma] \\ b \in A[\Gamma] \\ t \in \text{Id}(A, a, b)[\Gamma] \end{array} \quad e(x) \in M(x, x, \text{Id}(x))[\Gamma, x \in A]}{\text{El}_{Id}(t, (x).e(x)) \in M(a, b, t)[\Gamma]}$$

6.4 Regole di Conservazione

$$\text{E-Id)} \frac{\begin{array}{c} M(z_1, z_2, z_3) \text{ type}[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)] \\ a \in A[\Gamma] \end{array} \quad e(x) \in M(x, x, \text{Id}(x))[\Gamma, x \in A]}{\text{El}_{Id}(\text{Id}(a), (x).e(x)) = e(a) \in M(a, a, \text{Id}(a))[\Gamma]}$$

6.5 Regole di Uguaglianza

$$\text{eq-F-Id} \frac{A_1 = A_2 \text{ type}[\Gamma] \quad a_1 = a_2 \in A[\Gamma] \quad b_1 = b_2 \in A[\Gamma]}{\text{Id}(A_1, a_1, b_1) = \text{Id}(A_2, a_2, b_2) \text{ type}[\Gamma]}$$

$$\text{eq-I-Id} \frac{a_1 = a_2 \in A[\Gamma]}{\text{Id}(a_1) = \text{Id}(a_2) \in \text{Id}(A, a_1, a_1)[\Gamma]}$$

6.6 Eliminatore dipendente

L'eliminatore ha anche la forma dipendente.

$$\text{E-Id}_{\text{dip}} \frac{M(z_1, z_2, z_3) \text{ type}[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)] \quad e(x) \in M(x, x, \text{Id}(x))[\Gamma, x \in A]}{\text{El}_{\text{Id}}(z_3, (x).e(x)) \in M(z_1, z_2, z_3)[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]}$$

6.7 Lemma dell'Id

È derivabile la regola

$$\frac{a = b \in A[\Gamma]}{\text{Id}(a) \in \text{Id}(A, a, b)}$$

Con $\text{Id}(a) = \text{Id}(b)$

Ovvero l'uguaglianza proposizionale rende uguali i termini, definizionalmente, ma non vale il contrario. Percui se $p \in \text{Id}(A, a, b)[\Gamma]$ è derivabile $\nRightarrow a = b$ non è derivabile.

Formalmente si può trovare con il tipo singoletto, nel modo seguente: " $? \in \text{Id}(N_1, x, *) [x \in N_1] \nRightarrow x = * \in N_1 [x \in N_1]$ (non è vero che x è uguale a $*$)"

Dimostrazione

$$\begin{array}{c} \text{s-checks} \frac{\overline{a = b \in A[\Gamma]}}{a \in A[\Gamma]} \\ \text{I-Id} \frac{\overline{a \in A[\Gamma]}}{\text{Id}(a) \in \text{Id}(A, a, a)[\Gamma]} \quad \frac{\mathbf{1}}{\text{Id}(A, a, a) = \text{Id}(A, a, b) \text{ type}[\Gamma]} \\ \text{conv} \frac{\text{Id}(a) \in \text{Id}(A, a, a)[\Gamma] \quad \text{Id}(A, a, a) = \text{Id}(A, a, b) \text{ type}[\Gamma]}{\text{Id}(a) \in \text{Id}(A, a, b)[\Gamma]} \end{array}$$

1

$$\begin{array}{c} \text{s-checks} \frac{\overline{a = b \in [\Gamma]}}{a \in A[\Gamma]} \\ \text{s-checks} \frac{\overline{a \in A[\Gamma]}}{A \text{ type}[\Gamma]} \\ \text{ref} \frac{A = A \text{ type}[\Gamma]}{\text{Id}(A, a, a)[\Gamma] = \text{Id}(A, a, b)[\Gamma]} \quad \begin{array}{c} \text{s-checks} \frac{\overline{a = b \in [\Gamma]}}{a \in [\Gamma]} \\ \text{ref} \frac{a = a \in [\Gamma]}{a = a \in [\Gamma]} \end{array} \quad \frac{\overline{a = b \in [\Gamma]}}{a = b \in [\Gamma]} \end{array}$$

6.8 Esercizio di dimostrazione per induzione su addizione con zero

Nel dettaglio l'esercizio presentato in questa sezione, e svolto a lezione, viene risolto in §6.9, esercizio 5.

$\varphi(0) \ \& \ \forall_{x \in \text{Nat}} (\varphi(x) \rightarrow \varphi(\text{succ}(x))) \rightarrow \forall_{z \in \text{Nat}} \varphi(z)$ (§11).

Definizione Somma dei numeri Naturali $w + z \in \text{Nat} [w \in \text{Nat}, z \in \text{Nat}]$

$w + z \stackrel{\text{def}}{=} \text{El}_{\text{Nat}}(z, w, (x, y). \text{succ}(y))$

$w + 0 = w \in \text{Nat} [w \in \text{Nat}]$ per $\beta_{1\text{Nat-red}}$

ma definizionalmente $0 + z \neq z$

$\text{El}_{\text{Nat}}(z, 0, (x, y). \text{succ}(y))$ è in forma normale \neq da z , anch'essa in forma normale $\rightarrow 0 + z = z \in \text{Nat} [z \in \text{Nat}]$ non è derivabile in teoria dei tipi sui Naturali.

In realtà, si può risolvere $0 + z = z \in \text{Nat} [z \in \text{Nat}]$, non con l'uso dell'uguaglianza definizionale, ma con l'identità proposizionale, nel modo seguente.

Obiettivo: trovare $\text{pf} \in \forall_{z \in \text{Nat}} \text{Id}(\text{Nat}, 0+z, z)$

Sappiamo che

$\forall_{w \in \text{Nat}} \text{Id}(\text{Nat}, w+0, w) \equiv \lambda w. \text{id}(w) \ [\]$

Invece $\text{pf} \in \forall_{z \in \text{Nat}} \text{Id}(\text{Nat}, 0+z, z)$ va dimostrato per induzione.

Uso il *proof-term* così definito in una precedente lezione:

$h_2(z_1, z_2, z_3) \in \text{Id}(\text{Nat}, \text{succ}(z_1), \text{succ}(z_2)) [z_1 \in \text{Nat}, z_2 \in \text{Nat}, z_3 \in \text{Id}(\text{Nat}, z_2, z_3)]$

$$\text{E-Nat}_{\text{dip}} \frac{\frac{\text{Id}(\text{Nat}, 0+z, z) \text{ type} [z \in \text{Nat}]}{\text{Id}(0) \in \text{Id}(\text{Nat}, 0+0, 0) [\]} \quad \frac{h_2(0+x, x, y) \in \text{Id}(\text{Nat}, 0+\text{succ}(x), \text{succ}(x)) [x \in \text{Nat}, y \in \text{Id}(\text{Nat}, 0+x, x)]}{\text{I-}\Pi \frac{\text{El}_{\text{Nat}}(z, \text{id}(0), (x, y). h_2(0+x, x, y)) \text{Id}(\text{Nat}, 0+z, z) [z \in \text{Nat}]}{? \in \forall_{z \in \text{Nat}} \text{Id}(\text{Nat}, 0+z, z) [\]}}$$

In conclusione: $\forall_{z \in \text{Nat}} \text{Id}(\text{Nat}, 0+z, z)$

6.9 Esercizi

1) Si dimostri che esiste una funzione h_1 che mi permetta di dimostrare la simmetria dell'uguaglianza. Si dia la prova che $z_2 = z_1$ non appena $[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]$ con $A \text{ type}[\Gamma]$.

$h_1(z_1, z_2, z_3) \in \text{Id}(A, z_2, z_1) [\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]$

Soluzione (Si chiede di dimostrare che h_1 è *proof term* nella forma indicata sopra)

Assunzione dell'esercizio: $\text{Id}(A, z_1, z_2) \text{ type} [\Gamma, z_1 \in A, z_2 \in A]$

$M(z_1, z_2, z_3) \equiv \text{Id}(A, z_2, z_1)$

$h_1(z_1, z_2, z_3) \equiv \text{El}_{\text{Id}}(z_3, (x). \text{id}(x))$

$$\text{E-Id}_{dip} \frac{\frac{\mathbf{1}}{\text{Id}(A, z_2, z_1) \text{ type}[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]} \quad \frac{\mathbf{2}}{\text{id}(x) \in \text{Id}(A, x, x)[\Gamma, x \in A]}}{h_1(z_1, z_2, z_3) \in \text{Id}(A, z_2, z_1)[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]}$$

 $\mathbf{1}$

$$\text{F-id} \frac{\frac{\mathbf{1}_A}{A \text{ type} [\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]} \quad \frac{\mathbf{1}_B}{z_2 \in A[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]} \quad \frac{\mathbf{1}_C}{z_1 \in A[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]}}{\text{Id}(A, z_2, z_1) \text{ type}[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]}$$

 $\mathbf{1}_A$

$$\text{ind-ty} \frac{\frac{\mathbf{1}_A}{A \text{ type} [\Gamma]} \quad \text{F-Id} \frac{\frac{\mathbf{1}'_A}{A \text{ type}[\Gamma, z_1 \in A, z_2 \in A]} \quad \frac{\mathbf{1}'_B}{z_1 \in A[\Gamma, z_1 \in A, z_2 \in A]} \quad \frac{\mathbf{1}'_C}{z_2 \in A[\Gamma, z_1 \in A, z_2 \in A]}}{\text{Id}(A, z_1, z_2) \text{ type}[\Gamma, z_1 \in A, z_2 \in A]} \quad \frac{(z_3 \in \text{Id}(A, z_1, z_2)) \notin (\Gamma, z_1 \in A, z_2 \in A)}}{\text{F-c} \frac{\text{Id}(A, z_1, z_2) \text{ type}[\Gamma, z_1 \in A, z_2 \in A]}{\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2) \text{ cont}}} \quad \frac{(z_3 \in \text{Id}(A, z_1, z_2)) \notin (\Gamma, z_1 \in A, z_2 \in A)}}{A \text{ type} [\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]}$$

 $\mathbf{1}'_A$

$$\text{ind-ty} \frac{\frac{\overline{A \text{ type} [\Gamma]}}{\text{ind-ty} \frac{\overline{A \text{ type} [\Gamma]} \quad \text{F-c} \frac{\overline{A \text{ type} [\Gamma]}}{\Gamma, z_1 \in A \text{ cont}}} (z_1 \in A) \notin \Gamma}}{\text{F-c} \frac{\overline{A \text{ type} [\Gamma, z_1 \in A]}}{\Gamma, z_1 \in A, z_2 \in A \text{ cont}}} (z_2 \in A) \notin (\Gamma, z_1 \in A)} \quad \frac{\overline{A \text{ type} [\Gamma]}}{\text{ind-ty} \frac{\overline{A \text{ type} [\Gamma]} \quad \text{F-c} \frac{\overline{A \text{ type} [\Gamma]}}{\Gamma, z_1 \in A \text{ cont}}} (z_1 \in A) \notin \Gamma}}$$

 $\mathbf{1}'_B$

$$\text{ind-ty} \frac{\frac{\overline{A \text{ type} [\Gamma]} \quad \text{F-c} \frac{\overline{A \text{ type} [\Gamma]}}{\Gamma, z_1 \in A \text{ cont}}} (z_1 \in A) \notin \Gamma}{\text{F-c} \frac{\overline{A \text{ type} [\Gamma, z_1 \in A]}}{\Gamma, z_1 \in A, z_2 \in A \text{ cont}}} (z_2 \in A) \notin (\Gamma, z_1 \in A)} \quad \frac{\overline{A \text{ type} [\Gamma]}}{\text{ind-ty} \frac{\overline{A \text{ type} [\Gamma]} \quad \text{F-c} \frac{\overline{A \text{ type} [\Gamma]}}{\Gamma, z_1 \in A \text{ cont}}} (z_1 \in A) \notin \Gamma}}$$

 $\mathbf{1}'_C$

$$\text{ind-ty} \frac{\frac{\overline{A \text{ type} [\Gamma]} \quad \text{F-c} \frac{\overline{A \text{ type} [\Gamma]}}{\Gamma, z_1 \in A \text{ cont}}} (z_1 \in A) \notin \Gamma}{\text{F-c} \frac{\overline{A \text{ type} [\Gamma, z_1 \in A]}}{\Gamma, z_1 \in A, z_2 \in A \text{ cont}}} (z_2 \in A) \notin (\Gamma, z_1 \in A)} \quad \frac{\overline{A \text{ type} [\Gamma]}}{\text{ind-ty} \frac{\overline{A \text{ type} [\Gamma]} \quad \text{F-c} \frac{\overline{A \text{ type} [\Gamma]}}{\Gamma, z_1 \in A \text{ cont}}} (z_1 \in A) \notin \Gamma}}$$

 $\mathbf{2}$

$$\text{I-id} \frac{\frac{\text{F-c} \frac{\overline{A \text{ type} [\Gamma]}}{\Gamma, x \in A \text{ cont}}} (x \in A) \notin \Gamma}{\text{var} \frac{\overline{A \text{ type} [\Gamma]}}{x \in A[\Gamma, x \in A]}}}{\text{id}(x) \in \text{Id}(A, x, x)[\Gamma, x \in A]}$$

Le derivazioni $\mathbf{1}_B$ e $\mathbf{1}_C$ sono simili a quelle in $\mathbf{1}_A$. Le ometto per evitare una sovradimensionalità del numero di derivazioni.

Verifico che $\text{Id}(\mathbf{A}, z_1 \in \mathbf{A}, z_2 \in \mathbf{A})\text{type}[\Gamma, z_1 \in \mathbf{A}, z_2 \in \mathbf{A}]$ è una premessa valida rispetto al giudizio di conclusione $\text{Id}(\mathbf{A}, z_2 \in \mathbf{A}, z_1 \in \mathbf{A})\text{type}[\Gamma, z_1 \in \mathbf{A}, z_2 \in \mathbf{A}, z_3 \in \text{Id}(\mathbf{A}, z_1, z_2)]$

$$\begin{aligned}\Delta &= \Gamma, z_1 \in \mathbf{A}, z_2 \in \mathbf{A}, z_3 \in \text{Id}(\mathbf{A}, z_1, z_2) \\ \theta &= \mathbf{A}, z_1 \in \mathbf{A}, z_2 \in \mathbf{A}\end{aligned}$$

$$\text{sub-ty p} \frac{\frac{(*)}{z_2 \in \mathbf{A}[\Delta]} \quad \text{sub-ty p} \frac{\frac{(*)}{z_1 \in \mathbf{A}[\Delta, w \in \mathbf{A}]} \quad \text{sub-ty p} \frac{\frac{(*)}{w \in \mathbf{A}[\Delta, w \in \mathbf{A}]} \quad \text{ind-ty} \frac{\text{Id}(\theta)\text{type}[\Gamma, z_1 \in \mathbf{A}, z_2 \in \mathbf{A}]}{\text{Id}(\theta)\text{type}[\Delta, w \in \mathbf{A}]}}{\text{Id}(\mathbf{A}, w \in \mathbf{A}, z_2 \in \mathbf{A})\text{type}[\Delta, w \in \mathbf{A}]}}{\text{Id}(\mathbf{A}, w \in \mathbf{A}, z_1 \in \mathbf{A})\text{type}[\Delta, w \in \mathbf{A}]}}{\text{Id}(\mathbf{A}, z_2 \in \mathbf{A}, z_1 \in \mathbf{A})\text{type}[\Delta]} \quad \frac{(*)}{\Delta, w \in \mathbf{A} \text{ cont}}$$

Con il simbolo $(*)$ indico quelle derivazioni che terminano su un assioma, che per evitare ripetizioni nella derivazione ho omissso.

2) Si dimostri che esiste una funzione h_2 che mi permetta di dimostrare che il successore preserva l'uguaglianza proposizionale. Si dia la prova che $\text{succ}(z_1) = \text{succ}(z_2)$ non appena $[z_1 \in \text{Nat}, z_2 \in \text{Nat}, z_3 \in \text{Id}(\text{Nat}, z_1, z_2)]$ con $\text{Nat type}[\Gamma]$ non dipendente.

$$h_2(z_1, z_2, z_3) \in \text{Id}(\text{Nat}, \text{succ}(z_1), \text{succ}(z_2))[z_1 \in \text{Nat}, z_2 \in \text{Nat}, z_3 \in \text{Id}(\text{Nat}, z_1, z_2)]$$

Soluzione

$$\begin{aligned}M(z_1, z_2, z_3) &\equiv \text{Id}(\text{Nat}, \text{succ}(z_1), \text{succ}(z_2)) \\ h_2(z_1, z_2, z_3) &\equiv \text{El}_{Id}(z_3, (x). \text{id}(\text{succ}(x)))\end{aligned}$$

$$\Sigma \equiv z_1 \in \text{Nat}, z_2 \in \text{Nat}, z_3 \in \text{Id}(\text{Nat}, z_1, z_2)$$

$$\text{E-Id}_{dip} \frac{\frac{\mathbf{1}}{\text{Id}(\text{Nat}, \text{succ}(z_1), \text{succ}(z_2)) \in [\Sigma]} \quad \frac{\mathbf{2}}{\text{Id}(\text{succ}(x)) \in \text{Id}(\text{Nat}, \text{succ}(x), \text{succ}(x))[x \in \text{Nat}]}}{h_2(z_1, z_2, z_3) \in \text{Id}(\text{Nat}, \text{succ}(z_1), \text{succ}(z_2))[\Sigma]}$$

1

$$\text{F-id} \frac{\frac{\mathbf{1}_A}{\text{Nat type} [\Sigma]} \quad \frac{\mathbf{1}_B}{\text{succ}(z_1) \in \text{Nat}[\Sigma]} \quad \frac{\mathbf{1}_C}{\text{succ}(z_2) \in \text{Nat}[\Sigma]}}{\text{Id}(\text{Nat}, \text{succ}(z_1), \text{succ}(z_2)) \in [\Sigma]}$$

$\mathbf{1}_A$

$$\begin{array}{c}
\text{ind-ty} \frac{\text{Nat type } []}{\text{Nat type } [\Sigma]} \quad \text{F-Id} \frac{\frac{\mathbf{1}_A}{\text{Nat type}[z_1 \in \text{Nat}, z_2 \in \text{Nat}]} \quad \frac{\mathbf{1}_B}{z_1 \in \text{Nat}[z_1 \in \text{Nat}, z_2 \in \text{Nat}]} \quad \frac{\mathbf{1}_C}{z_2 \in \text{Nat}[z_1 \in \text{Nat}, z_2 \in \text{Nat}]}}{\text{F-c} \frac{\text{Id}(\text{Nat}, z_1, z_2) \text{ type}[z_1 \in \text{Nat}, z_2 \in \text{Nat}]}{\Sigma \text{ cont}}} \quad (z_3 \in \text{Id}(\text{Nat}, z_1, z_2)) \notin (z_1 \in \text{Nat}, z_2 \in \text{Nat}) \\
\\
\mathbf{1}_A \\
\\
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{ind-ty} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} (z_1 \in \text{Nat}) \notin []}{z_1 \in \text{Nat cont}}}{\text{F-c} \frac{\text{Nat type}[z_1 \in \text{Nat}]}{z_1 \in \text{Nat}, z_2 \in \text{Nat cont}}} (z_2 \in \text{Nat}) \notin z_1 \in \text{Nat} \\
\\
\mathbf{1}_B \\
\\
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{ind-ty} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} (z_2 \in \text{Nat}) \notin []}{z_2 \in \text{Nat cont}}}{\text{F-c} \frac{\text{Nat type}[z_2 \in \text{Nat}]}{z_2 \in \text{Nat}, z_1 \in \text{Nat cont}}} (z_1 \in \text{Nat}) \notin z_2 \in \text{Nat} \\
\\
\text{var} \frac{z_1 \in \text{Nat} [z_2 \in \text{Nat}, z_1 \in \text{Nat}]}{\text{ex-te} \frac{z_1 \in \text{Nat} [z_1 \in \text{Nat}, z_2 \in \text{Nat}]}{z_1 \in \text{Nat} [z_1 \in \text{Nat}, z_2 \in \text{Nat}]}} \quad \text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} (z_1 \in \text{Nat}) \notin []}{z_1 \in \text{Nat cont}} (z_2 \in \text{Nat}) \notin z_1 \in \text{Nat} \\
\\
\mathbf{1}_C \\
\\
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{ind-ty} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} (z_1 \in \text{Nat}) \notin []}{z_1 \in \text{Nat cont}}}{\text{F-c} \frac{\text{Nat type}[z_1 \in \text{Nat}]}{z_1 \in \text{Nat}, z_2 \in \text{Nat cont}}} (z_2 \in \text{Nat}) \notin z_1 \in \text{Nat} \\
\\
\text{var} \frac{z_2 \in \text{Nat}[z_1 \in \text{Nat}, z_2 \in \text{Nat}]}{\text{var} \frac{z_2 \in \text{Nat}[z_1 \in \text{Nat}, z_2 \in \text{Nat}]}{z_2 \in \text{Nat}[z_1 \in \text{Nat}, z_2 \in \text{Nat}]}}
\end{array}$$

2

$$\begin{array}{c}
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{F-c} \frac{\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} (x \in \text{Nat}) \notin []}{x \in \text{Nat cont}} \\
\\
\text{var} \frac{x \in \text{Nat}[x \in \text{Nat}]}{\text{I}_2\text{-Nat} \frac{\text{succ}(x) \in \text{Nat}[x \in \text{Nat}]}{\text{I-id} \frac{\text{Id}(\text{succ}(x)) \in \text{Id}(\text{Nat}, \text{succ}(x), \text{succ}(x)) [x \in \text{Nat}]}{x \in \text{Nat}}}}
\end{array}$$

Le derivazioni $\mathbf{1}_B$ e $\mathbf{1}_C$ sono simili a quelle in $\mathbf{1}_A$. Le ometto per evitare una sovradimensionalità del numero di derivazioni.

3) Si dimostri che se $a = b \in A[\Gamma]$ è derivabile nella teoria dei tipi, con le regole finora introdotte, allora esiste un *proof-term* tale che

$$\text{pf} \in \text{Id}(A, a, b)$$

è derivabile.

Soluzione (Ricalca la soluzione del lemma in §6.7)

Un **pf** è un qualsiasi elemento di un qualsiasi tipo di uguaglianza proposizionale. Per essere conforme con le regole fornite in §6 definisco $\text{pf} \equiv \text{id}(a)$.

$$\text{conv} \frac{\text{s-checks} \frac{\text{I-Id} \frac{\text{s-checks} \frac{\overline{a = b \in A[\Gamma]}}{a \in A[\Gamma]}}{\text{id}(a) \in \text{Id}(A, a, a)[\Gamma]}}{\text{id}(a) \in \text{Id}(A, a, b)[\Gamma]}}{\text{s-checks} \frac{\text{s-checks} \frac{\overline{a = b \in A[\Gamma]}}{a \in A[\Gamma]}}{\text{ref} \frac{A = A \text{ type}[\Gamma]}{A = A \text{ type}[\Gamma]}} \quad \text{s-checks} \frac{\text{s-checks} \frac{\overline{a = b \in A[\Gamma]}}{a \in A[\Gamma]}}{\text{ref} \frac{a = a \in A[\Gamma]}{a = a \in A[\Gamma]}} \quad \overline{a = b \in A[\Gamma]}}{\text{Id}(A, a, a) = \text{Id}(A, a, b) \text{ type}[\Gamma]}}{\text{id}(a) \in \text{Id}(A, a, b)[\Gamma]}$$

4) Si dimostri che esiste un *proof-term* **pf** del tipo

$$\text{pf} \in \text{Id}(N_1, x, *) [x \in N_1]$$

Soluzione

Uso l'eliminatore dipendente del tipo N_1

$$\begin{aligned} z &\equiv x \\ \text{El}_{N_1}(z, c) &\equiv \text{pf} \equiv \text{El}_{N_1}(x, x.\text{id}(*)) \\ M(z) &\equiv \text{Id}(N_1, x, *) \\ c \in M(*)[\Gamma] &\equiv \text{id}(*) \in \text{Id}(N_1, *, *) [x \in N_1] \end{aligned}$$

$$\text{F-Id} \frac{\overline{N_1 \text{ type}[x \in N_1]} \quad \overline{x \in N_1 [x \in N_1]} \quad \overline{* \in N_1 [x \in N_1]} \quad \text{I-Id} \frac{\text{I-S} \frac{[] \text{ cont}}{* \in N_1 []}}{\text{id}(*) \in \text{Id}(N_1, *, *) []}}{\text{E-S}_{\text{dip}} \frac{\text{Id}(N_1, x, *) \text{ type}[x \in N_1]}{\text{El}_{N_1}(x, x.\text{id}(*)) \in \text{Id}(N_1, x, *) [x \in N_1]}}$$

5) Si dimostri che esiste un *proof-term* **pf** tale che sia possibile definire l'addizione tra numeri naturali

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

in modo tale che esistano dei *proof-term* **pf**₁ e **pf**₂ tali che

$$\text{pf}_1 \in \text{Id}(\text{Nat}, x+0, x) [x \in \text{Nat}] \quad \text{pf}_2 \in \text{Id}(\text{Nat}, 0+x, x) [x \in \text{Nat}]$$

Soluzione

$$1. \text{ pf}_1 \in \text{Id}(\text{Nat}, x+0, x) [x \in \text{Nat}]$$

$$\forall x \in \text{Nat} \text{ Id}(\text{Nat}, x+0, x) [x \in \text{Nat}] \equiv \lambda x. \text{id}(x)$$

$$\text{Id}(\text{Nat}, x+0, x) [x \in \text{Nat}] \stackrel{\text{def}}{=} \prod_{x \in \text{Nat}} \text{Id}(\text{Nat}, x+0, x) [x \in \text{Nat}]$$

$$\begin{array}{c}
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type } []} \quad \text{F-c} \frac{x \in \text{Nat cont}}{x \in \text{Nat} [x \in \text{Nat}]} \quad (x \in \text{Nat}) \notin [] \quad \text{ind-ty} \frac{(*)}{\text{Nat type } []} \quad \frac{(*)}{x \in \text{Nat cont}} \\
\text{var} \frac{x \in \text{Nat} [x \in \text{Nat}]}{x \in \text{Nat} [x \in \text{Nat}]} \quad \text{ref} \frac{\text{Nat type } [x \in \text{Nat}]}{\text{Nat} = \text{Nat type } [x \in \text{Nat}]} \quad \frac{1}{x = x+0 \in \text{Nat} [x \in \text{Nat}]} \quad \text{ref} \frac{(*)}{x = x \in \text{Nat} [x \in \text{Nat}]} \\
\text{I-Id} \frac{\text{id}(x) \in \text{Id}(\text{Nat}, x, x) [x \in \text{Nat}]}{\text{id}(x) \in \text{Id}(\text{Nat}, x, x) [x \in \text{Nat}]} \quad \text{eq-F-Id} \frac{\text{Id}(\text{Nat}, x, x) = \text{Id}(\text{Nat}, x+0, x) \text{ type } [x \in \text{Nat}]}{\text{Id}(\text{Nat}, x, x) = \text{Id}(\text{Nat}, x+0, x) \text{ type } [x \in \text{Nat}]} \\
\text{conv} \frac{\text{id}(x) \in \text{Id}(\text{Nat}, x, x) [x \in \text{Nat}]}{\lambda x. \text{id}(x) \in \prod_{x \in \text{Nat}} \text{Id}(\text{Nat}, x+0, x) [x \in \text{Nat}]}
\end{array}$$

1

$$\begin{aligned}
\text{El}_{\text{Nat}}(0, c, e) &\equiv \text{El}_{\text{Nat}}(0, x, (w, z). \text{succ}(z)) \\
c \in \text{M}(0)[\Gamma] &\equiv x \in \text{Nat} [x \in \text{Nat}]
\end{aligned}$$

$$\begin{aligned}
\Delta &\equiv x \in \text{Nat}, w \in \text{Nat}, z \in \text{Nat} \\
\Delta' &\equiv z \in \text{Nat}, x \in \text{Nat}, w \in \text{Nat}
\end{aligned}$$

$$\begin{array}{c}
\text{I}_2\text{-Nat}_{\text{prog}} \frac{[] \text{ cont}}{\text{succ}(z) \in \text{Nat} [z \in \text{Nat}]} \quad \frac{\star}{\Delta' \text{ cont}} \quad \frac{\star}{\Delta} \\
\text{ind-te} \frac{\text{succ}(z) \in \text{Nat} [\Delta']}{\text{succ}(z) \in \text{Nat} [\Delta]} \\
\text{ex-te} \frac{\text{succ}(z) \in \text{Nat} [\Delta]}{\text{succ}(z) \in \text{Nat} [\Delta]} \\
\text{C}_1\text{-Nat} \frac{(*)}{\text{Nat type } [x \in \text{Nat}]} \quad \frac{(*)}{x \in \text{Nat} [x \in \text{Nat}]} \quad \text{sym} \frac{x+0 = x \in \text{Nat} [x \in \text{Nat}]}{x = x+0 \in \text{Nat} [x \in \text{Nat}]}
\end{array}$$

Ho usato $(*)$ per concludere le derivazioni già svolte all'interno dell'albero.

\heartsuit derivazione già risolta in §3.10, esercizio 3 (necessaria applicazione di $\alpha\text{-eq}$).

\star derivazione già risolte negli esercizi precedenti, Prevede una combinazione di istruzioni di indebolimento/assunzione di variabili/formazione di contesto per verificare l'assioma $[] \text{ cont}$.

2. $\text{pf}_2 \in \text{Id}(\text{Nat}, 0+x, x) [x \in \text{Nat}]$

Uso l'eliminatore dipendente del tipo Nat .

Svolgo la ricorsione su x (altrimenti su y cambierei il contesto e "uscirei" dalle richieste dell'esercizio), dunque

• Premesse:

- $\text{M}(z) \text{ type } [\Gamma, z \in \text{Nat}] \equiv \text{Id}(\text{Nat}, 0+x, x) [x \in \text{Nat}]$
- $c \in \text{M}(0)[\Gamma] \equiv \text{id}(0) \in \text{Id}(\text{Nat}, 0+0, 0) []$
- $\text{M}(z) \equiv \text{M}(x)$ per $\alpha\text{-eq}$ (dimostrazione in §3.10 esercizio 5)
- $e(x, y) \in \text{M}(\text{succ}(x)) [\Gamma, x \in \text{Nat}, y \in \text{M}(x)] \equiv h(0+x, x, y) \in \text{Id}(\text{Nat}, 0+\text{succ}(x), \text{succ}(x)) [\Gamma, x \in \text{Nat}, y \in \text{Id}(\text{Nat}, 0+x, x)]$

• Conclusioni:

- $\text{El}_{\text{Nat}}(z, c, e) \equiv \text{El}_{\text{Nat}}(x, \text{id}(0), (z, v). h(0+z, z, v))$
- $\text{M}(z) [\Gamma, z \in \text{Nat}] \equiv \text{Id}(\text{Nat}, 0+x, x) [x \in \text{Nat}]$

3'

$$\begin{array}{c}
\frac{\mathbf{3}_A}{\text{F-c} \frac{\text{Id}(\text{Nat}, 0+z, z) \text{ type}[z \in \text{Nat}]}{\text{F-Nat} \frac{\Delta \text{ cont}}{\text{Nat type}[\Delta]} \text{ ref } \frac{\text{Nat} = \text{Nat type}[\Delta]}{\text{eq-F-Id}}}} \quad (y \in \text{Id}(\text{Nat}, 0+z, z)) \quad \notin z \in \text{Nat} \quad \mathbf{3}_B \quad \frac{\star}{\text{I}_2\text{-Nat} \frac{z \in \text{Nat}[\Delta]}{\text{succ}(z) \in \text{Nat}[\Delta]} \text{ ref } \frac{\text{succ}(z) = \text{succ}(z) \in \text{Nat}[\Delta]}} \\
\hline
\text{Id}(\text{Nat}, \text{succ}(0+z), \text{succ}(z)) = \text{Id}(\text{Nat}, 0+\text{succ}(z), \text{succ}(z))[\Delta]
\end{array}$$

 $\mathbf{3}_A$

$$\begin{array}{c}
\text{F-Nat} \frac{[] \text{ cont}}{\text{Nat type}[]} \quad \text{F-c} \frac{z \in \text{Nat} \text{ cont}}{z \in \text{Nat} \text{ cont}} \quad (z \in \text{Nat}) \notin [] \quad \heartsuit \quad (*) \quad \text{var} \frac{z \in \text{Nat} \text{ cont}}{z \in \text{Nat}[z \in \text{Nat}]} \\
\text{F-Nat} \frac{\text{Nat type}[z \in \text{Nat}]}{\text{F-Id}} \quad \frac{0+z \in \text{Nat}[x \in \text{Nat}]}{\text{Id}(\text{Nat}, 0+z, z) \text{ type}[z \in \text{Nat}]}
\end{array}$$

 $\mathbf{3}_B$

$$\begin{array}{l}
\text{El}_{\text{Nat}}(0, c, e) \equiv \text{El}_{\text{Nat}}(0, \text{succ}(0+z), (w, u). \text{succ}(u)) \\
c \in \text{M}(0)[\Gamma] \equiv \text{succ}(0+z) \in \text{Nat}[\Delta]
\end{array}$$

$$\begin{array}{c}
\star \quad \text{ind-te} \frac{\heartsuit \quad \star}{\text{I}_2\text{-Nat} \frac{0+z \in \text{Nat}[z \in \text{Nat}]}{\text{succ}(0+z) \in \text{Nat}[\Delta]} \quad \Delta \text{ cont}} \quad \text{var} \frac{\star \quad \Delta, w \in \text{Nat}, u \in \text{Nat} \text{ cont}}{\text{I}_2\text{-Nat} \frac{u \in \text{Nat}[\Delta, w \in \text{Nat}, u \in \text{Nat}]}{\text{succ}(u) \in \text{Nat}[\Delta, w \in \text{Nat}, u \in \text{Nat}]}} \\
\text{C}_1\text{-Nat} \frac{\text{Nat type}[\Delta]}{\text{sym} \frac{0+\text{succ}(z) = \text{succ}(0+z) \in \text{Nat}[\Delta]}{\text{succ}(0+z) = 0+\text{succ}(z) \in \text{Nat}[\Delta]}}
\end{array}$$

Ho usato $(*)$ per concludere le derivazioni già svolte all'interno dell'albero.
 \heartsuit derivazione già risolta in §3.10, esercizio 3 (necessaria applicazione di $\alpha\text{-eq}$).

\diamond derivazione già risolta nell'esercizio 2, di questa sezione (necessaria applicazione di $\alpha\text{-eq}$).

\star derivazione già risolte negli esercizi precedenti, Prevede una combinazione di istruzioni di indebolimento/assunzione di variabili/formazione di contesto per verificare l'assioma $[] \text{ cont}$.

Capitolo 7

Tipo somma indicata forte

La somma indicata forte è il potenzialmento indicato della somma disgiunta binaria (§5). Tipo induttivo, ovvero generato con il principio d'induzione della regola di eliminazione, di tipi dipendente.

Definizione set-teorica

$$\begin{aligned} & \bigcup_{x \in B} C(x) \quad (C(x) \text{ set}) \quad x \in B \\ \neq & \bigcup \{y : \exists x \in B \quad y \in C(x)\} \\ & \stackrel{def}{=} \{(b,c) : b \in B \text{ e } c \in C(b)\} \end{aligned}$$

È l'unione disgiunta di una famiglia di insiemi, definito dalle regole seguenti.

7.1 Regole di Formazione

$$F\text{-}\Sigma) \frac{C(x) \text{ type } [\Gamma, x \in B]}{\sum_{x \in B} C(x) \text{ type } [\Gamma]}$$

7.2 Regole di Introduzione

$$I\text{-}\Sigma) \frac{k \in B[\Gamma] \quad c \in C(b)[\Gamma] \quad \sum_{x \in B} C(x) \text{ type } [\Gamma]}{\langle b, c \rangle \in \sum_{x \in B} C(x) \text{ type } [\Gamma]}$$

7.3 Regole di Eliminazione

$$E\text{-}\Sigma) \frac{M(z) \text{ type } [\Gamma, z \in \sum_{x \in B} C(x)] \quad t \in \sum_{x \in B} C(x)[\Gamma] \quad e(x,y) \in M(\langle x,y \rangle)[\Gamma, x \in B, y \in C(x)]}{El_{\Sigma}(t, (x,y).e(x,y)) \in M(t)[\Gamma]}$$

7.4 Regole di Conservazione

$$\text{C-}\Sigma) \frac{M(z) \text{ type}[\Gamma, z \in \sum_{x \in B} C(x)] \quad b \in B[\Gamma] \quad c \in C(b) \quad e(x,y) \in M(<x,y>)[\Gamma, x \in B, y \in C(x)]}{\text{El}_\Sigma(<b,c>, e) = e(b,c) \in M(<b,c>)[\Gamma]}$$

7.5 Regole di Uguaglianza

$$\text{eq-F-}\Sigma) \frac{B_1 = B_2 \text{ type}[\Gamma] \quad C_1(x) = C_2(x) \text{ type}[\Gamma, x \in B_1]}{\sum_{x \in B_1} C_1(x) = \sum_{x \in B_2} C_2(x) \text{ type}[\Gamma]}$$

$$\text{eq-I-}\Sigma) \frac{\sum_{x \in B} C(x) \text{ type}[\Gamma] \quad b_1 = b_2 \in B[\Gamma] \quad c_1 = c_2 \in C(b_1)[\Gamma]}{<b_1, c_1> = <b_2, c_2> \in \sum_{x \in B} C(x) \text{ type}[\Gamma]}$$

$$\text{eq-E-}\Sigma) \frac{M(z) \text{ type}[\Gamma, z \in \sum_{x \in B} C(x)] \quad t_1 = t_2 \in \sum_{x \in B} C(x)[\Gamma] \quad e_1(x,y) = e_2(x,y) \in M(<x,y>)[\Gamma, x \in B, y \in C(x)]}{\text{El}_\Sigma(t_1, e_1) = \text{El}_\Sigma(t_2, e_2) \in M(t_1)[\Gamma]}$$

7.6 Eliminatore dipendente

L'eliminatore ha anche la forma dipendente.

$$\text{E-}\Sigma_{\text{dip}}) \frac{M(z) \text{ type}[\Gamma, z \in \sum_{x \in B} C(x)] \quad e(x,y) \in M(<x,y>)[\Gamma, x \in B, y \in C(x)]}{\text{El}_\Sigma(z, (x,y).e(x,y)) \in M(z)[\Gamma, z \in \sum_{x \in B} C(x)]}$$

7.7 Semantica operativa della somma indicata forte

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

- $\beta_\Sigma\text{-red}$ $\text{El}_\Sigma(<b,c>, e) \rightarrow_1 e(b,c)$
- $\Sigma\text{-red}$ $\frac{t_1 \rightarrow_1 t_2}{\text{El}_\Sigma(t_1, e) \rightarrow_1 \text{El}_\Sigma(t_2, e)}$
- Novità della somma indicata forte rispetto al tipo singoletto $\Sigma\text{-red}_I) \frac{b_1 \rightarrow_1 b_2}{<b_1, c> \rightarrow_1 <b_2, c>}$
- $\Sigma\text{-red}_{II}) \frac{c_1 \rightarrow_1 c_2}{<b, c_1> \rightarrow_1 <b, c_2>}$

7.8 Esercizi

1) Provare a scrivere le regole del prodotto cartesiano $A \times B$ di un tipo A con un tipo B .

Svolgimento

$$A \times B \stackrel{def}{=} \sum_{x \in A} B$$

• **Regole del tipo prodotto cartesiano:**

– *Regole di Formazione*

$$F-x) \frac{A \text{ type } [\Gamma] \quad B \text{ type } [\Gamma]}{A \times B \text{ type } [\Gamma]}$$

– *Regole di Introduzione*

$$I-x) \frac{a \in A[\Gamma] \quad b \in B[\Gamma]}{\langle a, b \rangle \in A \times B[\Gamma]}$$

– *Regole di Proiezione*

$$PJ_1-x) \frac{d \in A \times B[\Gamma]}{\pi_1 d \in A[\Gamma]}$$

$$PJ_2-x) \frac{d \in A \times B[\Gamma]}{\pi_2 d \in B[\Gamma]}$$

– *Regole di Uguaglianza delle proiezioni*

$$PJ_1\text{-eq}) \frac{a \in A[\Gamma] \quad b \in B[\Gamma]}{\pi_1(\langle a, b \rangle) = a \in A[\Gamma]}$$

$$PJ_2\text{-eq}) \frac{a \in A[\Gamma] \quad b \in B[\Gamma]}{\pi_2(\langle a, b \rangle) = b \in B[\Gamma]}$$

• **Semantica operativa del tipo prodotto cartesiano:**

– $\beta_x\text{-red}_1) \text{El}_\Sigma(\langle a, b \rangle, e) \rightarrow_1 e(a, b)$

– $\beta_x\text{-red}_2) \pi_1(\langle a, b \rangle) \rightarrow_1 a$

– $\beta_x\text{-red}_3) \pi_2(\langle a, b \rangle) \rightarrow_1 b$

– $x\text{-red}) \frac{t_1 \rightarrow_1 t_2}{\text{El}_\Sigma(t_1, e) \rightarrow_1 \text{El}_\Sigma(t_2, e)}$

– $x\text{-red}_I) \frac{a_1 \rightarrow_1 a_2}{\langle a_1, b \rangle \rightarrow_1 \langle a_2, b \rangle}$

– $x\text{-red}_{II}) \frac{b_1 \rightarrow_1 b_2}{\langle a, b_1 \rangle \rightarrow_1 \langle a, b_2 \rangle}$

2) Provare a scrivere le regole del tipo delle funzioni $A \rightarrow B[\Gamma]$ da un tipo A a un tipo B .

Svolgimento

• **Regole del tipo delle funzioni:**

– *Regole di Formazione*

$$F \rightarrow \frac{A \text{ type } [\Gamma] \quad B \text{ type } [\Gamma]}{A \rightarrow B \text{ type } [\Gamma]}$$

– *Regole di Introduzione*

$$I \rightarrow \frac{b(x) \in B[\Gamma, x \in A]}{\lambda x^A. b(x) \in A \rightarrow B[\Gamma]}$$

– *Regole di Eliminazione*

$$E \rightarrow \frac{f \in A \rightarrow B[\Gamma] \quad a \in A[\Gamma]}{Ap(f, a) \in B[\Gamma]}$$

– *Regole di Conversione*

$$C \rightarrow \frac{b(x) \in B[\Gamma, x \in A] \quad a \in A[\Gamma]}{Ap(\lambda x^A. b(x), a) = b(a) \in B[\Gamma]}$$

– *Regole di Uguaglianza*

$$eq-I \rightarrow \frac{b_1(x) = b_2(x) \in B[\Gamma, x \in A]}{\lambda x^A. b_1(x) = \lambda x^A. b_2(x) \in A \rightarrow B[\Gamma]}$$

$$eq-E \rightarrow \frac{f_1 = f_2 \in A \rightarrow B[\Gamma] \quad a_1 = a_2 \in A[\Gamma]}{Ap(f_1, a_1) = Ap(f_2, a_2) \in B[\Gamma]}$$

• **Semantica operativa del tipo prodotto cartesiano:**

– $\beta_x\text{-red}$) $Ap(\lambda x^A. b(x), a) \rightarrow_1 b(a)$

– $\rightarrow\text{-red}_1$) $\frac{f_1 \rightarrow_1 f_2}{Ap(f_1, a) \rightarrow_1 Ap(f_2, a)}$

– $\rightarrow\text{-red}_2$) $\frac{a_1 \rightarrow_1 a_2}{Ap(f, a_1) \rightarrow_1 Ap(f, a_2)}$

– $\rightarrow\text{-red}$) $\frac{b_1 \rightarrow_1 b_2}{\lambda x^A. b_1 \rightarrow \lambda x^A. b_2}$

7.9 Rappresentazione del tipo prodotto cartesiano

Il **tipo somma indicata forte** è un potenziamento con tipi dipendenti del tipo prodotto cartesiano.

Allora ripartendo dal concetto di somma indicata forte

$$\bigcup_{x \in B \text{ set}} C(x) \quad (C(x) \text{ set}) \quad x \in B \stackrel{\text{def}}{=} \{(b,c): b \in B \text{ e } c \in C(b)\}$$

$$\text{Se } C(x) \equiv C^* \text{ dove } C^* \text{ non varia} \quad \Rightarrow \quad \stackrel{\text{def}}{=} \{(b,c): b \in B \text{ e } c \in C(b)\} \simeq B \times C^*$$

$$B \times C \stackrel{\text{def}}{=} \bigcup_{x \in B \text{ set}} C$$

Supposto che

$$\text{F-}\Sigma \frac{\text{B type}[\Gamma] \quad \text{ind-ty} \frac{C \text{ type}[\Gamma]}{C \text{ type}[\Gamma, x \in B]}}{B \times C \stackrel{\text{def}}{=} \sum_{x \in B} C}$$

Il tipo prodotto cartesiano è definito dalle seguenti regole.

7.9.1 Regole di Formazione

$$\text{F-x)} \frac{B \text{ type } [\Gamma] \quad C \text{ type } [\Gamma]}{B \times C \text{ type}[\Gamma]}$$

7.9.2 Regole di Introduzione

$$\text{I-x)} \frac{b \in B[\Gamma] \quad c \in C[\Gamma]}{\langle b,c \rangle \in B \times C[\Gamma]}$$

7.9.3 Regole di Proiezione

$$\text{PJ}_{1\text{-x)}} \frac{d \in B \times C[\Gamma]}{\pi_1 d \in B[\Gamma]}$$

$$\text{PJ}_{2\text{-x)}} \frac{d \in B \times C[\Gamma]}{\pi_2 d \in C[\Gamma]}$$

7.9.4 Regole di Uguaglianza delle proiezioni

$$\text{PJ}_{1\text{-eq)}} \frac{b \in B[\Gamma] \quad c \in C[\Gamma]}{\pi_1(\langle b,c \rangle) = b \in B[\Gamma]}$$

$$\text{PJ}_{2\text{-eq)}} \frac{b \in B[\Gamma] \quad c \in C[\Gamma]}{\pi_2(\langle b,c \rangle) = c \in C[\Gamma]}$$

7.9.5 Semantica operativa del prodotto cartesiano

- $\beta_x\text{-red}_1$) $\text{El}_\Sigma(<b,c>, e) \rightarrow_1 e(b,c)$
- $\beta_x\text{-red}_2$) $\pi_1(<b,c>) \rightarrow_1 b$
- $\beta_x\text{-red}_3$) $\pi_1(<b,c>) \rightarrow_1 c$
- $x\text{-red}$) $\frac{t_1 \rightarrow_1 t_2}{\text{El}_\Sigma(t_1, e) \rightarrow_1 \text{El}_\Sigma(t_2, e)}$
- $x\text{-red}_I$) $\frac{b_1 \rightarrow_1 b_2}{<b_1, c> \rightarrow_1 <b_2, c>}$
- $x\text{-red}_{II}$) $\frac{c_1 \rightarrow_1 c_2}{<b, c_1> \rightarrow_1 <b, c_2>}$

7.9.6 Correttezza del prodotto cartesiano

Dimostrazione

Dimostro che $B \times C \stackrel{\text{def}}{=} \bigcup_{x \in B \text{ set}} C$, trovando $PJ_1\text{-}x$ e $PJ_2\text{-}x$ usando $E\text{-}\Sigma_{dip}$.

$$\begin{aligned} M(z) &\equiv B \\ \sum_{x \in B} C &\equiv B \times C \end{aligned}$$

Assumo derivabili $B \text{ type}[\Gamma]$ e $C \text{ type}[\Gamma]$

1. Sulla prima proiezione

$$\text{El}_\Sigma(z, (x,y).x) \equiv \pi_1 z$$

La prima proiezione sulla coppia canonica $M(<x,y>)$ è x

$$\begin{array}{c} \text{ind-ty} \frac{\overline{B \text{ type}[\Gamma]} \quad \text{F-x} \frac{\overline{B \text{ type}[\Gamma]} \quad \overline{C \text{ type}[\Gamma]}}{\sum_{x \in B} C \text{ type}[\Gamma]} \quad \text{F-c} \frac{\overline{B \text{ type}[\Gamma]} \quad \overline{C \text{ type}[\Gamma]}}{\Gamma, z \in \sum_{x \in B} C \text{ cont}} \quad (z \in \sum_{x \in B} C) \notin \Gamma}{\text{E-}\Sigma_{dip} \frac{\overline{B \text{ type}[\Gamma]} \quad \overline{B \text{ type}[\Gamma, z \in \sum_{x \in B} C]}}{\pi_1 z \in B[\Gamma, z \in \sum_{x \in B} C]}} \\ \text{ind-ty} \frac{\overline{C \text{ type}[\Gamma]} \quad \text{F-c} \frac{\overline{B \text{ type}[\Gamma]} \quad \overline{\Gamma, x \in B \text{ cont}}}{\Gamma, x \in B, y \in C \text{ cont}} \quad (x \in B) \notin \Gamma}{\text{var} \frac{\overline{C \text{ type}[\Gamma, x \in B]} \quad \overline{\Gamma, x \in B, y \in C \text{ cont}}}{x \in B[\Gamma, x \in B, y \in C]}} \end{array}$$

In conclusione

$$\pi_1 z \stackrel{\text{def}}{=} \text{El}_\Sigma(z, (x,y).x) \in B[\Gamma, z \in \sum_{x \in B} C]$$

2. Sulla seconda proiezione

$$\text{El}_\Sigma(z, (x, y).y) \equiv \pi_2 z$$

La prima proiezione sulla coppia canonica $M(<x, y>)$ è y

$$\text{E-}\Sigma_{dip} \frac{(*)}{\frac{\text{B type}[\Gamma, z \in \sum_{x \in B} C]}{\pi_2 z \in C[\Gamma, z \in \sum_{x \in B} C]}} \quad \frac{\text{ind-ty} \frac{\frac{\text{C type}[\Gamma]}{\text{F-c } \frac{\overline{\text{B type}[\Gamma]}{\Gamma, x \in B \text{ cont}}} (x \in B) \notin \Gamma}}{\text{F-c } \frac{\text{C type}[\Gamma, x \in B]}{\Gamma, x \in B, y \in C \text{ cont}}} (y \in C) \notin (\Gamma, x \in B)} \quad \frac{\text{var} \frac{\text{C type}[\Gamma, x \in B]}{y \in C[\Gamma, x \in B, y \in C]}}{y \in C[\Gamma, x \in B, y \in C]}}$$

In conclusione

$$\pi_2 z \stackrel{\text{def}}{=} \text{El}_\Sigma(z, (x, y).y) \in C[\Gamma, z \in \sum_{x \in B} C]$$

Ho concluso con (*) i sequenti già risolti, per evitare ripetizioni.

Per verificare che effettivamente $\pi_1 z$ e $\pi_2 z$ siano $PJ_1\text{-}x$ e $PJ_2\text{-}x$, utilizzo le riduzioni $\beta_x\text{-red}_1$ e $\beta_x\text{-red}_2$.

La riduzione $\text{El}_\Sigma(<b, c>, e) \rightarrow_1 e(b, c)$ riscritta con sostituzioni diventa $\text{El}_\Sigma(<b, c>, e) \rightarrow_1 e[\frac{x}{b}, \frac{y}{c}]$

Allora applicata a una coppia $<b, c>$:

- $\pi_1(<b, c>) \stackrel{\text{def}}{=} \text{El}_\Sigma(z, (x, y).x) = \text{El}_\Sigma(<b, c>, (x, y).x) \rightarrow_1 x[\frac{x}{b}, \frac{y}{c}] \rightarrow b$
- $\pi_2(<b, c>) \stackrel{\text{def}}{=} \text{El}_\Sigma(z, (x, y).y) = \text{El}_\Sigma(<b, c>, (x, y).y) \rightarrow_1 y[\frac{x}{b}, \frac{y}{c}] \rightarrow c$

Entrambe le $\beta_x\text{-red}$ sono verificate. Percui la definizione $B \times C \stackrel{\text{def}}{=} \sum_{x \in B} C$ è giustificata.

7.9.7 Esempi

L'unione disgiunta non è l'unione insiemistica e questo lo si vede chiaramente dalla relazione sotto

$$\bigcup_{x \in \text{Nat}} \text{Nat} \equiv \{(x, y) \bullet x \in \text{Nat}, y \in \text{Nat}\} \equiv \text{Nat} \times \text{Nat} \neq \bigcup_{x \in \text{Nat}} \text{Nat} \{y: y \in \text{Nat}, \text{ per un certo } x \in \text{Nat}\}$$

La prima parte della relazione, se la famiglia è costante ingloba il prodotto cartesiano.

Un esempio di unioni genuine, che si possono formare con tipi che non coincidono con il prodotto cartesiano, è $\sum_{x \in \text{Nat}} \text{Mat}(x \bullet x)$, ove Mat sono le matrici quadratiche.

A livello insiemistico potremmo dover proiettare, data una coppia $\langle x, k \rangle$ (con x elementi e k una certa matrice), sull'indice o sull'elemento di cui stiamo parlando. Dunque dovremmo poter avere a disposizione le due proiezioni PJ_1-x e PJ_2-x anche su famiglie non costanti.

Per individuare correttamente $\pi_1(z)$ e $\pi_2(z)$ assumo che $\sum_{x \in B} C(x) \text{ type}[\Gamma]$ sia derivabile. Tale ipotesi è vincolante per l'ottenimento della famiglia.

Definisco $\omega = z \in \sum_{x \in B} C(x)$

1. Per la prima proiezione

$$\text{ind-ty} \frac{\frac{\text{B type}[\Gamma]}{\text{E-}\Sigma_{dip} \frac{\text{B type}[\Gamma, \omega]}{\text{El}_{\Sigma}(z, (x, y).x) \in B[\Gamma, \omega]}} \quad \text{F-c} \frac{\frac{\sum_{x \in B} C(x) \text{ type}[\Gamma]}{\Gamma, \omega \text{ cont}} \quad \omega \notin \Gamma \quad \text{F-c} \frac{\frac{C(x) \text{ type}[\Gamma, x \in B]}{\Gamma, x \in B, y \in C(x) \text{ cont}} (y \in C(x)) \notin (\Gamma, x \in B)}{\text{var} \frac{x \in B[\Gamma, x \in B, y \in C(x)]}{\text{El}_{\Sigma}(z, (x, y).x) \in B[\Gamma, \omega]}}}{\text{El}_{\Sigma}(z, (x, y).x) \in B[\Gamma, \omega]}$$

In conclusione

$$\pi_1 z \stackrel{\text{def}}{=} \text{El}_{\Sigma}(z, (x, y).x) \in B[\omega]$$

2. Per la seconda proiezione

$$\pi_2 z \stackrel{\text{def}}{=} \text{El}_{\Sigma}(z, (x, y).y) \in C(\pi_1(z))[\omega]$$

Allora usando che $M(z) \equiv C(\pi_1 z) \Rightarrow M(\langle x, y \rangle) \equiv C(\pi_1 \langle x, y \rangle)$

$$\text{E-}\Sigma_{dip} \frac{\frac{\mathbf{1}}{C(\pi_1 z) \text{ type}[\Gamma, \omega]} \quad \frac{\mathbf{2}}{y \in C(\pi_1 \langle x, y \rangle)[\Gamma, x \in B, y \in C(x)]}}{\text{El}_{\Sigma}(z, (x, y).y) \in C(\pi_1 z)[\Gamma, \omega]}$$

1

$$\text{sub-typ} \frac{\text{E-}\Sigma_{dip} \frac{(*)}{\pi_1 z \in B[\Gamma, \omega]} \quad \text{ind-ty} \frac{\frac{C(x) \text{ type}[\Gamma, x \in B]}{\text{ex-ty} \frac{C(x) \text{ type}[\Gamma, x \in B, \omega]}{C(x) \text{ type}[\Gamma, \omega, x \in B]}} \quad \frac{(*)}{\Gamma, x \in B, \omega \text{ cont}} \quad \text{ind-ty} \frac{\frac{\text{B type}[\Gamma]}{\text{F-c} \frac{\text{B type}[\Gamma, \omega]}{\Gamma, \omega, x \in B \text{ cont}} (x \in B) \notin (\Gamma, \omega)}}{\text{F-c} \frac{\sum_{x \in B} C(x) \text{ type}[\Gamma]}{\Gamma, \omega \text{ cont}} \quad \omega \notin \Gamma}}{C(\pi_1 z) \text{ type}[\Gamma, \omega]}$$

2

$$\text{conv} \frac{\text{var} \frac{(*)}{\Gamma, x \in B, y \in C(x) \text{ cont}} \quad \text{sub-eq-typ} \frac{\frac{\mathbf{2}^A}{x = \pi_1 \langle x, y \rangle \in B[\Gamma, x \in B, y \in C(x)]} \quad \frac{\mathbf{2}^B}{C(w) \text{ type}[\Gamma, x \in B, y \in C(x), w \in B]}}{y \in C(\pi_1 \langle x, y \rangle)[\Gamma, x \in B, y \in C(x)]}$$

2^A

$$C-\Sigma \frac{\frac{(*)}{B \text{ type}[\Gamma]} \quad \frac{(*)}{x \in B[\Gamma]} \quad \frac{(*)}{y \in C(x)} \quad \frac{(*)}{x \in B[\Gamma, x \in B, y \in C(x)]} \quad \frac{(*)}{\Gamma, x \in B, y \in C(x) \text{ cont}}}{\text{ind-ty} \frac{\pi_1 \langle x, y \rangle = x \in B[\Gamma]}{\text{sym} \frac{\pi_1 \langle x, y \rangle = x \in B[\Gamma, x \in B, y \in C(x)]}{x = \pi_1 \langle x, y \rangle \in B[\Gamma, x \in B, y \in C(x)]}}}$$

 2^B

$$\text{ind-ty} \frac{\frac{C(x) \text{ type}[\Gamma, x \in B]}{\alpha\text{-eq} \frac{C(x) \text{ type}[\Gamma, x \in B]}{C(w) \text{ type}[\Gamma, w \in B]}} \quad \frac{(*)}{\Gamma, x \in B \text{ cont}}}{\text{ind-ty} \frac{C(w) \text{ type}[\Gamma, w \in B, x \in B, y \in C(x)]}{\text{ex-ty} \frac{C(w) \text{ type}[\Gamma, w \in B, x \in B, y \in C(x)]}{C(w) \text{ type}[\Gamma, x \in B, y \in C(x), w \in B]}}} \quad \frac{(*)}{\Gamma, w \in B, x \in B, y \in C(x) \text{ cont}} \quad \frac{(*)}{\Gamma, x \in B, y \in C(x), w \in B \text{ cont}}$$

In conclusione vale $\pi_2 z$

Ho concluso con (*) una derivazione per semplificare o evitare ripetizioni nella derivazione stessa.

7.10 Usi del tipo somma indicata per rappresentare l'assioma di separazione e di quantificazione universale

Il tipo $\sum_{x \in B} C(x)$ viene usato per rappresentare:

1. uso *set* teoretico: l'unione indicata disgiunta insiemistica ($C(x) \text{ set}[x \in B]$)
2. uso *set* teoretico: con assioma di separazione
3. uso logico: proposizionale

Il punto (1) l'ho già spiegato nella prima parte di questo capitolo, ora presento le implicazioni di (2) e (3).

7.10.1 L'assioma di separazione

L'assioma di separazione dice che dato B , insieme, esiste l'insieme ottenuto per separazione da B tramite $\varphi(x)$ definito come $\{x \in B \mid \varphi(x)\}$ con $\varphi(x)$ predicato. Perciò per ogni y insieme, $y \& \{x \in B \mid \varphi(x)\} \Leftrightarrow \varphi(y)$ vale.

Si può, dunque, simulare l'insieme degli y per cui vale $\varphi(y)$ nel momento in cui si ha una proposizione

$$\text{F-}\Sigma \frac{\varphi(x) \text{ prop } \text{type}[\Gamma, x \in B]}{\sum_{x \in B} \varphi(x) \text{ type}_{\text{set}}[\Gamma]}$$

dove, nella regola sopra, prop viene pensato come un type

Dimostrazione

- (\Leftarrow) Da $\sum_{x \in B} \varphi(x)$ significa avere $b \in B + \text{pf} \in \varphi(b)$, e quindi b soddisfa $\varphi(x)$. Ecco che non è solo b ma $\langle b, \text{pf} \rangle \in \{x \in B \mid \varphi(x)\}$ con $\text{pf} \in \varphi(b)$.
 $I-\Sigma \Leftarrow \varphi(b)$.
- (\Rightarrow) $z \in \{x \in B \mid \varphi(x)\}$ con z che sarebbe una coppia e $x \in B \equiv \sum_{x \in B} \varphi(x) \Rightarrow \varphi(z)$ vale, in quanto z è una coppia tipata.
 Ma posso tuttavia dimostrare che $\varphi(\pi_1 z)$ vale usando le proiezioni. Ecco che $\pi_2(z) \in \varphi(\pi_1 z)$. $\pi_2(z)$ è così *proof term* per cui $\varphi(\pi_1 z)$ vale.

Questo ha permesso a *Martin-Löf* di inglobare la *set theory* all'interno della teoria dei tipi.

7.10.2 Proporzionale

$$\Sigma\text{-F} \frac{C(x) \text{ type}[\Gamma, x \in B_{\text{set}}] \equiv \varphi(x) \text{ prop}}{\sum_{x \in B} \varphi(x) \text{ type}_{\text{prop}}[\Gamma]} \stackrel{\text{def}}{=} \exists_{x \in B} \varphi(x)$$

È la quantificazione esistenziale, in logica, di φ che varia sull'insieme B .

Giustificazione

$\phi \text{ prop}[\Gamma, x_1 \in \varphi_1, \dots, x_n \in \varphi_n]$

Sotto ipotesi $\varphi_i \text{ prop}[\Gamma] \quad i = 1..n$

e $\varphi \text{ prop}[\Gamma]$ con $\text{prop} \equiv \text{type}$ (ovvero tipi/*set* delle loro dimostrazioni)

Definizione proof-term

$t \in \varphi[\Gamma]$ dove φ è prop

Se devo dire che $\text{pf} \in \phi[\Gamma, x_1 \in \varphi_1, \dots, x_n \in \varphi_n] \equiv \phi$ è vero $[\Gamma, \varphi \text{ vero}, \dots, \varphi_n \text{ vero}]$ (nessuno degli elementi del contesto dipende dal precedente) $\equiv \varphi_1, \dots, \varphi_n \vdash_{\Gamma} \phi$

Sappiamo che la logica lavora con variabili *untyped*, dunque noi ora aggiungiamo in $\exists_{x \in B} \varphi(x)$ e $\forall_{x \in B} \varphi(x)$ l'informazione sul tipo.

Per quanto assunto sopra, la regola di Introduzione viene nel modo seguente riscritta

Assumendo che $\exists_{x \in B} \varphi(x) \text{ type}[\Gamma]$ sia ben formata

7.10. USIDEL TIPO SOMMA INDICIATA PER RAPPRESENTARE L'ASSIOMA DI SEPARAZIONE E DI QUANTIFICAZIONE

$$\Sigma\text{-I} \frac{b \in B_{set}[\Gamma] \quad c \in \varphi(b)[\Gamma]}{\leq b, c \succ \in \sum_{x \in B} C(x)[\Gamma]}$$

$$c \in \varphi(b)[\Gamma] \equiv \varphi(b) \text{ è vero } [\Gamma, x_1 \in \gamma_1, \dots, x_n \in \gamma_n] \equiv \gamma_1, \dots, \gamma_n \vdash \varphi(b)$$

$$\sum_{x \in B} C(x)[\Gamma] \equiv \exists x \in B \varphi(x) \text{ vero}[\Gamma]$$

In conclusione, riscritta, la regola di Introduzione, con il calcolo dei sequenti diventa

$$\Sigma\text{-I} \frac{\gamma_1, \dots, \gamma_n \vdash \varphi(b)}{\gamma_1, \dots, \gamma_n \vdash \exists x \in B \varphi(x)}$$

Che non è altro che la regola a destra dell'esiste.

Ora definisco la regola dell'Eliminatore dipendente.

Per farlo suppongo di avere un'altra proposizione definita come segue

$$\frac{\phi \text{ prop}[\Gamma]}{\phi \text{ prop}[\Gamma, z \in \sum_{x \in B} \varphi(x)]}$$

$$E\text{-}\Sigma_{dip} \frac{\phi \text{ prop}[\Gamma] \quad e(x, y) \in \phi[\Gamma, x \in B, y \in \varphi(x)]}{\gamma \in \phi[\Gamma, z \in \exists x \in B \varphi(x)]}$$

$$e(x, y) \equiv \varphi \text{ vero}$$

$$\phi[\Gamma, x \in B, y \in \varphi(x)] \equiv \phi \text{ è vero}[\Gamma, x \in B, \varphi(x) \text{ vero}]$$

$$\gamma \in \phi \equiv \phi \text{ vera}$$

$$z \in \exists x \in B \varphi(x) \equiv \exists x \in B \varphi(x) \text{ vero}$$

In conclusione, riscritta, la regola dell'Eliminatore dipendente, con il calcolo dei sequenti diventa

$$E\text{-}\Sigma_{dip} \frac{\gamma_1, \dots, \gamma_n \varphi(x) \text{ vero } \vdash_{\Gamma, x \in B} \phi}{\gamma_1, \dots, \gamma_n, \exists x \in B \varphi(x) \vdash_{\Gamma} \phi}$$

Che non è altro che la regola a sinistra dell'esiste.

7.10.3 Conclusioni sugli usi del tipo della somma indicata disgiunta

Per

1. $\sum_{x \in B} C(x)$ è un *set* (dove sia $x \in B$ che $C(x)$ sono rispettivamente dei *set*)
 $\Rightarrow B \times C$ è il prodotto cartesiano
2. $\sum_{x \in B} \varphi(x)$ dove φ è una proposizione \Rightarrow rappresenta l'assioma di separazione
 $\{x \in B \mid \varphi(x)\}$
Esempio: $\sum_{z \in list(Nat)} Id(Nat, lh(z), 2) \Rightarrow$ rappresenta il sottoinsieme delle
liste di lunghezza 2

3. $\exists x \in B \varphi(x) \Rightarrow \sum_{x \in B} \varphi(x)$, ove $\varphi(x)$ e $\sum_{x \in B} \varphi(x)$ sono proposizioni

Questo serve per:

- Capire in che modo formalizzare
- Comprendere il significato della teoria dei tipi che sto leggendo

7.11 Congiunzione come tipo prodotto cartesiano

Il tipo prodotto cartesiano permette di interpretare la congiunzione della logica.

Date due proposizioni $\varphi \text{ prop}[\Gamma]$ e $\psi \text{ prop}[\Gamma]$

$$\varphi \ \& \ \psi \stackrel{\text{def}}{=} \varphi \times \psi$$

L'uguaglianza vale in quanto $\varphi \text{ prop}[\Gamma] \equiv \varphi \text{ type}[\Gamma]$ che è il tipo delle sue dimostrazioni

La regola di *Introduzione* del prodotto cartesiano è la seguente

$$\text{I-x} \frac{a \in \varphi[\Gamma] \quad b \in \psi[\Gamma]}{< a, b > \in \varphi \times \psi [\Gamma]}$$

$$\Rightarrow a \in \varphi[\Gamma] \equiv \varphi \text{ true}[\Gamma]$$

$$\Rightarrow b \in \psi[\Gamma] \equiv \psi \text{ true}[\Gamma]$$

$$\Rightarrow \varphi \times \psi \equiv \varphi \ \& \ \psi$$

$$\Rightarrow \varphi \ \& \ \psi \text{ vero}[\Gamma]$$

$$(1) \text{ Dunque } \varphi \text{ true}[\Gamma] \text{ e } \psi \text{ true}[\Gamma] (\Rightarrow) \varphi \ \& \ \psi \text{ true}[\Gamma]$$

Con la regola di *Eliminazione* riesco a provare anche il verso opposto dell'implicazione. In quanto $d \in \varphi \times \psi[\Gamma]$ è come dire $\varphi \ \& \ \psi \text{ true}[\Gamma]$, per cui, grazie alle regole di Proiezione del prodotto cartesiano

$$\text{PJ}_1\text{-x} \frac{d \in \varphi \ \& \ \psi[\Gamma]}{\pi_1 d \in \varphi [\Gamma]}$$

$$d \in \varphi \ \& \ \psi[\Gamma] \equiv \varphi \ \& \ \psi \text{ true}[\Gamma]$$

$$\pi_1 d \in \varphi [\Gamma] \equiv \varphi \text{ vero}[\Gamma]$$

$$\text{PJ}_2\text{-x} \frac{d \in \varphi \ \& \ \psi[\Gamma]}{\pi_2 d \in \psi [\Gamma]}$$

$$d \in \varphi \ \& \ \psi[\Gamma] \equiv \varphi \ \& \ \psi \text{ true}[\Gamma]$$

$$\pi_2 d \in \psi [\Gamma] \equiv \psi \text{ vero}[\Gamma]$$

(2) Dunque con le regole di Eliminazione si è mostrato come se $\varphi \ \& \ \psi \text{ true}[\Gamma] \Rightarrow \varphi \text{ true}[\Gamma]$ e $\psi \text{ true}[\Gamma]$.

Quanto definito in (1) e (2) permettono di concludere che date $\varphi \text{ prop}[\Gamma]$ e

$\psi \text{ prop}[\Gamma]$ (interpretate come tipi delle loro dimostrazioni), è corretto definire $\varphi \& \psi \stackrel{\text{def}}{=} \varphi \times \psi$.

Questo perchè una formula è vera quando $\varphi \text{ true}[\Gamma]$ sse $(\text{def}) \exists \text{ pf} \in \varphi[\Gamma]$. Tale definizione di *proof-term* rende vera entrambe le proprietà (1) e (2), e quindi si può ben interpretare la congiunzione come prodotto cartesiano.

In conclusione $\varphi \& \psi \stackrel{\text{def}}{=} \varphi \times \psi \equiv \sum_{x \in \varphi} \psi$.

Nella teoria dei tipi semplici $\varphi \& \psi \stackrel{\text{def}}{=} \varphi \times \psi$ era già stato introdotto da *Curry-Howard*.

Capitolo 8

Tipo delle funzioni

Il tipo delle funzione è definito dalle regole seguenti.

8.1 Regole di Formazione

$$F \rightarrow) \frac{B \text{ type } [\Gamma] \quad C \text{ type } [\Gamma]}{B \rightarrow C \text{ type } [\Gamma]}$$

8.2 Regole di Introduzione

$$I \rightarrow) \frac{c(x) \in C[\Gamma, x \in B]}{\lambda x^B. c(x) \in B \rightarrow C[\Gamma]}$$

c è una meta-variabile, con il quale si indica un termine che può dipendere o meno da x.

8.3 Regole di Eliminazione

$$E \rightarrow) \frac{f \in B \rightarrow C[\Gamma] \quad b \in B[\Gamma]}{Ap(f, b) \in C[\Gamma]}$$

Il tipo non è induttivo, per cui la regola di eliminazione non definisce un ricorsivo, nè principi di induzione.

8.4 Regole di Conservazione

$$C \rightarrow) \frac{c(x) \in C[\Gamma, x \in B] \quad b \in B[\Gamma]}{Ap(\lambda x^B. c(x), b) = c(b) \in C[\Gamma]}$$

8.5 Regole di Uguaglianza

$$\text{eq-E} \rightarrow \frac{f_1 = f_2 \in B \rightarrow C[\Gamma] \quad b_1 = b_2 \in B[\Gamma]}{\text{Ap}(f_1, b_1) = \text{Ap}(f_2, b_2) \in C[\Gamma]}$$

$$\text{eq-I} \rightarrow \frac{c_1(x) = c_2(x) \in C[\Gamma, x \in B]}{\lambda x^B. c_1(x) = \lambda x^B. c_2(x) \in B \rightarrow C[\Gamma]}$$

La regola (*eq-I*→) prende anche il nome di *ξ-rule* ed è una regola difficile da modellare.

8.6 Semantica operativa del tipo funzione

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

- β_{\rightarrow} -red) $\text{Ap}(\lambda x. c(x), b) \rightarrow_1 c(b)$
Per la programmazione: $\text{Ap}(\lambda x. c, b) \rightarrow_1 c[\frac{x}{b}]$
- \rightarrow -red₁) $\frac{f_1 \rightarrow_1 f_2}{\text{Ap}(f_1, b) \rightarrow_1 \text{Ap}(f_2, b)}$
- \rightarrow -red₂) $\frac{b_1 \rightarrow_1 b_2}{\text{Ap}(f, b_1) \rightarrow_1 \text{Ap}(f, b_2)}$
- Novità della somma indicata forte rispetto al tipo singoletto \rightarrow -red) $\frac{c_1 \rightarrow_1 c_2}{\lambda x. c_1 \rightarrow_1 \lambda x. c_2}$

8.7 Osservazioni dal punto di vista logico

8.7.1 La regola di Introduzione

Questo tipo è importante perchè c'è di mezzo la logica. Se difatti prendo la regola di *Introduzione* e ci metto le preposizioni

$$\text{I} \rightarrow \frac{\beta \text{ prop}[\Gamma] \quad \gamma \text{ prop}[\Gamma]}{\beta \rightarrow \gamma \text{ prop}[\Gamma]}$$

Ove $\beta \rightarrow \gamma$ è un'implicazione logica.

$$\beta \rightarrow \gamma \text{ (implicazione)} = \beta \rightarrow \gamma \text{ (tipo)}$$

Questo risulta vero perchè (*I*→) afferma che se si ha

$$\text{I} \rightarrow \frac{c(x) \in \gamma[\Gamma, \Delta, x \in \beta]}{\lambda x. c(x) \in \beta \rightarrow \gamma[\Gamma, \Delta]}$$

$$c(x) \in \gamma \equiv \gamma \text{ vero}$$

$$x \in \beta \equiv \beta \text{ vero}$$

$$\beta \rightarrow \gamma \equiv \beta \rightarrow \gamma \text{ vero}$$

Che non è altro che la regola, nel calcolo dei sequenti, dell'implica a destra

$$\rightarrow\text{-D) } \frac{\varphi_1, \dots, \varphi_n, \beta \vdash_{\Gamma} \gamma}{\varphi_1, \dots, \varphi_n \vdash_{\Gamma} \beta \rightarrow \gamma}$$

8.7.2 La regola di Eliminazione (Modus Ponens)

$$E_{\rightarrow}) \frac{f \in \beta \rightarrow \gamma[\Gamma] \quad b \in \beta[\Gamma]}{? \in \gamma[\Gamma]}$$

$f \in \beta \rightarrow \gamma \equiv \beta \rightarrow \gamma$ vero

$b \in \beta \equiv \beta$ vero

$? \in \gamma \equiv \gamma$ vero

Che non è altro che la regola, nel calcolo dei sequenti, del Modus Ponens

$$MP) \frac{\gamma_1, \dots, \gamma_n \vdash_{\Gamma} \beta \rightarrow \gamma \quad \gamma_1, \dots, \gamma_n \vdash_{\Gamma} \beta}{\gamma_1, \dots, \gamma_n \vdash_{\Gamma} \gamma}$$

8.8 Tipo prodotto dipendente

Il tipo prodotto dipendente o indicato (si chiama anche \prod), come già riscontrato per il tipo funzione, è un tipo non induttivo. Inoltre è un potenziamento espressivo, sempre, del tipo dello spazio delle funzioni (tipo funzioni è un caso particolare).

Le regole del prodotto indicato dipendente che agisce su un tipo dipendente sono le seguenti.

8.8.1 Regole di Formazione

$$F\text{-}\prod) \frac{B \text{ type } [\Gamma] \quad C(x) \text{ type } [\Gamma, x \in B]}{\prod_{x \in B} C(x) \text{ type } [\Gamma]}$$

8.9 Regole di Introduzione

$$I\text{-}\prod) \frac{c(x) \in C(x)[\Gamma, x \in B]}{\lambda x^B. c(x) \in \prod_{x \in B} C(x)[\Gamma]}$$

8.10 Regole di Eliminazione

$$E\text{-}\prod) \frac{f \in \prod_{x \in B} C(x)[\Gamma] \quad b \in B[\Gamma]}{Ap(f, b) \in C(b)[\Gamma]}$$

8.11 Regole di Conservazione

$$C\text{-}\prod) \frac{c(x) \in C(x)[\Gamma, x \in B] \quad b \in B[\Gamma]}{Ap(\lambda x^B. c(x), b) = c(b) \in C(b)[\Gamma]}$$

8.12 Regole di Uguaglianza

$$\text{eq-E-}\Pi) \frac{f_1 = f_2 \in \prod_{x \in B} C(x)[\Gamma] \quad b_1 = b_2 \in B[\Gamma]}{\text{Ap}(f_1, b_1) = \text{Ap}(f_2, b_2) \in C(b_1)[\Gamma]}$$

$$\text{eq-I-}\Pi) \frac{c_1(x) = c_2(x) \in C(x)[\Gamma, x \in B]}{\lambda x^B. c_1(x) = \lambda x^B. c_2(x) \in \prod_{x \in B} C(x)[\Gamma]}$$

La regola (eq-I- Π) prende anche il nome di ξ -rule ed è una regola difficile da modellare.

$$\text{eq-F-}\Pi) \frac{B_1 = B_2 \in \text{type}[\Gamma] \quad c_1(x) = c_2(x) \text{ type}[\Gamma, x \in B_1]}{\prod_{x \in B} C_1(x)[\Gamma] = \prod_{x \in B} C_2(x)[\Gamma]}$$

8.13 Osservazioni dal punto di vista logico

8.13.1 La regola di Formazione

Se prendo la regola di *Formazione* e ci metto le preposizioni, questa diventa

$$\text{F-}\Pi) \frac{B \text{ type } [\Gamma] \quad \varphi(x) \text{ prop } [\Gamma, x \in B]}{\forall_{x \in B} \varphi(x) \text{ prop } [\Gamma]}$$

che rende vera la seguente definizione

$$\forall_{x \in B} \varphi(x) \stackrel{\text{def}}{=} \prod_{x \in B} \varphi(x)$$

dalle regole (I- Π) e (E- Π).

La regola di Introduzione

$$\text{I-}\Pi) \frac{c(x) \in \varphi(x)[\Gamma, x \in B]}{? \in \prod_{x \in B} \varphi(x)[\Gamma]}$$

che definisce nella *set-theory* la regola del \forall . Difatti, la regola, con le opportune sostituzioni diventa

$$\text{I-}\Pi) \frac{\varphi(x) \text{ vero}[\Gamma, \Delta, x \in B]}{\forall_{x \in B} \varphi(x) \text{ vero}[\Gamma, \Delta]}$$

con $\Delta \equiv [x_1 \dots \varphi_1 \dots x_n \dots \varphi_n]$ e $\varphi_i \text{ prop}[\Gamma]$

Nel calcolo dei sequenti equivale al per ogni a destra

$$\forall\text{-D}) \frac{\Delta \vdash_{\Gamma, x \in B} \varphi(x)}{\Delta \vdash_{\Gamma} \forall_{x \in B} \varphi(x)} \quad x \notin \text{fv}(\Delta)$$

La regola di Eliminazione

$$\text{E-}\Pi) \frac{f \in \forall_{x \in B} \varphi(x)[\Gamma] \quad b \in B[\Gamma]}{? \in \varphi(b)[\Gamma]}$$

che con le opportune sostituzioni diventa, nel calcolo dei sequenti, una regola di eliminazione della riduzione naturale

$$\text{E-}\Pi) \frac{\Delta \vdash \forall_{x \in B} \varphi(x) \quad b \in B[\Gamma]}{\Delta \vdash \varphi(b)}$$

con $\Delta \equiv$ tutte assunzioni di $\text{prop}[\Gamma]$

Per concludere, il prodotto dipendente definisce sia:

1. L'implicazione

$$\beta \rightarrow \gamma \equiv \prod_{x \in B} \gamma$$

dove \rightarrow indica famiglia costante

2. La quantificazione esistenziale

$$\forall_{x \in B} \varphi(x) \equiv \prod_{x \in B} \varphi(x)$$

Capitolo 9

Tipo vuoto

Il tipo vuoto (N_0 , *empty-set*) è un tipo induttivo generato dalle sue regole di introduzione (che non ci sono).

Il tipo N_0 è definito dalle seguenti regole.

9.1 Regole di Formazione

$$N_0\text{-F}) \frac{\Gamma \text{ cont}}{N_0 \text{ type } [\Gamma]}$$

9.2 Regole di Introduzione

$N_0[\Gamma] = \emptyset \Rightarrow$ non ci sono valori canonici

Non ci sono le regole di Introduzione, in quanto N_0 si vuole che sia un tipo vuoto, e di conseguenza non serve introdurre elementi

9.3 Regole di Eliminazione

$$E\text{-}N_0) \frac{t \in N_0[\Gamma] \quad M(z) \text{ type } [\Gamma, z \in N_0]}{\text{El}_{N_0}(t) \in M(t)[\Gamma]}$$

9.4 Regole di Conservazione

Non esiste alcuna regola di Conversione, a causa della mancanza di ipotesi dell'eliminatore sui termini $M(0)$.

9.5 Regole di Uguaglianza

$$\text{eq-E-}N_0) \frac{t_1 = t_2 \in N_0[\Gamma] \quad M(z) \text{ type } [\Gamma, z \in N_0]}{\text{El}_{N_0}(t_1) = \text{El}_{N_0}(t_2) \in M(t_1)[\Gamma]}$$

Non serve nessun'altra ipotesi non essendoci alcun elemento canonico. Inoltre questo comporta la non necessità delle altre regole di uguaglianza

9.6 Eliminatorio dipendente

$$E\text{-}N_{0\text{dip}} \frac{M(z) \text{ type}[\Gamma, z \in N_0]}{El_{N_0}(z) \in M(z)[\Gamma, z \in N_0]}$$

9.7 Semantica operativa del tipo funzione

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

- β_{\rightarrow} -red) NON ESISTE
- Per far interagire N_0 con altri tipi (altrimenti N_0 da solo non fa nulla)
 $N_0\text{-red}) \frac{t_1 \rightarrow_1 t_2}{El_{N_0}(t_1) \rightarrow_1 El_{N_0}(t_2)}$

La regola ($N_0\text{-red}$) è associata all'uguaglianza dell'eliminazione.

9.8 Osservazioni dal punto di vista logico

Il tipo vuoto serve in logica per interpretare il falso

$$\perp \stackrel{def}{=} N_0$$

\perp è vero sse $\exists \in N_0$

$$E_{N_0} \frac{t \in N_0[\Gamma] \quad \text{ind-ty} \frac{\perp \text{ è vero}[\Gamma] \quad \varphi \text{ prop}[\Gamma]}{\varphi \text{ prop}[\Gamma, z \in N_0]}}{El_{N_0}(t) \in \varphi[\Gamma]}$$

Quella appena definita rappresenta la regola $\frac{\perp \text{ vero}[\Gamma]}{\varphi \text{ vero}[\Gamma]}$

Inoltre, con la regola di *Eliminazione dipendente*, definita in §9.6, posso dire che, per ogni $\varphi \text{ prop}[\Gamma]$
 $\Rightarrow M(z) \text{ type}[\Gamma, z \in N_0] \equiv \varphi(z) \text{ è prop}$
 $\Rightarrow El_{N_0}(z) \in M(z)[\Gamma, z \in N_0] \equiv \varphi \text{ è vero}[\Gamma, \Delta, N_0 \text{ vero}] \equiv \perp \vdash \varphi$ che è l'**assioma del falso** in logica ($\perp\text{-ax}$)

In conclusione, dunque, la regola di eliminazione di N_0 rappresenta $\perp\text{-ax}$ in calcolo dei sequenti/regola del falso *quodlibet* in deduzione naturale.

Capitolo 10

La logica della teoria dei tipi di *Martin-Löf*

Molti tipi, come visto nei capitoli precedenti hanno un significato logico.
Ma qual'è la logica che viene affrontata in type-theory?

I tipi descritti fino a ora sono sufficienti per interpretare le formule, con predicati dipendenti dal tipo, della logica **predicativa con l'uguaglianza**. Tuttavia la logica che la teoria dei tipi, di questi tipi, rende valida è la **logica intuizionistica** e non quella classica.

Quale è la logica valida in teoria dei tipi?

Dipende dalla teoria dei tipi (dalla teoria dei tipi di *Martin-Löf* si rende valida solo quella intuizionistica).

Di seguito do la definizione per induzione sulle formule. In queste il contesto tipa le variabili libere della formula.

$$\begin{aligned}(\perp)^I[\Gamma] &\stackrel{def}{=} N_0 \text{ (tipo vuoto) type}[\Gamma] \\(tt)^I[\Gamma] &\stackrel{def}{=} N_1 \text{ (tipo singoletto) type}[\Gamma] \\(\varphi \vee \psi)^I[\Gamma] &\stackrel{def}{=} \varphi^I + \psi^I \text{ type}[\Gamma] \\(\varphi \& \psi)^I[\Gamma] &\stackrel{def}{=} \varphi^I \times \psi^I \text{ type}[\Gamma] \\(\varphi \rightarrow \psi)^I[\Gamma] &\stackrel{def}{=} \varphi^I \rightarrow \psi^I \text{ type}[\Gamma] \\(\neg \varphi)^I &\stackrel{def}{=} \varphi^I \rightarrow \perp^I \stackrel{def}{=} \varphi^I \rightarrow N_0 \text{ type}[\Gamma] \\(\forall_{x \in A} \varphi(x)) &\stackrel{def}{=} \prod_{x \in A} (\varphi(x))^I[\Gamma, x \in A] \text{ type}[\Gamma] \\(t \equiv s)^I[\Gamma] &\stackrel{def}{=} \text{Id}(A, t, s) \text{ type}[\Gamma] \Rightarrow \text{unico predicato atomico} \\(\exists_{x \in A} \varphi(x))^I[\Gamma] &\stackrel{def}{=} \sum_{x \in A} (\varphi(x))^I[\Gamma]\end{aligned}$$

Da tenere presente che per noi vale

$$\varphi(\mathbf{x}) \text{ type}[\mathbf{x} \in \mathbf{A}]$$

e in generale una formula $\varphi(x)$ $\text{prop}[x \in A]$, con prop che dipendono da tipi e A tipi.

Invece sono formule non valide ~~$\forall \varphi(x) \exists \varphi(x)$~~ , in quanto le variabili senza tipo non esistono in teoria dei tipi.

Lemma per ogni formula interpretata, il giudizio di tipo che la interpreta è derivabile nella teoria dei tipi.

Definizione

Con il giudizio α **prop** $[\Gamma]$ (sintassi della logica del primo ordine), con α formula del linguaggio predicativo con l'uguaglianza a variabili tipate, si intende il giudizio α^I **type** $[\Gamma]$ (in teoria dei tipi), ove questo è la traduzione di α nel linguaggio di una formula predicativa con l'uguaglianza.

Va ricordato che α **prop** $[\Gamma] \equiv$ predicato che è una formula proposizionale su Γ

$$\alpha \text{ **prop** (secondo il solito uso di preposizione) } \quad \text{sse} \quad \alpha \text{ **prop** }[\Gamma]$$

Definizione

Introduciamo il concetto di **dimostrazioni logiche**.

Date le preposizioni ϕ **prop** $[\Gamma]$ e α_i **prop** $[\Gamma]$ per $i = 1, \dots, n$ derivabili in teoria dei tipi.

Introduciamo il **giudizio** ϕ **true** $[\Gamma, \alpha_1 \text{ true}, \dots, \alpha_n \text{ true}]$. In questo modo, si definisce il **contesto spuro**, dove Γ è il solito contesto e $\alpha_1 \text{ true}, \dots, \alpha_n \text{ true}$ è un contesto nuovo di assunzioni di proposizioni.

Introdotta il giudizio diciamo che è derivabile se esiste un termine (detto *proof-term*) pf tale che $pf \in \phi[\Gamma, x_1 \in \alpha_1, \dots, x_n \in \alpha_n]$ è derivabile in una teoria dei tipi.

Definizione

(Per ricollegare il calcolo del primo ordine con la logica del primo anno)

Preso un sequente, in formule della logica predicativa con l'uguaglianza e variabili, nel linguaggio L , tipate vale che

$$(\Sigma \vdash \Delta)^I \stackrel{\text{def}}{=} (\Delta^\vee)^I \text{ **type** } [\Gamma, x \in (\Sigma^\&)^I]$$

Per il quale vale una definizione per induzione dei termini:

- $\Sigma^\&$
 - $[\]^\& \equiv tt$
 - $[\Sigma^I, \varphi]^\& \equiv (\Sigma^\&) \& \varphi$
- Δ^\vee
 - $[\]^\vee \equiv \perp$
 - $[\Delta^I, \varphi]^\vee \equiv (\Delta^\vee) \vee \varphi$

Definizione

Un sequente $\Sigma \vdash_{\Gamma} \Delta$ è valido in teoria dei tipi sse $(\Delta^{\vee})^I \text{ true } [\Gamma, (\Sigma^{\&})^I \text{ true}]$ è derivabile.

Ovvero se esiste un pf tale che $\text{pf} \in (\Delta^{\vee})^I [\Gamma, x \in (\Sigma^{\&})^I]$.

Si può dimostrare $(\Sigma \vdash_{\Gamma} \Delta)^I$ è derivabile sse $(\Delta^{\vee})^I \text{ type}[\Gamma]$ è derivabile.

Teorema i sequenti derivabili del calcolo della deduzione naturale *DNI*¹ sono validi.

Non sono validi quelli della logica classica. Perché? Per il Principio del terzo escluso.

Il Principio del terzo escluso non è valido in teoria dei tipi perchè per ogni φ noi dovremmo dedurre, in teoria dei tipi, un $\text{pf} \in (\varphi \vee \neg\varphi)^I [] \equiv \varphi^I + \neg\varphi^I []$.

DNI + Principio del terzo escluso \equiv calcolo dei sequenti della logica classica
prediativa con l'uguaglianza

\Rightarrow logica classica = logica intuizionistica + la derivabilità $\vdash \varphi \vee \neg\varphi$

Il Principio del terzo escluso è la derivabilità $\vdash \varphi \vee \neg\varphi$, per la formula φ .
Ma $\vdash \varphi \vee \neg\varphi$ non è valido in teoria dei tipi perchè esiste G , formula, per cui $\vdash G \vee \neg G$ non è valido.

Idea della Dimostrazione

Metto assieme il teorema di *Gödel* con quello della forma normale forte.

$$\text{pf} \in G \vee \neg G = G^I + \neg G^I$$

Questa va ridotta in forma normale, che corrisponde a una delle due forme seguenti:

- $\text{NF}(\text{pf}) = \text{inl}(\text{pf}_1)$ con $\text{pf}_1 \in G[]$
- $\text{NF}(\text{pf}) = \text{inr}(\text{pf}_2)$ con $\text{pf}_2 \in G[]$

Ma *Gödel* ha dimostrato che nè $\vdash G[]$ e neanche $\vdash \neg G[]$ sono derivabili (\Leftrightarrow non valido in teoria dei tipi.)

¹Logica Intuizionistica.

Capitolo 11

Principi di induzione per tipi induttivi

Principi di induzione associati alle regole di eliminazione dei cosiddetti tipi induttivi T , per renderne la ragione del nome; quando il tipo $M(z)$ $\text{type}[\Gamma, z \in T]$, verso cui si elimina, è in realtà una $\varphi(z)$ $\text{prop}[\Gamma, z \in T]$.

Tipi induttivi: N_0 , N_1 , $\text{List}(A)$, Nat , $B + C$, $\sum_{x \in B} C(x)$, $\text{Id}(A, a, b)$

11.1 Proprietà per i tipi induttivi

Per poter enunciare il principio di induzione, ci servono delle **proprietà di manipolazione dei contesti**.

- Σ serve per **raggruppare le assunzioni in un contesto**:
 $c(x, y) \in C(x, y)[x \in A, y \in B]$ derivabile $\Rightarrow c(\pi_1 z, \pi_2 z) \in C(\pi_1 z, \pi_2 z)[z \in \sum_{x \in A} B(x)]$ è derivabile
Si può anche andare nella direzione opposta, dunque
 $d(z) \in D(z)[z \in \sum_{x \in A} B(x)]$ derivabile $\Rightarrow d(\langle x, y \rangle) \in D(\langle x, y \rangle)[x \in A, y \in B(x)]$
- Π serve per **togliere le assunzioni in un contesto**:
 $c(x, y) \in C(x, y)[x \in A, y \in B(x)]$ derivabile $\Rightarrow \lambda y. C(x) \in \Pi_{y \in B(x)} C(x, y)[x \in A]$
Per cui posso sempre togliere l'ultima assunzione.

Notazione: quando ho $\varphi(z) \rightarrow \varphi^I(z)$

Di seguito enuncio i vari tipi di induzione, con spiegazione per il principio dei numeri Naturali.

11.2 Principio di induzione numeri Naturali

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

È da tale regola che si ricavano i principi di induzione.

Esempio

$$\text{E-Nat}_{dip} \frac{\varphi(z)\text{prop}[\Gamma, z \in \text{Nat}] \quad c \in \varphi(0)[\Gamma] \quad e(x,y) \in \varphi(\text{succ}(x))[\Gamma, z \in \text{Nat}, y \in M(x)]}{\text{El}_{Nat}(z,c,e) \in \varphi(z)[\Gamma, z \in \text{Nat}]}$$

$$\Leftrightarrow \lambda z. \text{El}_{Nat}(z,c,e) \in \forall_{z \in \text{Nat}} \varphi(z)$$

A questo punto posso raggruppare in un unico sequente

$$\begin{aligned} & \lambda z. \text{El}_{Nat}(z,w,(x,y)e(x,y)) \in \forall_{z \in \varphi(z)} [\Gamma, w \in \varphi(0); z_f \in \forall_{x \in \text{Nat}} (\varphi(x) \rightarrow \varphi(\text{succ}(x)))] \\ \Rightarrow & \lambda z. \text{El}_{Nat}(z,w,(x,y).\text{Ap}(\text{Ap}(z,f,x),y)) \in \forall_{z \in \varphi(z)} [\Gamma, w \in \varphi(0) \times / \& z_f \in \forall_{x \in \text{Nat}} \\ & (\varphi(x) \rightarrow \varphi(\text{succ}(x)))] \end{aligned}$$

Assumendo che, quando astraggo, $\varphi \text{ true}[\Gamma, \Psi \text{ true}]$ derivabile $\Leftrightarrow \Psi \rightarrow \varphi \text{ true}[\Gamma]$

In questo modo arrivo ad avere il principio di induzione che rende valido l'eliminatore dei numeri Naturali

$$\text{Nat}) \varphi(0) \& \forall_{x \in \text{Nat}} (\varphi(x) \rightarrow \varphi(\text{succ}(x))) \rightarrow \forall_{z \in \text{Nat}} \varphi(z)$$

è valido e segue dalla E-Nat_{dip}

11.3 Principio di induzione su N_0

$$N_0) (\perp \text{ true} \rightarrow \varphi[\Gamma])$$

è valido per ogni $\varphi \text{ prop}[\Gamma]$

11.4 Principio di induzione su N_1

$$N_1) \varphi(*) \rightarrow \forall_{x \in N_1} \varphi(x) \text{ true}[\Gamma]$$

è valido per ogni $\varphi(z) \text{ prop}[\Gamma, z \in N_1]$

11.5 Principio di induzione su $\text{List}(A)$

Supponiamo $A \text{ type}[\Gamma]$ derivabile

$$\text{List}(A)) \varnothing(\text{nil}) \& \forall_{x \in \text{List}(A)} \forall_{w \in A} (\varnothing(x) \rightarrow \varnothing(\text{cons}(x, w))) \rightarrow \forall_{z \in \text{List}(A)} \varphi(z) \text{ true}[\Gamma]$$

è derivabile

11.6 Principio di induzione su $B+C$

$\varphi(z)$ prop[$\Gamma, z \in B + C$] derivabile

$$B+C) \forall_{x_1 \in B} \varphi(nl(x_1)) \ \& \ \forall_{x_2 \in C} \varphi(nr(x_2)) \rightarrow \forall_{z \in B+C} \varphi(z) \text{ true}[\Gamma]$$

è derivabile

11.7 Principio di induzione su $\sum_{x \in B} C(x)$

Supponiamo $\varphi(z)$ prop[$\Gamma, z \in B + C$]

$$\Sigma) \forall_{x \in B} \forall_{y \in C(x)} \varnothing(< x, y >) \rightarrow \forall_{z \in \sum_{x \in B} C(x)} \varphi(z) \text{ true}[\Gamma]$$

è derivabile

con $\forall_{z \in \sum_{x \in B} C(x)}$ famiglia di insiemi = unione indicata disgiunta

11.8 Principio di induzione su $\text{Id}(A, a, b)$

È un tipo induttivo, grande novità, introdotta da *Martin-Löf*, all'interno della teoria dei tipi.

$$\begin{aligned} &\text{Dato } \varphi(z_1, z_2, z_3) \text{ prop}[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)] \text{ derivabile} \\ &\forall_{x \in A} \varphi(x, x, \text{id}(x)) \rightarrow \forall_{z_1 \in A} \forall_{z_2 \in A} \forall_{z_3 \in \text{Id}(z_1, z_2, z_3)} \varphi(z_1, z_2, z_3) \text{ true}[\Gamma] \end{aligned}$$

è derivabile

Tale principio non è formulato su $\varphi(z)$ type[$\Gamma, z \in \text{Id}(A, a, b)$] con A e a, b fissati, ma dipende da tre parametri z_1, z_2 e z_3 , come accade nella regola del E-Id.

In conclusione, ho mostrato, come i **principi di induzione** servono per dimostrare le preposizioni; e le **regola di eliminazione** non determinano solo ricursori, ma anche regole di induzione.

11.9 Esercizi

2 Mostrare che per ogni tipo $B + C$ esiste un *proof-term* del tipo

$$\begin{aligned} &\text{Id}(B + C, z, z') \leftrightarrow \\ &\exists_{x \in B} \exists_{x' \in B} \text{Id}(B + C, z, \text{inl}(x)) \wedge \text{Id}(B + C, z', \text{inl}(x')) \wedge \text{Id}(B, x, x') \\ &\quad \vee \\ &\exists_{y \in C} \exists_{y' \in C} \text{Id}(B + C, z, \text{inr}(y)) \wedge \text{Id}(B + C, z', \text{inr}(y')) \wedge \text{Id}(B, y, y') \\ &\text{type}[z \in B + C, z' \in B + C] \end{aligned}$$

Soluzione

Appendice A

Tipi principali e regole

A.1 Il tipo singoletto

A.1.1 Regola di Formazione

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

A.1.2 Regole di Introduzione

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

A.1.3 Regole di Eliminazione

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

A.1.4 Regole di Conversione

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

A.1.5 Eliminatore dipendente

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

A.2 Tipo dei numeri Naturali

A.2.1 Regole di Formazione

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

A.2.2 Regole di Introduzione

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

A.2.3 Regole di Eliminazione

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

A.2.4 Regole di Conversione

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

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

A.2.5 Regole di Uguaglianza

$$\text{eq-Nat)} \frac{t_1 = t_2 \in \text{Nat}[\Gamma]}{\text{succ}(t_1) = \text{succ}(t_2) \in \text{Nat}[\Gamma]}$$

A.2.6 Introduzione ed Eliminatore dipendente

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

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

A.3 Tipo delle liste di un tipo

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

A.3.1 Regole di Introduzione

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

A.3.2 Regole di Eliminazione

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

A.3.3 Regole di Conservazione

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

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

A.3.4 Regole di Uguaglianza

$$\text{eq-I}_1\text{-List)} \frac{A_1 = A_2 \in \text{type}[\Gamma]}{\text{List}(A_1) = \text{List}(A_2) \text{ type}(\Gamma)}$$

$$\text{eq-I}_2\text{-List)} \frac{s_1 = s_2 \in \text{List}(A)[\Gamma] \quad a_1 = a_2 \in A[\Gamma]}{\text{cons}(s_1,a_1) = \text{cons}(s_2,a_2) \in \text{List}(A)(\Gamma)}$$

$$\text{E-eq-List)} \frac{\begin{array}{l} M(z) \text{ type}[\Gamma, z \in \text{List}(A)] \\ t_1 = t_2 \in \text{list}(A)[\Gamma] \\ c_1 = c_2 \in M(\text{nil})[\Gamma] \end{array} \quad e_1(x,w,y) = e_2(x,w,y) \in M(\text{cons}(x,w))[\Gamma, x \in \text{List}(A), w \in A, y \in M(x)]}{\text{El}_{list}(t_1, c_1, e_1) = \text{El}_{list}(t_2, c_2, e_2 \in M(t_1)[\Gamma])}$$

A.3.5 Eliminatore dipendente

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

A.4 Tipo della somma disgiunta

A.4.1 Regole di Formazione

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

A.4.2 Regole di Introduzione

$$I_1-+) \frac{b \in B[\Gamma] \quad B + C \text{ type}[\Gamma]}{\text{inl}(b) \in B + C[\Gamma]} \quad I_2-+) \frac{c \in C[\Gamma] \quad B + C \text{ type}[\Gamma]}{\text{inr}(c) \in B + C[\Gamma]}$$

A.4.3 Regole di Eliminazione

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

A.4.4 Regole di Conservazione

$$C_1-+) \frac{\begin{array}{c} M(z) \text{ type}[\Gamma, z \in B + C] \\ b \in B[\Gamma] \end{array} \quad e_B(x_1) \in M(\text{inl}(x_1))[\Gamma, x_1 \in B] \quad e_C(x_2) \in M(\text{inr}(x_2))[\Gamma, x_2 \in C]}{\text{El}_+(\text{inl}(b), e_B, e_C) = e_B(b) \in M(\text{inl}(b))[\Gamma]}$$

$$C_1-+) \frac{\begin{array}{c} M(z) \text{ type}[\Gamma, z \in B + C] \\ c \in C[\Gamma] \end{array} \quad e_B(x_1) \in M(\text{inl}(x_1))[\Gamma, x_1 \in B] \quad e_C(x_2) \in M(\text{inr}(x_2))[\Gamma, x_2 \in C]}{\text{El}_+(\text{inr}(c), e_B, e_C) = e_C(c) \in M(\text{inr}(c))[\Gamma]}$$

A.4.5 Regole di Uguaglianza

$$\text{eq-F-+)} \frac{B_1 = B_2 \in \text{type}[\Gamma] \quad C_1 = C_2 \in \text{type}[\Gamma]}{B_1 + C_1 = B_2 + C_2 \text{ type}(\Gamma)}$$

A.4.6 Eliminatore dipendente

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

A.5 Tipo dell'uguaglianza proposizionale

A.5.1 Regole di Formazione

$$\text{F-Id)} \frac{A \text{ type } [\Gamma] \quad a \in A[\Gamma] \quad b \in A[\Gamma]}{\text{Id}(A, a, b) \text{ type } [\Gamma]}$$

A.5.2 Regole di Introduzione

$$\text{I-Id)} \frac{a \in A[\Gamma]}{\text{id}(a) \in \text{Id}(A, a, a)[\Gamma]}$$

A.5.3 Regole di Eliminazione

$$\text{E-Id)} \frac{\begin{array}{c} M(z_1, z_2, z_3) \text{ type } [\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)] \\ a \in A[\Gamma] \\ b \in A[\Gamma] \\ t \in \text{Id}(A, a, b)[\Gamma] \end{array} \quad e(x) \in M(x, x, \text{Id}(x))[\Gamma, x \in A]}{\text{El}_{Id}(t, (x).e(x)) \in M(a, b, t)[\Gamma]}$$

A.5.4 Regole di Conservazione

$$\text{E-Id)} \frac{\begin{array}{c} M(z_1, z_2, z_3) \text{ type } [\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)] \\ a \in A[\Gamma] \end{array} \quad e(x) \in M(x, x, \text{Id}(x))[\Gamma, x \in A]}{\text{El}_{Id}(\text{Id}(a), (x).e(x)) = e(a) \in M(a, a, \text{Id}(a))[\Gamma]}$$

A.5.5 Regole di Uguaglianza

$$\text{eq-F-Id)} \frac{A_1 = A_2 \text{ type } [\Gamma] \quad a_1 = a_2 \in A[\Gamma] \quad b_1 = b_2 \in A[\Gamma]}{\text{Id}(A_1, a_1, b_1) = \text{Id}(A_2, a_2, b_2) \text{ type } [\Gamma]}$$

$$\text{eq-I-Id)} \frac{a_1 = a_2 \in A[\Gamma]}{\text{Id}(a_1) = \text{Id}(a_2) \in \text{Id}(A, a_1, a_1)[\Gamma]}$$

A.5.6 Eliminatore dipendente

$$\text{E-Id}_{dip)} \frac{M(z_1, z_2, z_3) \text{ type } [\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)] \quad e(x) \in M(x, x, \text{Id}(x))[\Gamma, x \in A]}{\text{El}_{Id}(z_3, (x).e(x)) \in M(z_1, z_2, z_3)[\Gamma, z_1 \in A, z_2 \in A, z_3 \in \text{Id}(A, z_1, z_2)]}$$

A.6 Tipo somma indicata forte

A.6.1 Regole di Formazione

$$F-\Sigma) \frac{C(x) \text{ type } [\Gamma, x \in B]}{\sum_{x \in B} C(x) \text{ type } [\Gamma]}$$

A.6.2 Regole di Introduzione

$$I-\Sigma) \frac{k \in B[\Gamma] \quad c \in C(b)[\Gamma] \quad \sum_{x \in B} C(x) \text{ type } [\Gamma]}{\langle b, c \rangle \in \sum_{x \in B} C(x) \text{ type } [\Gamma]}$$

A.6.3 Regole di Eliminazione

$$E-\Sigma) \frac{M(z) \text{ type } [\Gamma, z \in \sum_{x \in B} C(x)] \quad t \in \sum_{x \in B} C(x)[\Gamma] \quad e(x, y) \in M(\langle x, y \rangle)[\Gamma, x \in B, y \in C(x)]}{El_{\Sigma}(t, (x, y).e(x, y)) \in M(t)[\Gamma]}$$

A.6.4 Regole di Conservazione

$$C-\Sigma) \frac{M(z) \text{ type } [\Gamma, z \in \sum_{x \in B} C(x)] \quad b \in B[\Gamma] \quad c \in C(b) \quad e(x, y) \in M(\langle x, y \rangle)[\Gamma, x \in B, y \in C(x)]}{El_{\Sigma}(\langle b, c \rangle, e) = e(b, c) \in M(\langle b, c \rangle)[\Gamma]}$$

A.6.5 Regole di Uguaglianza

$$eq-F-\Sigma) \frac{B_1 = B_2 \text{ type } [\Gamma] \quad C_1(x) = C_2(x) \text{ type } [\Gamma, x \in B_1]}{\sum_{x \in B_1} C_1(x) = \sum_{x \in B_2} C_2(x) \text{ type } [\Gamma]}$$

$$eq-I-\Sigma) \frac{\sum_{x \in B} C(x) \text{ type } [\Gamma] \quad b_1 = b_2 \in B[\Gamma] \quad c_1 = c_2 \in C(b_1)[\Gamma]}{\langle b_1, c_1 \rangle = \langle b_2, c_2 \rangle \in \sum_{x \in B} C(x) \text{ type } [\Gamma]}$$

$$eq-E-\Sigma) \frac{M(z) \text{ type } [\Gamma, z \in \sum_{x \in B} C(x)] \quad t_1 = t_2 \in \sum_{x \in B} C(x)[\Gamma] \quad e_1(x, y) = e_2(x, y) \in M(\langle x, y \rangle)[\Gamma, x \in B, y \in C(x)]}{El_{\Sigma}(t_1, e_1) = El_{\Sigma}(t_2, e_2) \in M(t_1)[\Gamma]}$$

A.6.6 Eliminatore dipendente

$$E-\Sigma_{dip}) \frac{M(z) \text{ type } [\Gamma, z \in \sum_{x \in B} C(x)] \quad e(x, y) \in M(\langle x, y \rangle)[\Gamma, x \in B, y \in C(x)]}{El_{\Sigma}(z, (x, y).e(x, y)) \in M(z)[\Gamma, z \in \sum_{x \in B} C(x)]}$$

A.7 Tipo prodotto cartesiano

A.7.1 Regole di Formazione

$$\text{F-x)} \frac{B \text{ type } [\Gamma] \quad C \text{ type } [\Gamma]}{B \times C \text{ type } [\Gamma]}$$

A.7.2 Regole di Introduzione

$$\text{I-x)} \frac{b \in B[\Gamma] \quad c \in C[\Gamma]}{\langle b, c \rangle \in B \times C[\Gamma]}$$

A.7.3 Regole di Proiezione

$$\text{PJ}_{1\text{-x}}) \frac{d \in B \times C[\Gamma]}{\pi_1 d \in B[\Gamma]}$$

$$\text{PJ}_{2\text{-x}}) \frac{d \in B \times C[\Gamma]}{\pi_2 d \in C[\Gamma]}$$

A.7.4 Regole di Uguaglianza delle proiezioni

$$\text{PJ}_{1\text{-eq}}) \frac{b \in B[\Gamma] \quad c \in C[\Gamma]}{\pi_1(\langle b, c \rangle) = b \in B[\Gamma]}$$

$$\text{PJ}_{2\text{-eq}}) \frac{b \in B[\Gamma] \quad c \in C[\Gamma]}{\pi_2(\langle b, c \rangle) = c \in C[\Gamma]}$$

A.8 Tipo delle funzioni

A.8.1 Regole di Formazione

$$F \rightarrow) \frac{B \text{ type } [\Gamma] \quad C \text{ type } [\Gamma]}{B \rightarrow C \text{ type } [\Gamma]}$$

A.8.2 Regole di Introduzione

$$I \rightarrow) \frac{c(x) \in C[\Gamma, x \in B]}{\lambda x^B. c(x) \in B \rightarrow C[\Gamma]}$$

A.8.3 Regole di Eliminazione

$$E \rightarrow) \frac{f \in B \rightarrow C[\Gamma] \quad b \in B[\Gamma]}{\text{Ap}(f, b) \in C[\Gamma]}$$

A.8.4 Regole di Conservazione

$$C \rightarrow) \frac{c(x) \in C[\Gamma, x \in B] \quad b \in B[\Gamma]}{\text{Ap}(\lambda x^B. c(x), b) = c(b) \in C[\Gamma]}$$

A.8.5 Regole di Uguaglianza

$$\text{eq-E} \rightarrow) \frac{f_1 = f_2 \in B \rightarrow C[\Gamma] \quad b_1 = b_2 \in B[\Gamma]}{\text{Ap}(f_1, b_1) = \text{Ap}(f_2, b_2) \in C[\Gamma]}$$

$$\text{eq-I} \rightarrow) \frac{c_1(x) = c_2(x) \in C[\Gamma, x \in B]}{\lambda x^B. c_1(x) = \lambda x^B. c_2(x) \in B \rightarrow C[\Gamma]}$$

A.9 Tipo vuoto

A.9.1 Regole di Formazione

$$N_0\text{-F}) \frac{\Gamma \text{ cont}}{N_0 \text{ type } [\Gamma]}$$

A.9.2 Regole di Eliminazione

$$E\text{-}N_0) \frac{t \in N_0[\Gamma] \quad M(z) \text{ type}[\Gamma, z \in N_0]}{\text{El}_{N_0}(t) \in M(t)[\Gamma]}$$

A.9.3 Regole di Uguaglianza

$$\text{eq-}E\text{-}N_0) \frac{t_1 = t_2 \in N_0[\Gamma] \quad M(z) \text{ type}[\Gamma, z \in N_0]}{\text{El}_{N_0}(t_1) = \text{El}_{N_0}(t_2) \in M(t_1)[\Gamma]}$$

A.9.4 Eliminatore dipendente

$$E\text{-}N_{0dip}) \frac{M(z) \text{ type}[\Gamma, z \in N_0]}{\text{El}_{N_0}(z) \in M(z)[\Gamma, z \in N_0]}$$