INTRODUZIONE ALLA TEORIA DEI TIPI DI MARTIN LÖF

L. CROSILLA

Appunti per le lezioni tenute in occasione del seminario
Tipi e Informazione
Prof. A. Cantini, Prof. P. Minari
Università degli Studi di Firenze
Dipartimento di Filosofia
A.A. 2003 - 04, Primo semestre[1]

Date: 18 Novembre 2003.

[1]Questi appunti sono scritti in LaTeX 2_ε. Nel redigere le regole della teoria dei tipi mi sono avvalsa della macro *bussproofs.sty* di S. Buss, University of California, San Diego.

Indice

1. Prerequisiti e suggerimenti bibliografici

Oggetto di queste lezioni è la presentazione delle basi della teoria degli insiemi che costituisce parte fondamentale della teoria costruttiva dei tipi di Martin Löf.

1.1. **Prerequisiti.** Per la trattazione si presuppone una conoscenza di base della logica proposizionale e predicativa del primo ordine, sia intuizionistica che classica, in particolare nella forma di calcolo di deduzione naturale. (Per una esposizione della deduzione naturale, si veda per esempio il Capitolo II di [33], vol. I).

1.2. **Testi fondamentali.** La monografia di Martin Löf del 1984 ([23]) costituisce il principale testo di riferimento. Il libro di Nordström, Petersson e Smith ([25]) fornisce una presentazione più semplice e moderna, orientata alle applicazioni informatiche. Il Capitolo 11 di Beeson ([4]) e i capitoli pertinenti del secondo volume di Troelstra e Van Dalen ([33]) possono costituire una buona introduzione alla teoria dei tipi, anche se, contrariamente a [25], non risultano aggiornati sugli sviluppi più recenti della teoria.

Si segnala che essendo il testo [25] attualmente fuori stampa, è stato dagli autori reso accessibile via Internet all'indirizzo:
`www.cs.chalmers.se/Cs/Research/Logic`

1.3. **Alcuni suggerimenti bibliografici.** Qualora si intenda proseguire nello studio della teoria dei tipi, o la si assuma quale punto di inizio per ricerche relative a tematiche ad essa in vario modo connesse, si possono tenere presente i testi seguenti:

Tra gli scritti di Martin Löf si consigliano: [21], [22].

Testi di approfondimento di carattere storico e filosofico: [31], [18], [11], [17]. Si suggerisce inoltre la lettura di alcuni dei contributi dati da Russell alla teoria dei tipi.

Testi specialistici o di approfondimento: [8], [28], [29], [15], [19].

Testi per approfondire argomenti correlati: [1], [2], [4], [5], [10], [24], [26], [33].

Infine si fa presente che Constable [6] fornisce una ricca bibliografia commentata della teoria dei tipi e di argomenti affini, con particolare attenzione, ancora, agli aspetti di carattere informatico.

2. Introduzione

2.1. **Brevi notizie storiche.** La teoria costruttiva dei tipi nasce intorno ai primi anni '70 ad opera di Per Martin Löf. Martin Löf intende con essa codificare una teoria degli insiemi che costituisca una fondazione per la matematica costruttiva e che sia al contempo adeguata alle nuove esigenze derivanti dallo sviluppo dell'informatica, ovvero che rappresenti anche un flessibile linguaggio di programmazione.

La teoria di Martin Löf si propone presto come fondazione per la matematica costruttiva (soprattutto analisi) così come sviluppata da E. Bishop e dalla sua scuola. Nel 1967 Bishop aveva pubblicato *Foundations of constructive analysis*, testo che dispiegò nuovi orizzonti per la matematica costruttiva. Si mostrava in esso come fosse possibile sviluppare l'analisi sulla base della logica intuizionistica, senza per questo contraddire l'analisi classica; i nuovi risultati furono inoltre presentati con uno stile assai simile a quello tipico degli usuali testi di introduzione all'analisi, facilitandone così la diffusione al di fuori dei ristretti ambiti dei simpatizzanti dell'intuizionismo. In altri termini, l'influente matematico americano dimostrò che era possibile lavorare costruttivamente senza per questo rinunciare ad una parte sostanziale della matematica classica; d'altro canto si perveniva, così facendo, ad un'analisi che assumeva un distintivo carattere algoritmico. L'eco di quel testo fu ampia, tanto che numerosi logici si apprestarono immediatamente a elaborare sistemi formali che fungessero da base per la matematica alla Bishop, che questi e i suoi collaboratori avevano presentato in modo puramente informale. Esempi di rilievo sono la *Explicit mathematics* di S. Feferman ([12]), la Constructive set theory di J. Myhill ([24]), ma anche la teoria degli insiemi intuizionistica di H. Friedmann e J. Myhill (si veda per esempio l'articolo [13]). Come ricordato, anche la teoria dei tipi di Martin Löf si inserisce in questo contesto, sebbene già dall'inizio assuma uno spessore più marcatamente filosofico rispetto ad alcuni sistemi concorrenti.

Nello sviluppare la teoria che porta il suo nome, Martin Löf ha numerosi punti di riferimento. Introdotta da B. Russell per far fronte al paradosso che porta il suo nome, la teoria dei tipi aveva poi assunto nuova forma nell'opera di A. Church. Qui essa si fonde con i concetti propri di un calcolo che ha lo scopo di formalizzare e rendere esplicita la nozione di funzione. Martin Löf riprende l'idea fondamentale russelliana per cui un tipo è una collezione di oggetti che funge da dominio di quantificazione. Si inserisce però anche nella tradizione del Lambda Calcolo, la teoria costruttiva dei tipi costituendo di fatto una particolare estensione (caratterizzata dalla presenza di tipi dipendenti) del Lambda Calcolo tipato. Nell'adattare ad un contesto costruttivo la nozione di tipo, egli ha inoltre presenti le recenti riflessioni di D. Scott [27], G. Kreisel e N. D. Goodman ([19], [20], [16]) (si veda [21] per qualche accenno storico). Egli è inoltre certamente consapevole delle ricerche e degli esperimenti di implemetazione e formalizzazione di porzioni rilevanti di matematica compiuti all'interno del progetto AUTHOMATH, guidato da N. G. de Brujin (si veda per esempio [10]). Proprio nell'ambito del progetto AUTHOMATH si fa per la prima volta ampio uso della cosiddetta corrispondenza di Curry-Howard tra tipi e proposizioni, che è elemento essenziale della teoria costruttiva dei tipi.

La teoria di Martin Löf si sviluppa non senza incontrare difficoltà. La prima versione contemplava infatti l'esistenza di un tipo di tutti i tipi, ma J. Y. Girard ([14]) dimostrò come si potesse da tale assunzione derivare un paradosso (e conseguentemente l'inconsistenza della teoria). Martin Löf presto intervenne a correggere l'errore, ricorrendo ad una più attenta caratterizzazione della teoria che prevenisse l'impredicatività implicita nell'esistenza di un tipo di tutti i tipi: introdusse in suo luogo una gerarchia di tipi universo, costituenti principi di riflessione sempre più forti. Numerose versioni della teoria si sono susseguite (questo contribuì con tutta probabilità a creare alcune delle forme di diffidenza e incomprensione che hanno accompagnato il lento affermarsi della teoria).

2.2. **Teoria dei tipi come teoria degli insiemi.** Nel seguito ci concentreremo sulla teoria degli insiemi che è descritta nel libro [23]. Nelle attuali versioni della teoria dei tipi, questa non è che una particolare porzione di una più ampia teoria, essendo la nozione di insieme una specifica e particolare istanza della più vasta e generica nozione di tipo. Essenzialmente, ciò che caratterizza un insieme è il suo poter essere esaustivamente descritto dando una definizione induttiva dei suoi elementi. Caso paradigmatico di insieme è la collezione dei numeri naturali, $\mathbf{N}$, definita stabilendo che 0 è un elemento di $\mathbf{N}$, e che per ogni elemento di $\mathbf{N}$, il suo successore è elemento di $\mathbf{N}$. Si determina inoltre quando due elementi di $\mathbf{N}$ sono uguali. Nel caso di un tipo generico, invece, non è sempre possibile caratterizzarne gli elementi mediante un processo induttivo; la nozione di tipo è dunque più aperta e meno definita di quella di insieme. Un tipo è semplicemente una collezione di oggetti assieme ad una realazione di equivalenza. Per esempio, la collezione di tutti gli insiemi forma un tipo, ma non un insieme (si veda [25], parte II, per una presentazione della teoria dei tipi come *logical framework*).[2]

Poichè la teoria dei tipi introdotta in [23] si presenta come teoria degli insiemi in cui formalizzare la matematica costruttiva, può essere naturale chiedersi quale sia la sua relazione con la teoria assiomatica degli insiemi, **ZF**, sia nella versione classica che nelle sue versioni costruttive. Tentiamo qui un semplice confronto tra teoria degli insiemi alla Martin Löf e teoria assiomatica degli isiemi alla Zermelo Fraenkel.

Caratteristico della teoria assiomatica degli insiemi è un universo uniforme di oggetti, gli insiemi, i quali sono connessi attraverso due relazioni fondamentali: l'appartenenza e l'uguaglianza. In particolare, gli elementi di un insieme sono a loro volta essi stessi insiemi. Nelle teorie tipate, invece,

[2] La nozione di tipo corrisponde essenzialmente alla nozione di categoria in [23], sebbene essa sia ivi presentata in modo puramente informale.

la relazione di appartenenza lega oggetti di genere diverso: elementi con insiemi oppure oggetti di livello inferiore con oggetti di livello superiore.

La teoria *classica* degli insiemi ha invece come riferimento, secondo la nostra concezione, un universo dato di insiemi, paradigmaticamente rappresentato dalla gerarchia di Von Neumann. Questo universo "platonico" di insiemi viene caratterizzato e descritto attraverso gli assiomi della teoria. A seconda della forza dei principi assiomatici assunti, la teoria data rappresenta un frammento piu' o meno ampio dell'universo di insiemi. Per esempio, aggiungendo alla teoria forti principi di riflessione come assiomi asserenti l'esistenza di grandi cardinali, si ottiene una teoria capace di rappresentare un universo in cui alcuni insiemi costituiscono modelli dell'intera teoria **ZFC**. Nella versione costruttiva della teoria dei tipi, invece, l'universo nel suo complesso non è dato, ma inteso come in fieri, essendovi sempre la possibilità di estenderlo mediante l'introduzione di nuovi costrutti e tipi. L'idea fondamentale è in questo contesto che l'aggiunta di principi più forti non costituisca una più potente caratterizzazione del medesimo universo, ma dia luogo a un nuovo universo, da noi postulato e per questo esistente.

Passiamo ora alla versione *costruttiva* della teoria assiomatica degli insiemi ([24], [1]). Essa è caratterizzata dal far proprio lo stesso linguaggio del primo ordine della teoria Zermelo Fraenkel, poggiandosi però ora sul calcolo intuizionistico anzichè su quello classico. Vengono di conseguenza modificati opportunamente gli assiomi insiemistici classici, cosí da pervenire ad una teoria che ben si presti come fondamento della matematica costruttiva e al contempo eviti il principio del terzo escluso. La somiglianza con la teoria classica è una scelta consapevole che trae chiaramente vantaggio proprio dalla familiarità con essa, per una più facile comprensione e un più agevole uso della nuova assiomatica. Nella teoria costruttiva dei tipi, invece, il calcolo intuizionistico non viene presupposto, ma inserito all'interno della teoria stessa. La teoria dei tipi è al tempo stesso una teoria degli insiemi e un calcolo logico, del tipo del calcolo della deduzione naturale; il passaggio dall'una all'altro è dato semplicemente da una diversa lettura della nozione di insieme. Questo aspetto cruciale della teoria dei tipi è conseguenza del suo far propria la cosiddetta corrispondenza di Curry e Howard. La teoria dei tipi richiede un radicale e nuovo modo di pensare, lontano dalla tradizione; per questo può risultare di difficile accesso per chi in questa tradizione opera da tempo. Il vantaggio è d'altro canto duplice. Da un lato la costruttività della nozione di insieme è evidente, essendo il concetto di insieme postulato in modo originale e non ottenuto attraverso variazioni sintattiche degli assiomi classici. Dall'altro la corrispondenza Curry-Howard consente di leggere la teoria dei tipi come un ricco linguaggio di programmazione, adeguato alla meccanizzazione della matematica costruttiva. Per questo motivo, P. Aczel [1], nel presentare una modifica della teoria assiomatica degli insiemi di J. Myhill, ne ha contemporaneamente fornito una naturale interpretazione in un'appropriata versione di teoria dei tipi di Martin Löf. Egli ha così ottenuto una giustificazione della nozione di insieme che scaturisce dagli assiomi della teoria, ha fornito un contenuto computazionale per i suoi teoremi, ma ha al contempo presentato un sistema formale di facile accesso per il logico tradizionale, abituato a operare all'interno della assiomatica Zermelo Fraenkel.

2.3. **Isomorfismo di Curry-Howard.** Con isomorfismo di Curry-Howard si intende una corrispondenza tra proposizioni e dimostrazioni da un lato e, rispettivamente, tipi e termini (o elementi) dall'altro. L'idea fondamentale alla base della corrispondenza è l'identificazione di una proposizione con l'insieme delle sue dimostrazioni.[3]

La corrispondenza può essere più facilmente compresa confrontando le nozioni di verità classica e intuizionistica. Classicamente ogni proposizione è o vera o falsa, per cui la verità di un enunciato si riduce al suo aver valori entro l'insieme $\{v, f\}$ del vero e del falso. Intuizionisticamente, invece, il concetto di verità assume una connotazione più dinamica, poiché l'essere vero di un enunciato coincide con il suo essere dimostrabile (non necessariamente in un sistema formale). In altri termini, un enunciato è vero qualora possiamo fornirne, almeno in linea di principio, una giustificazione o una dimostrazione.

Vediamo ora di chiarire con un esempio la corrispondenza di Curry-Howard. Si ricordi la regola di introduzione del connettivo implicazione della deduzione naturale:

$$\frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \supset B}$$

Essa consente di derivare un'implicazione $A \supset B$, ogni qualvolta da una dimostrazione di A possiamo ottenere una dimostrazione di B. Questo può essere anche espresso dicendo che una dimostrazione di $A \supset B$ è una funzione che trasforma una qualsiasi dimostrazione a di A in una dimostrazione $b(a)$ di B. Nella notazione propria del Lambda calcolo una tale funzione viene espressa come $\lambda x.b(x)$ oppure, più precisamente, $\lambda x^A.b(x)^B$, dove x^A esprime il fatto che x è un oggetto di tipo A.[4] Tale λ termine è un elemento del tipo in genere indicato $A \to B$ e rappresentante, da un punto di vista insiemistico, lo spazio di funzioni dall'insieme A all'insieme B.

La corrispondenza associa allora alla proposizione $A \supset B$ l'insieme o il tipo $A \to B$, i cui elementi esprimono dimostrazioni della proposizione $A \supset B$.

L'isomorfismo di Curry e Howard sarà ulteriormente chiarito nel corso della esposizione della teoria dei tipi.

[3] Si veda per esempio [32] per una caratterizzazione della nozione e per indicazioni di carattere storico.

[4] Si è qui utilizzata la notazione del Lambda calcolo. Nella teoria dei tipi $\lambda x^A.b(x)^B$ viene reso con $\lambda x \in A.b(x) \in B$, oppure semplicemente $\lambda(b)$.

2.4. **Teoria dei tipi come linguaggio di programmazione.** La teoria dei tipi di Martin Löf è negli ultimi anni tornata ad essere attivamente oggetto di studio anche grazie alla avvenuta implementazione di essa in software che vanno sotto la denominazione di proof assistants e proof checkers. I sistemi più noti sono NuPRL (R. Constable et al., Cornell University, vd. [7]) e i sistemi della famiglia ALF ([6]), sviluppati dal gruppo logico di Göteborg. Entrambi sono stati utilizzati in "case studies" anche di notevole ampiezza.[5]

Per comprendere meglio la relazione tra teoria dei tipi e informatica si consideri un enunciato del tipo

$$\forall x \in A \exists y \in B\, P(x,y) \tag{1}$$

La corrispondenza di Curry-Howard, associando a ciascuna proposizione un insieme, consente di fatto di passare da una dimostrazione dell'enunciato di forma (1) ad un lambda termine (o programma) nella teoria dei tipi.

Più precisamente, secondo l'interpretazione BHK, una dimostrazione di (1) è una funzione che ad ogni dimostrazione a di A assegna una dimostrazione $b(a)$ di $\exists y \in BP(a,y)$. A sua volta $b(a)$ è una coppia la cui prima componente, d, è un elemento di B (il witness del quantificatore esistenziale) mentre la seconda componente è una dimostrazione del fatto che $P(a,d)$ vale. Possiamo esprimere questo in modo più formale utilizzando la notazione del lambda calcolo, così da ottenere un lambda termine rappresentante la nostra funzione (questo processo è essenzialmente una versione di "realizability"). Uno dei sistemi sopra menzionati puó essere inoltre scelto sia per farci assistere nel dimostrare un particolare enunciato di forma (1), sia per ottenere automaticamente da esso un lambda termine. L'idea è ora di leggere il lambda termine così ottenuto come un programma, di cui (1) è *specificazione*, mentre l'insieme A fornisce l'insieme da cui attingere per gli inputs e l'insieme B costituisce invece la collezione in cui variano gli outputs. Il processo da noi compiuto, prima dimostrando l'enunciato e poi calcolando (con l'aiuto della macchina) un lambda termine (o realizer) per esso, ci consente non solo di estrarre un programma che soddisfi la specificazione data dall'enunciato di partenza, ma che sia anche corretto, ovvero che di

---

[5]NuPRL è abbreviazione di "New Proof Refinement Logic", mentre ALF è acronimo di "Another Logical Framework". Si osservi che attualmente Alfa è il proof assistant in uso a Göteborg, mentre Agda è il proof cheker utilizzato in connessione ad esso. I sistemi possono essere ottenuti attraverso `ftp` dai rispettivi siti internet:
`http://www.cs.chalmers.se/Cs/Research/Logic/`
e `http://www.cs.cornell.edu/Info/Projects/NuPrl/`

Altri noti proof assistants o proof checkers sono: AUTOMATH (N. G. de Brujin et al.), prototipo di proof checker, utilizzato per controllare ampie parti di matematica classica; LCF (L. Paulson) e HOL (M. Gordon); COQ (introdotto presso INRIA) e LEGO (Z. Luo e R. Pollack) entrambi basati sul Calculus of Constructions di T. Coquand e G. Huet [9]; Isabelle (L. Paulson et al.), un proof assistant generico, la cui parte classica ha fino ad oggi conosciuto un maggior sviluppo; Minlog (H. Schwichtenberg et al.), basato sulla logica minimale e specializzato nella estrazione di programmi anche da dimostrazioni classiche (attraverso una versione di A-translation).

fatto soddisfi tale specificazione. Otteniamo infatti non solo un lambda termine ma anche la dimostrazione o *verifica* che esso è un lambda termine che realizza l'enunciato di partenza. Di conseguenza questo processo dovrebbe garantire la produzione di un programma sicuro, che non richieda tutta la serie di checks che sono in genere necessari per eliminare gli usuali errori di programmazione.

3. Giudizi e proposizioni

Veniamo ora ad una esposizione della teoria costruttiva dei tipi.

Essa è formulata attraverso una serie di giudizi che asseriscono proprietà di insiemi o proposizioni. Più precisamente, le regole che definiscono la teoria dei tipi consentono di asserire un nuovo giudizio, in genere a partire da uno o più giudizi (ipotesi).

È importante sottolineare la distinzione tra giudizi e proposizioni. Un giudizio asserisce qualcosa su un insieme o, in virtù della corrispondenza di Curry e Howard, su una proposizione. Per esempio, il giudizio 'A *set*' asserisce che la collezione A è un insieme della teoria, oppure che A è una proposizione. È qui esemplificata una caratteristica della teoria dei tipi: la teoria non è introdotta mediante un formalismo basato sulla logica del primo ordine, in cui le proposizioni sono definite attraverso una (meta) induzione e gli assiomi sono proposizioni specifiche da cui altre conseguono secondo regole. Al contrario qui le proposizioni sono oggetti della teoria e non sono definite induttivamente (mentre i loro elementi lo sono), ma sono intese come concetto aperto, passibile di ampliamento (in altri termini, la collezione delle proposizioni forma un tipo, non un insieme). Inoltre le derivazioni seguono una struttura ad albero, del tutto analoga a quella della deduzione naturale: si parte da assunzioni (che sono qui giudizi) e attraverso le regole del calcolo (che relazionano giudizi e non proposizioni) si ottengono nuovi giudizi.

Martin Löf sottolinea infine che nella teoria dei tipi la semantica non sorge a posteriori, per giustificare le regole sintattiche dopo la loro introduzione, ma è parte integrante della teoria poiché accompagna sempre l'introduzione di concetti e regole, specificandone il senso. Seguendo l'autore, anche noi uniremo all'introduzione delle diverse forme di giudizi una spiegazione informale del loro significato.

3.1. **Giudizi categorici.** Introduciamo ora prima di tutto le diverse forme di giudizio categorico, ovvero di quel genere di giudizio che non dipende da ipotesi. Nella prossima sottosezione vedremo invece una notazione per giudizi ipotetici, dipendenti da una o più ipotesi.

Vi sono 4 forme fondamentali di giudizi, qui espressi nella forma insiemistica:

(a) A *set*, ovvero A è un insieme;
(b) $A = B$, ossia A e B sono insiemi uguali;
(c) $a \in A$, ovvero a è un elemento dell'insieme A;

(d) $a = b \in A$, ossia a e b sono elementi uguali dell'insieme A.[6]

Questi giudizi fondamentali possono alternativamente essere visti come asserzioni su proposizioni; per es il primo e il terzo giudizio possono essere letti come segue: A è una proposizione; a è una dimostrazione della proposizione A. In questo caso, il giudizio A *set* verrà anche scritto A *prop*.

Intuizionisticamente la verità di una proposizione coincide con il suo essere dimostrabile (anche in modo informale); ossia una proposizione è vera se esiste almeno una dimostrazione di essa, cioé se noi possiamo almeno in linea di principio dimostrarla. Data la corrispondenza tra insiemi e proposizioni, asserire che una proposizione è vera coincide con il dichiarare che l'insieme delle sue dimostrazioni è abitato.[7] Inoltre il concetto di esistenza viene preso in modo radicale e serio dal costruttivista: asserire l'esistenza di una entità matematica significa poterne produrre un esempio. Quindi per asserire che una proposizione è vera (ovvero che l'insieme delle sue dimostrazioni è abitato) se ne deve produrre una dimostrazione (rispettivamente un elemento). Di conseguenza il giudizio A *true*, che asserisce la verità della proposizione A, corrisponde ad un giudizio del terzo tipo, che asserisce che un certo oggetto, per es a, è elemento dell'insieme A. Unica differenza è che in un giudizio della forma A *true* si omette volontariamente l'informazione su quale sia il witness della verità di A. La validià del giudizio però dipende sempre dalla possibilità di recuperare in qualsiasi momento questa informazione, presentando una dimostrazione di A.

Vi è un'ulteriore lettura dei giudizi categorici sopra presentati. Martin Löf suggerisce infatti di interpretare una proposizione come un problema da risolvere, o un compito da assolvere. In questo caso, $a \in A$ viene letto come: a è un metodo per risolvere il problema A, oppure per assolvere il compito A.

Veniamo ora alle spiegazioni informali che accompagnano l'introduzione delle 4 forme di giudizi categorici.

(a) La prima forma di giudizio permette di asserire che cosa costituisca un insieme della teoria. Essa risponde alla domanda: "che cosa è un insieme?"

In accordo con la concezione di Bishop, un insieme è determinato dai suoi elementi; non solo, ma con ogni insieme è data contemporaneamente anche una relazione di equivalenza che stabilisce quali dei suoi elementi sono uguali (ovvero un insieme è il quoziente rispetto a tale relazione di equivalenza di una certa collezione di elementi). La spiegazione è quindi la seguente:

Definiamo un insieme A prescrivendo:

(i) come formare un elemento canonico di A,
(ii) come formare due elementi canonici uguali di A.

---

[6]Si noti ancora una volta che, contrariamente alla teoria assiomatica degli insiemi, $a \in A$ e $A = B$ sono qui giudizi e non proposizioni.

[7]Si osservi che in ambito costruttivo si preferisce in genere utilizzare caratterizzazioni positive, per cui un insieme non vuoto viene detto insieme *abitato*.

Esempio caratteristico è quello dell'insieme dei numeri naturali. $\mathbf{N}$ è definito attraverso le regole:

$$0 \in \mathbf{N} \qquad\qquad 0 = 0 \in \mathbf{N}$$

$$\frac{a \in \mathbf{N}}{suc(a) \in \mathbf{N}} \qquad\qquad \frac{a = b \in \mathbf{N}}{suc(a) = suc(b) \in \mathbf{N}}$$

Queste regole consentono di formare i cosiddetti *elementi canonici* di $\mathbf{N}$, ovvero lo 0 e tutti gli elementi della forma $suc(a)$ per qualche elemento a di $\mathbf{N}$, e stabiliscono inoltre quando due elementi sono uguali.

Si osservi che in genere un insieme contiene altri elementi oltre a quelli canonici. Nel caso di $\mathbf{N}$ gli elementi non canonici sono quelli che possono essere ricondotti o allo 0 o alla forma $suc(a)$ (per $a \in \mathbf{N}$) mediante una computazione o un processo di riduzione. Ad esempio, il termine $2 + 2$ rappresenta un elemento non canonico di $\mathbf{N}$, la sua appartenenza ad $\mathbf{N}$ essendo giustificata dal fatto che l'applicazione delle regole di computazione che definiscono il simbolo $+$ produce un elemento di $\mathbf{N}$.[8]

(b) Il secondo tipo di giudizio, presuppone sia noto che A e B sono insiemi (quindi presuppone due giudizi del primo tipo), e asserisce che $A = B$. Questo si verifica qualora A e B siano estensionalmente uguali, cioè se ogniqualvolta si possa asserire che $a \in A$, si possa anche asserire che $a \in B$, e viceversa. Inoltre, pioché un insieme è non solo determinato dai suoi elementi ma porta con sé sempre informazione sulla relazione di uguaglianza tra i suoi elementi, per poter stabilire che $A = B$, dobbiamo anche poter appurare che dati due elementi a, b di A, se $a = b \in A$ allora anche $a = b \in B$, e viceversa. Questo viene espresso in modo più compatto come segue:

$A = B$ *se e solo se*

$$\frac{a \in A}{\overline{a \in B}} \qquad\qquad \frac{a = b \in A}{\overline{a = b \in B}}$$

Si noti la linea doppia, che indica la possibilità di passare dal giudizio sopra la linea a quello sotto *e viceversa*, per esempio nel primo caso si asserisce che $a \in B$ è derivabile da $a \in A$ ma anche $a \in A$ da $a \in B$.

(c) Il terzo tipo di giudizio consente di asserire quando un oggetto è elemento di un insieme. Martin Löf lo spiega in modo piuttosto suggestivo,

[8]Si osservi che le regole di computazione che consentono di passare da un elemento qualsiasi ad uno canonico seguono la cosiddetta *lazy evaluation*, ovvero il processo di calcolo si ferma non appena la componente esterna del termine abbia forma di elemento canonico. Questo in molti casi non richiede di calcolare tutti i sottotermini del termine dato. Per esempio, nel caso appena ricordato, $2 + 2$ è riconosciuto come numero naturale poiché computando $2 + 2$ si ottiene il valore $suc(2 + 1)$ e $2 + 1 \in \mathbf{N}$.

ricorrendo a una delle letture alternative della nozione di insieme. Egli scrive:

Un elemento, a, di un insieme A è un metodo (o programma) che quando eseguito produce un elemento canonico di A come risultato.

(d) Giungiamo così all'ultimo tipo di giudizio, che viene spiegato in modo analogo al terzo, ricorrendo nuovamente all'interpretazione computazionale:

Due elementi, a e b, di un insieme A sono uguali se, una volta eseguiti, producono uguali elementi canonici di A come risultato.

3.2. **Giudizi ipotetici.** Le forme di giudizio sopra presentate sono indipendenti da assunzioni. Qualora, però si intenda rappresentare l'attività matematica per mezzo della teoria dei tipi, occorre essere in grado di esprimere anche giudizi che dipendano dall'assunzione di una o più ipotesi. L'analogia con derivazioni dipendenti da premesse nella deduzione naturale è evidente, si pensi per esempio alla regola di introduzione dell'implicazione.

Prima di passare ad una generalizzazione induttiva ad n premesse delle 4 forme fondamentali di giudizio appena introdotte, soffermiamoci sul caso specifico di un giudizio ipotetico del primo tipo dipendente da una sola ipotesi. La sua forma è:

$$B(x) \ set \ (x \in A),$$

e si asserisce con esso che $B(x)$ è un insieme sotto l'ipotesi $x \in A$. In tal caso diremo anche che $B(x)$ è una *famiglia di insiemi dipendente da A*. Il giudizio afferma che $B(a)$ è un insieme ogni qualvolta a è un elemento di A, e che se a e b sono elementi uguali di A, $B(a)$ e $B(b)$ sono uguali.

Si noti che nel seguito per ragioni tipografiche faremo uso di due distinte notazioni per giudizi ipotetici; alla notazione lineare del tipo:

$$B(x) \ set \ (x \in A)$$

potrà essere sostituita (soprattutto nella redazione delle regole) la notazione:

$$\begin{gathered}(x \in A)\\ B(x) \ set\end{gathered}$$

Veniamo dunque ora alla generalizzazione ad n premesse delle 4 forme di giudizio.

(i) Assumiamo prima di tutto i seguenti giudizi:

$$A_1 \ set,$$

$$A_2(x_1) \ set \quad (x_1 \in A_1),$$

$$A_3(x_1, x_2) \ set \quad (x_1 \in A_1, x_2 \in A_2(x_1)),$$

etc.

$$A(x_1, \ldots, x_{n-1}) \ set$$

sotto l'ipotesi

$$(x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_{n-1} \in A_n(x_1, \ldots, x_{n-2})).$$

Possiamo allora formulare il seguente giudizio:

$$A(x_1, \ldots, x_n) \ set$$

sotto l'ipotesi:

$$(x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_n \in A_n(x_1, \ldots, x_{n-1})).$$

Come visto sopra, $A_2(x_1)$ *set* $(x_1 \in A_1)$ esprime il fatto che $A_2(x_1)$ è un insieme sotto l'ipotesi $x_1 \in A_1$. Un tale giudizio significa che $A_2(a)$ è un insieme ogniqualvolta $a \in A_1$ e $A_2(a) = A_2(b)$ se $a = b \in A_1$. Similmente, $A_3(x_1, x_2)$ *set* $(x_1 \in A_1, x_2 \in A_2(x_1))$ esprime il fatto che $A_3(x_1, x_2)$ è una famiglia di insiemi dipendente dagli insiemi A_1 e $A_2(x_1)$ (quest'ultimo essendo a sua volta una famiglia di insiemi dipendente da A_1). Gli altri giudizi si motivano in modo analogo.

Le n assunzioni in un giudizio come sopra conformato costituiscono il cosiddetto *contesto*, spesso indicato nella letteratura con le lettere Δ, Γ. Convenzione implicita è che A_i *può* (ma non *deve*) dipendere da x_j con $j < i$, ma non dipende da x_i, e così anche A *può* dipendere da x_i ma non da x.

Similmente a quanto avviene nella deduzione naturale, anche nella teoria dei tipi alcune regole consentono di scaricare assunzioni (un'assunzione viene scaricata se compare nell'ipotesi ma non nella conclusione di una regola). Naturalmente nel caso in cui si operi con un contesto piuttosto complesso va posta attenzione nello scaricare le assunzioni in ordine giusto, senza violare in alcun modo le dipendenze.

Dovrebbe essere chiaro come giustificare i seguenti giudizi ipotetici (ora riportati nel secondo tipo di notazione):

(ii)

$$(x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$$

$$A(x_1, \ldots, x_n) = B(x_1, \ldots, x_n).$$

(iii)

$$(x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$$

$$a(x_1, \ldots, x_n) \in A(x_1, \ldots, x_n).$$

(iv)

$$(x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$$

$$a(x_1, \ldots, x_n) = b(x_1, \ldots, x_n) \in A(x_1, \ldots, x_n).$$

4. Regole di assunzione, uguaglianza, sostituzione

4.1. **Convenzioni.** Possiamo ora riepilogare le regole della teoria costruttiva dei tipi. È opportuno però precisare che Martin Löf, nel presentare la teoria, fa implicitamente uso di alcune convenzioni per eliminare l'eccesso di informazione che renderebbe difficile la comprensione delle regole. Spetta al lettore il compito di ricostruire di volta in volta la forma completa di ciascuna regola. Sono adottate in particolare le seguenti convenzioni:

(i) Non viene trascritta un'assunzione che compare sia nella premessa che nella conclusione di una regola.
(ii) Non si richiede che un'assunzione scaricata figuri realmente nella derivazione della conclusione.
(iii) Nel caso delle regole che hanno conclusione della forma: $a \in A$ oppure $a = b \in A$, vengono esplicitamente riportate solo le premesse aventi la stessa forma.

4.2. **Regola di assunzione.** Vi è una sola regola che introduce nuove assunzioni rispetto a quelle presenti nelle ipotesi: la regola di assunzione. Nella sua forma completa essa si presenta come segue:

$$\frac{A\ set \quad (x_1 \in A_1, \ldots, x_n \in A_n)}{x \in A \quad (x_1 \in A_1, \ldots, x_n \in A_n, x \in A)}$$

4.3. **Regole di uguaglianza.** Le regole di ugualianza sono formulate per elementi e per insiemi ed esprimono le usuali proprietà di una relazione di equivalenza, ovvero *riflessività, simmetria* e *transitività.* Nel caso degli insiemi si postula inoltre una regola secondo la quale insiemi uguali hanno gli stessi elementi.

Riteniamo che queste regole siano particolarmente chiare, per cui non verrà per esse data alcuna spiegazione (per chiarimenti si veda Martin Löf [23]).

Riflessività:

$$\frac{a \in A}{a = a \in A} \qquad\qquad \frac{A\ set}{A = A}$$

Simmetria:

$$\frac{a = b \in A}{b = a \in A} \qquad\qquad \frac{A = B}{B = A}$$

Transitività:

$$\frac{a = b \in A \qquad b = c \in A}{a = c \in A} \qquad\qquad \frac{A = B \qquad B = C}{A = C}$$

Regola di Uguaglianza per insiemi:

$$\frac{a \in A \qquad A = B}{a \in B} \qquad\qquad \frac{a = b \in A \qquad A = B}{a = b \in B}$$

4.4. **Regole di sostituzione.** Le seguenti regole di sostituzione sono giustificate dal significato dei giudizi introdotti in Sezione 3.2. Esse consentono di operare sostituzioni per variabili in insiemi e in elementi.

Sostituzione in insiemi:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ \\ B(x)\ set\end{array}}{B(a)\ set} \qquad\qquad \frac{a = c \in A \qquad \begin{array}{c}(x \in A)\\ \\ B(x)\ set\end{array}}{B(a) = B(c)}$$

Sostituzione in insiemi uguali:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ \\ B(x) = D(x)\end{array}}{B(a) = D(a)}$$

Sostituzione in elementi:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ \\ b(x) \in B(x)\end{array}}{b(a) \in B(a)} \qquad\qquad \frac{a = c \in A \qquad \begin{array}{c}(x \in A)\\ \\ b(x) \in B(x)\end{array}}{b(a) = b(c) \in B(a)}$$

Sostituzione in elementi uguali:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ \\ b(x) = d(x) \in B(x)\end{array}}{b(a) = d(a) \in B(a)}$$

4.5. **Nota sul concetto di uguaglianza.** Nella teoria dei tipi vi sono 3 distinte nozioni di uguaglianza.

(1) *Judgemental equality.* Abbiamo già trattato della nozione di uguaglianza che compare nei giudizi. Questa è duplice: asseriamo che
   (i) due insiemi sono uguali oppure
   (ii) due elementi di un insieme sono uguali.
(2) *Definitional equality.* L'uguaglianza definitoria, qui indicata con il simbolo $:=$, viene utilizzata per stipulare convenzioni sintattiche oppure regole di riscrittura (rewriting).
(3) *Propositional equality.* Un'ulteriore nozione di uguaglianza consente di riflettere a livello di proposizioni l'uguaglianza dei giudizi. Nella sezione 5.7, introdurremo infatti un insieme $\mathbf{I}(A, a, b)$ che è abitato qualora il giudizio $a = b \in A$ sia valido.[9]

5. Regole insiemistiche

5.1. **Tipi di regole.** Ci occuperemo ora delle regole che consentono di introdurre insiemi e di formare nuovi insiemi da insiemi dati. Definiremo anzitutto l'insieme dei numeri naturali e quello degli insiemi finiti; ci assicureremo così immediatamente che la nostra teoria si occupa di oggetti significativi (e non è vuota). Passeremo quindi a definire particolari forme di insieme: il prodotto, la somma etc. ottenibili da insiemi dati mediante applicazione degli operatori insiemistici Π, Σ, $+$, $\mathbf{I}$. Le regole che introdurremo hanno una struttura comune; sono infatti raggruppabili in 4 tipi fondamentali:

(i) Formazione
(ii) Introduzione
(iii) Eliminazione
(iv) Uguaglianza

(i) Il primo tipo di regola indica in quali circostanze si possa formare un nuovo insieme (in genere a partire da insiemi dati o da famiglie di insiemi). Stabilisce inoltre quando si possono formare due insiemi uguali. Alternativamente queste regole esprimono le condizioni per ottenere proposizioni nuove a partire da proposizioni o da funzioni proposizionali date.

(ii) Il secondo tipo esprime quali siano gli elementi canonici di un insieme e quando essi siano uguali. Come visto sopra, un insieme è definito stabilendo quali sono i suoi elementi canonici e fissando una relazione di equivalenza tra elementi canonici. Sono dunque queste le regole che assegnano significato a ciascun nuovo insieme. In tali regole si introducono in genere delle nuove costanti di base (per esempio 0) e/o dei simboli per funzioni, i cosiddetti costruttori (per esempio *suc*).

[9] Si osservi che le modalità di definizione dell'uguaglianza proposizionale influenzano proprietà fondamentali come quella di decidibilità della relazione di appartenenza della teoria. Se ne comprende pertanto il notevole impatto su applicazioni informatiche.

(iii) Il terzo tipo di regola consente in genere di definire funzioni aventi come dominio l'insieme in considerazione e permette quindi di utilizzarne gli elementi. Le regole di eliminazione costituiscono spesso una forma di induzione strutturale: per stabilire che una qualche proprietà vale di un elemento qualsiasi di un insieme è sufficiente dimostrare che vale di un elemento canonico di esso. Questo genere di regola introduce infatti una nuova costante di eliminazione (o distruttore) che, applicata ad argomenti rappresentanti la validità dell'ipotesi induttiva, consente di ottenere dimostrazioni di una proposizione esprimente una proprietà di un elemento arbitrario dell'insieme.

(iv) L'ultimo tipo pone in relazione le regole di introduzione ed eliminazione, mostrando come una funzione introdotta mediante una regola di eliminazione operi sugli elementi canonici di un insieme.

5.2. **Numeri Naturali.** L'esistenza di un insieme dei numeri naturali é raramente posta in dubbio da matematici, logici e filosofi della matematica. Anche i sostenitori del predicativismo più estremo assumono in genere come dati i numeri naturali. L'esistenza di una collezione di numeri naturali è in particolare accolta dall'intuizionismo. Secondo uno dei principi fondamentali di questa scuola di pensiero la matematica si occupa di costrutti mentali aventi come oggetto l'infinito. Esempio paradigmatico è proprio la collezione dei numeri naturali. Questo aver a che fare con l'infinito, che caratterizza e definisce l'attività matematica, richiede e giustifica l'allontanamento dalla logica classica, che si costituisce e opera invece efficacemente solo sul finito.

Si osservi inoltre che i numeri naturali rappresentano un fondamentale esempio di datatype in molti linguaggi di programmazione.

Nella teoria dei tipi, dunque, si assume l'esistenza di un insieme dei numeri naturali, $\mathbf{N}$, definito induttivamente come segue: si stabilisce che 0 è un numero naturale e che per ogni numero naturale, a, il suo successore, $suc(a)$, è anch'esso un numerto naturale. Simultaneamente si stabilisce quando due elementi di $\mathbf{N}$ sono uguali.

Dato l'insieme dei naturali, lo vogliamo ovviamente utilizzare, per esempio, per effettuare dei calcoli aritmetici: desideriamo definire su di esso le ordinarie funzioni di addizione, sottrazione, etc. A questo scopo introduciamo nella regola di eliminazione una nuova costante, $\mathbf{R}$, e definiamo uniformemente per suo mezzo tutte le funzioni primitive recursive. Il comportamento di questo "distruttore" verrà chiarito solo dopo aver presentato le regole.

Si fa presente che riporteremo qui le regole di $\mathbf{N}$ al completo, ovvero anche le corrispondenti regole di uguaglianza. Nel seguito, nel presentare gli altri tipi di insiemi, queste ultime saranno in genere omesse per brevità.[10]

[10]Si veda per esempio Beeson [4] per un elenco completo di regole.

(i) **N**-Formazione

$$\mathbf{N}\ set \qquad\qquad \mathbf{N} = \mathbf{N}$$

(ii) **N**-Introduzione

$$0 \in \mathbf{N} \qquad\qquad 0 = 0 \in \mathbf{N}$$

$$\frac{a \in \mathbf{N}}{suc(a) \in \mathbf{N}} \qquad\qquad \frac{a = b \in \mathbf{N}}{suc(a) = suc(b) \in \mathbf{N}}$$

(iii) **N**-Eliminazione

$$\frac{c \in \mathbf{N} \qquad d \in C(0) \qquad \begin{array}{c}(x \in \mathbf{N},\ y \in C(x))\\ e(x,y) \in C(suc(x))\end{array}}{\mathbf{R}(c,d,e) \in C(c)}$$

$$\frac{c = f \in \mathbf{N} \qquad d = g \in C(0) \qquad \begin{array}{c}(x \in \mathbf{N},\ y \in C(x))\\ e(x,y) = h(x,y) \in C(suc(x))\end{array}}{\mathbf{R}(c,d,e) = \mathbf{R}(f,g,h) \in C(c)}$$

(iv) **N**-Uguaglianza

$$\frac{d \in C(0) \qquad \begin{array}{c}(x \in \mathbf{N},\ y \in C(x))\\ e(x,y) \in C(suc(x))\end{array}}{\mathbf{R}(0,d,e) = d \in C(0)}$$

$$\frac{a \in \mathbf{N} \qquad d \in C(0) \qquad \begin{array}{c}(x \in \mathbf{N},\ y \in C(x))\\ e(x,y) \in C(suc(x))\end{array}}{\mathbf{R}(suc(a),d,e) = e(a,\mathbf{R}(a,d,e)) \in C(suc(a))}$$

Nelle ultime due regole $C(z)$ è una famiglia di insiemi dipendente da $\mathbf{N}$.

L'unica regola che richiede spiegazioni è la regola di eliminazione.

Dato un elemento arbitrario (ovvero non necessariamente canonico) c di $\mathbf{N}$, possiamo leggere $C(c)$ come una proposizione, e pensare di volerne fornire una dimostrazione. La regola consente di dimostrare $C(c)$ per induzione, ovvero fornendo dapprima una dimostrazione d di $C(0)$ e dando poi una dimostrazione $e(x,y)$ di $C(suc(x))$ qualora sia stato appurato che per $x \in \mathbf{N}$, y è una dimostrazione di $C(x)$. Secondo la regola, possiamo allora costruire una dimostrazione, $\mathbf{R}(c, d, e)$ (dipendente da c, d ed e), della proposizione $C(c)$.

$\mathbf{R}(c, d, e)$ viene calcolato così:

(1) si prenda un elemento arbitrario, c, di $\mathbf{N}$ e se ne calcoli il valore canonico;
(2) se la computazione dà come risultato $c = 0 \in \mathbf{N}$, allora si calcoli $d \in C(0)$, ottenendo così un elemento canonico f di $C(0)$. Si noti che $c = 0 \in \mathbf{N}$, per cui $C(c) = C(0)$ e dunque f sarà elemento canonico anche di $C(c)$;
(3) se invece la computazione produce un elemento di $\mathbf{N}$ della forma $suc(a)$ per $a \in \mathbf{N}$, si proceda come segue: si sostituisca a ad x e $\mathbf{R}(a, d, e)$ ad y, ottenendo $e(a, \mathbf{R}(a, d, e)) \in C(suc(a))$. Si osservi che $C(suc(a)) = C(c)$, per cui $e(a, \mathbf{R}(a, d, e)) \in C(c)$. Si calcoli quest'ultimo, ottenendo un elemento canonico, g, di $C(c)$;
(4) se a risulta avere come valore 0, allora $\mathbf{R}(a, d, e) \in C(c)$ per (2); altrimenti si proceda nuovamente come in (3).

5.2.1. *Esercizio.* Si definiscano le seguenti funzioni:

$$pred(x) := \mathbf{R}(x, 0, (x, y).x),$$

$$+(x, y) := \mathbf{R}(x, y, (z, w).suc(w)),$$

$$*(x, y) := \mathbf{R}(x, 0, (z, w). + (y, w)).$$

Si osservi che la notazione $(x, y).t$ indica che x, y sono le variabili di t su cui si può compiere sostituzione. L'uso delle parentesi corrisponde essenzialmente alla λ notazione. La scelta notazionale è dovuta alla necessità di non confondere questa astrazione con l'uso della costante λ nella teoria dei tipi, costante primitiva del sistema che consente di ottenere elementi canonici del prodotto cartesiano (quindi funzioni in quanto oggetti della teoria).

Si lascia come esercizio la verifica che queste operazioni su $\mathbf{N}$ definiscono rispettivamente: predecessore, addizione e moltiplicazione.

5.3. **Insiemi Finiti.** Vogliamo ora introdurre insiemi finiti di oggetti. Per ciascun numero naturale k formeremo dunque l'insieme $\mathbf{N_k}$, composto da k elementi (nella usuale notazione insiemistica, l'insieme intuitivamente rappresenta $\{0, 1, \ldots, k-1\}$). Si noti che gli oggetti di $\mathbf{N_k}$ non coincidono con i primi k elementi di $\mathbf{N}$: per esempio, 0 è da intendersi oggetto distinto da 0_k.

Per ciascun numero naturale k, vi sono le 4 regole di $\mathbf{N_k}$-formazione, $\mathbf{N_k}$-introduzione, $\mathbf{N_k}$-eliminazione e $\mathbf{N_k}$-uguaglianza, con rispettive regole di uguaglianza.[11] Si fa presente che per motivi di brevità, in questo contesto sono state omesse le regole di uguaglianza.

(i) $\mathbf{N_k}$-Formazione

$$\mathbf{N_k}\ set$$

(ii) $\mathbf{N_k}$-Introduzione

$$m_k \in \mathbf{N_k} \qquad (m = 0, 1, \ldots, k-1)$$

(iii) $\mathbf{N_k}$-Eliminazione

$$\frac{c \in \mathbf{N_k} \qquad c_m \in C(m_k) \qquad (m = 0, 1, \ldots, k-1)}{\mathbf{R_k}(c, c_0, \ldots, c_{k-1}) \in C(c)}$$

(iv) $\mathbf{N_k}$-Uguaglianza

$$\frac{c_m \in C(m_k) \qquad (m = 0, 1, \ldots, k-1)}{\mathbf{R_k}(m_k, c_0, \ldots, c_{k-1}) = c_m \in C(m_k)}$$

Ancora una volta $C(x)\,(x \in \mathbf{N_k})$ è una famiglia di insiemi, e può essere letta come una funzione proposizionale, asserente una proprietà di elementi di $\mathbf{N_k}$. $\mathbf{R_k}$ rappresenta una forma di definizione per casi.

Spiegazione di $\mathbf{R_k}$:

(1) si esegua c, ottenendo un elemento canonico, m_k, di $\mathbf{N_k}$ come risultato $(m = 0, 1, \ldots, k-1)$;
(2) si selezioni l'elemento corrispondente c_m di $C(m_k)$ e lo si esegua. Si otterrà come risultato un elemento canonico $d \in C(m_k)$. Essendo però $c = m_k \in \mathbf{N_k}$ si ha $C(c) = C(m_k)$ e dunque $d \in C(c)$.

Si noti che l'insieme $\mathbf{N_0}$ corrisponde all'insieme vuoto, non possedendo alcun elemento. La regola di introduzione è nel suo caso vacua. La regola di eliminazione diviene invece:

$\mathbf{N}_0$-Eliminazione

$$\frac{c \in \mathbf{N}_0}{\mathbf{R}_0(c) \in C(c)}$$

L'insieme $\mathbf{N}_1$ ha invece un solo elemento, 0_1.[12]

[11] La presenza di infinite regole può in alcuni casi essere problematica; ad esempio qualora alla teoria dei tipi si aggiunga un tipo universo e si desideri avere una regola di eliminazione per esso. In tal caso si può optare per l'introduzione di due soli insiemi finiti, $\mathbf{N}_0$ e $\mathbf{N}_1$, utilizzando ripetutamente la somma disgiunta, $+$, per produrre un qualsiasi insieme finito. In genere si preferisce assumere $\mathbf{N_k}$ come primitivo per semplicità notazionale e concettuale.

[12] Questo è dimostrabile, ma richiede concetti al momento non ancora presentati. Si veda Esercizio 5.7.1 (i).

5.3.1. *Vero e Falso, Booleani.* Qualora si adotti una prospettiva proposizionale, l'insieme $\mathbf{N}_0$ viene a coincidere con il falso, non possedendo alcun elemento (ovvero dimostrazione) e viene anche indicato con $\perp$. L'insieme $\mathbf{N}_1$ coincide con il vero, essendo sempre abitato dall'elemento canonico 0_1. L'insieme $\mathbf{N}_2$, i cui elementi sono 0_2 e 1_2, rappresenta invece l'insieme dei Booleani utilizzato in numerosi linguaggi di programmazione.

Quando si interpreta $\mathbf{N}_0$ con il falso, la regola di eliminazione diviene la nota regola *ex falso quodlibet*:

$\perp$-Eliminazione

$$\frac{\perp \ true}{C \ true}$$

Si osservi come in questo caso si possa omettere l'informazione circa il witness della verità sia nella premessa che nella conclusione della regola.

5.3.2. *Esercizio.* È possibile definire il tipico costrutto "if c then d else f", utilizzato in numerosi linguaggi di programmazione, con $\mathbf{R}_2(c, d, f)$. Si lascia la verifica come esercizio.

Le regole insiemistiche fino ad ora presentate consentono di introdurre insiemi nuovi come $\mathbf{N}$, oppure $\mathbf{N_k}$ (per qualche k). Le regole sotto riportate assumono invece l'esistenza di un insieme o di una famiglia di insiemi dati, e consentono di formare un nuovo insieme a partire da questi. Sintatticamente le regole di formazione sono dunque regole con premesse.

5.4. **Prodotto (cartesiano) generalizzato.** Dati un insieme A e una famiglia di insiemi $B(x)\,(x \in A)$ ci proponiamo di formare un insieme i cui elementi rappresentino funzioni con argomenti in A e valori nella famiglia di insiemi. Il concetto di funzione qui rappresentato è più generale di quello ordinario di funzione da un insieme A ad un insieme B. Una funzione farà corrispondere ad un elemento a di A un elemento $b(a)$ di un insieme, $B(a)$, che dipende da a. Il codominio non sarà dunque un insieme prefissato, ma dipenderà dall'argomento della funzione. Quello ora riportato è il primo esempio di tipi dipendenti che incontriamo nelle teoria dei tipi (un altro esempio importante è la somma generalizzata). La presenza di tipi dipendenti dà notevole forza e capacità espressiva alla teoria, senza per questo richiedere l'utilizzo di elementi impredicativi. In particolare, il prodotto generalizzato consente di esprimere il quantificatore universale, così come la somma generalizzata consente di esprimere il quantificatore esistenziale.

Gli elementi canonici di $\Pi(A, B)$ sono funzioni ottenute per λ astrazione: da un elemento $b(x)$ di $B(x)$, si astrae b e ad esso si applica λ ottenendo $\lambda(b)$, elemento canonico del prodotto. L'idea di fondo è che $b(x) \in B(x)(x \in A)$ sia una funzione di cui $\lambda(b)$ è un nome o rappresentante.

L'uguaglianza di due elementi canonici del prodotto è derivata dall'uguaglianza nella famiglia di insiemi $B(x)(x \in A)$. Si ha infatti che due elementi canonici $\lambda(b)$ e $\lambda(d)$ sono uguali se per $x \in A$, si ha $b(x) = d(x) \in B(x)$

(regola di uguaglianza di Π-introduzione). La nozione di funzione è dunque *estensionale.*[13]

Indichiamo con $\Pi(A, B)$ oppure $(\Pi x \in A)B(x)$ il prodotto di A con la famiglia B dipendente da A, e con $\lambda(b)$ gli elementi canonici del prodotto. La spiegazione della costante **Ap** di eliminazione verrà data dopo aver presentato le regole.

Per completezza, nel caso del prodotto riportiamo anche le regole di uguaglianza.

(i) Π - Formazione

$$\frac{A\ set \qquad \begin{matrix}(x \in A)\\ \\ B(x)\ set\end{matrix}}{\Pi(A, B)\ set} \qquad\qquad \frac{A = C \qquad \begin{matrix}(x \in A)\\ \\ B(x) = D(x)\end{matrix}}{\Pi(A, B) = \Pi(C, D)}$$

(ii) Π - Introduzione

$$\frac{\begin{matrix}(x \in A)\\ \\ b(x) \in B(x)\end{matrix}}{\lambda(b) \in \Pi(A, B)} \qquad\qquad \frac{\begin{matrix}(x \in A)\\ \\ b(x) = d(x) \in B(x)\end{matrix}}{\lambda(b) = \lambda(d) \in \Pi(A, B)}$$

(iii) Π - Eliminazione

$$\frac{c \in \Pi(A, B) \qquad a \in A}{\mathbf{Ap}(c, a) \in B(a)} \qquad\qquad \frac{c = f \in \Pi(A, B) \qquad a = d \in A}{\mathbf{Ap}(c, a) = \mathbf{Ap}(f, d) \in B(a)}$$

(iv) Π - Uguaglianza

$$\frac{a \in A \qquad \begin{matrix}(x \in A)\\ \\ b(x) \in B(x)\end{matrix}}{\mathbf{Ap}(\lambda(b), a) = b(a) \in B(a)} \qquad\qquad \frac{c \in \Pi(A, B)}{(\lambda x)\mathbf{Ap}(c, x) = c \in \Pi(A, B)}$$

Spiegazione di **Ap**:

$\mathbf{Ap}(c, a)$ è un metodo che consente di ottenere elementi canonici di $B(a)$. Viene utilizzato nel modo seguente:

(1) si calcoli $c \in \Pi(A, B)$, ottenendo come valore un elemento canonico $\lambda(b)$ per $b(x) \in B(x)(x \in A)$;
(2) si sostituisca $a \in A$ ad x in b. Si ottiene così $b(a) \in B(a)$;
(3) si calcoli $b(a)$, ottenendo un elemento canonico di $B(a)$.

[13] Si veda la sezione 5.7 per un accenno alla distinzione tra versioni estensionali e intensionali della teoria.

La regola di eliminazione costituisce chiaramente un analogo della β riduzione del λ calcolo. In alternativa a questa regola se ne potrebbe assumere una simile alle regole di eliminazione per gli altri operatori, ovvero un'eliminazione che rappresenti un principio di induzione strutturale. Si veda per esempio [25] pagine 51 e 52.

5.4.1. *Quantificatore Universale.* Le regole possono essere anche lette in chiave proposizionale e costituiscono in questo caso regole per il quantificatore universale.

$\forall$ - Formazione

$$\frac{A\ set \qquad \begin{matrix}(x \in A)\\ \\ B(x)\ prop\end{matrix}}{(\forall x \in A)B(x)\ prop}$$

Si noti che il giudizio A *set* resta immutato nella traduzione proposizionale, essendo A l'insieme dominio di quantificazione.

$\forall$ - Introduzione

$$\frac{\begin{matrix}(x \in A)\\ \\ B(x)\ true\end{matrix}}{(\forall x \in A)B(x)\ true}$$

$\forall$ - Eliminazione

$$\frac{(\forall x \in A)B(x)\ true \qquad a \in A}{B(a)\ true}$$

Dato che nella 'traduzione' in termini proposizionali si perde traccia dei witnesses, non si ha qui una regola di uguaglianza.

5.4.2. *Spazio di funzioni e implicazione.* Otteniamo un caso particolare del prodotto $\Pi(A, B)$, quando la famiglia B non dipenda dall'insieme A. Il prodotto rappresenta allora lo spazio di funzioni da un insieme A ad un insieme B, e viene indicato con $A \to B$, oppure con B^A. Si osservi che a differenza della teoria assiomatica degli insiemi, nella teoria dei tipi le funzioni non sono rappresentate da insiemi di coppie ordinate, ma costituiscono un concetto primitivo.

In chiave proposizionale, lo spazio di funzioni $A \to B$ corrisponde all'implicazione $A \supset B$, di cui riportiamo qui le regole derivate.

$\supset$ - Formazione

$$\dfrac{A\ prop \qquad \begin{matrix}(A\ true)\\ \\ B\ prop\end{matrix}}{A \supset B\ prop}$$

$\supset$ - Introduzione

$$\dfrac{\begin{matrix}(A\ true)\\ \\ B\ true\end{matrix}}{A \supset B\ true}$$

$\supset$ - eliminazione

$$\dfrac{A \supset B\ true \qquad A\ true}{B\ true}$$

Osserviamo che le regole di introduzione ed eliminazione di $\supset$ corrispondono esattamente alle rispettive regole di deduzione naturale. La regola di formazione si caratterizza per la particolarità di permettere l'utilizzo dell'assunzione $A\ true$ per dimostrare che B è una proposizione.

5.4.3. *Negazione.* Nel calcolo proposizionale intuizionistico il concetto di negazione è ottenuto mediante implicazione e falso. Anche qui si definisce: $\neg A := A \supset \bot$.

5.4.4. *Esercizi.* Si giustifichino i seguenti giudizi:

(i) $A \supset (B \supset A)\ true$.
(ii) $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))\ true$.
(iii) $(A \supset B) \supset ((B \supset C) \supset (A \supset C))\ true$.

Si osservi che i combinatori

$$\mathbf{K} := \lambda(x)\lambda(y).x$$

ed

$$\mathbf{S} := \lambda(x)\lambda(y)\lambda(z).\mathbf{Ap}(\mathbf{Ap}(x,z), \mathbf{Ap}(y,z))$$

rappresentano dimostrazioni rispettivamente della prima e della seconda proposizione.

5.5. **Somma generalizzata.** Dati un insieme A e una famiglia $B(x)(x \in A)$ dipendente da A, vogliamo formare un insieme i cui elementi siano coppie, la prima componente delle quali sia elemento di A e la seconda elemento di un insieme della famiglia B. Si desidera in altri termini definire una generalizzazione del prodotto cartesiano di due insiemi al caso di una famiglia di insiemi. Tale generalizzazione viene introdotta attraverso l'operatore di insiemi Σ, e indicata con $\Sigma(A, B)$ oppure $(\Sigma x \in A)B(x)$.

(i) Σ - Formazione

$$\frac{A\ set \qquad \begin{matrix}(x \in A)\\ B(x)\ set\end{matrix}}{\Sigma(A,B)\ set}$$

(ii) Σ - Introduzione

$$\frac{a \in A \qquad b \in B(a)}{\langle a,b\rangle \in \Sigma(A,B)}$$

(iii) Σ - Eliminazione

$$\frac{c \in \Sigma(A,B) \qquad \begin{matrix}(x \in A,\ y \in B(x))\\ d(x,y) \in C(\langle x,y\rangle)\end{matrix}}{\mathbf{E}(c,d) \in C(c)}$$

(iv) Σ - Uguaglianza

$$\frac{a \in A \qquad b \in B(a) \qquad \begin{matrix}(x \in A,\ y \in B(x))\\ d(x,y) \in C(\langle x,y\rangle)\end{matrix}}{\mathbf{E}(\langle a,b\rangle,d) = d(a,b) \in C(\langle a,b\rangle)}$$

Nelle ultime due regole C è una famiglia di insiemi dipendente da $\Sigma(A,B)$.
Spiegazione di $\mathbf{E}(c,d)$:

(i) si esegua c, ottenendo un elemento canonico, $\langle a,b\rangle$, di $\Sigma(A,B)$;
(ii) si sostituisca $a \in A$ ad x e $b \in B(a)$ ad y nella seconda premessa, ottenendo $d(a,b) \in C(\langle a,b\rangle)$;
(iii) si esegua $d(a,b)$ così da pervenire ad un elemento canonico, e, di $C(\langle a,b\rangle)$. Si osservi che $c = \langle a,b\rangle \in \Sigma(A,B)$, quindi $C(c) = C(\langle a,b\rangle)$, e dunque e è elemento canonico di $C(c)$.

5.5.1. *Quantificatore Esistenziale.* La lettura proposizionale di queste regole fornisce le regole di deduzione naturale per il quantificatore esistenziale.

$\exists$ - Formazione

$$\frac{A\ set \qquad \begin{matrix}(x \in A)\\ B(x)\ prop\end{matrix}}{(\exists x \in A)B(x)\ prop}$$

$\exists$ - Introduzione

$$\frac{a \in A \qquad B(a)\ true}{(\exists x \in A)B(x)\ true}$$

$\exists$ - Eliminazione

$$\frac{(\exists x \in A)B(x)\ \mathit{true} \qquad \begin{array}{c}(x \in A,\ B(x)\ \mathit{true}) \\ \\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}$$

5.5.2. *Prodotto cartesiano e congiunzione.* Qualora B non dipenda da A otteniamo, come caso particolare, il prodotto cartesiano dei due insiemi. La lettura proposizionale fornisce invece la congiunzione, &.

& - Formazione

$$\frac{A\ \mathit{prop} \qquad \begin{array}{c}(A\ \mathit{true}) \\ \\ B\ \mathit{prop}\end{array}}{A\&B\ \mathit{prop}}$$

& - Introduzione

$$\frac{A\ \mathit{true} \qquad B\ \mathit{true}}{A\&B\ \mathit{true}}$$

& - Eliminazione

$$\frac{A\&B\ \mathit{true} \qquad \begin{array}{c}(A\ \mathit{true},\ B\ \mathit{true}) \\ \\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}$$

La regola di formazione di & segue il pattern già visto, permettendo di derivare B *prop* dal giudizio A *true*. La regola di introduzione è esattamente la regola che si incontra nella deduzione naturale. La regola di eliminazione è invece una generalizzazione delle regole di deduzione naturale che consente di esprimere entrambe le regole di eliminazione in modo molto compatto. Le usuali regole sono infatti ottenute da questa semplicemente istanziando C con A e con B, rispettivamente.

5.5.3. *Esercizi.*

(i) Proiezioni sinistra e destra: siano

$$\begin{aligned} \pi_0(c) &:= \mathbf{E}(c, (x,y).x) \\ \pi_1(c) &:= \mathbf{E}(c, (x,y).y) \end{aligned}$$

Si mostri la derivabilità delle seguenti regole:

$$\frac{c \in \Sigma(A,B)}{\pi_0(c) \in A} \qquad \frac{a \in A \qquad b \in B(a)}{\pi_0(\langle a,b\rangle) = a \in A}$$

$$\frac{c \in \Sigma(A, B)}{\pi_1(c) \in B(\pi_0(c))} \qquad \frac{a \in A \qquad b \in B(a)}{\pi_1(\langle a, b\rangle) = b \in B(a)}$$

(ii) Si giustifichino i giudizi:
  (a) $A \supset (B \supset (A\&B))$ *true*.
  (b) $A\&B \supset B$ *true*.
  (c) $(A \supset B) \supset ((A \supset C) \supset (A \supset B\&C))$ *true*.

5.6. **Somma disgiunta di due insiemi.** Vogliamo ora rappresentare la somma disgiunta di due insiemi. Si noti che anche in questo caso si potrebbe optare per una somma disgiunta non primitiva, definita utilizzando Σ. Si preferisce però assumere $+$ come primitivo, soprattutto perchè nell'interpretazione proposizionale esso rappresenta la disgiunzione, in genere primitiva nei sistemi intuizionistici.

Seguendo Martin Löf usiamo notazione infissa per il simbolo della somma disgiunta, $+$.

Gli elementi canonici di $A{+}B$ sono della forma $\mathbf{inl}(a)$ per $a \in A$ oppure $\mathbf{inr}(b)$ per $b \in B$; le costanti **inl** e **inr** hanno la funzione di indicare se un elemento di $A{+}B$ "proviene da" (ovvero è giustificato da un elemento di) A oppure B. Si ossservi che di conseguenza dato un elemento non canonico c di $A{+}B$, possiamo stabilire se c è elemento di A o di B, poiché calcolandolo si ottiene un elemento canonico della forma $\mathbf{inl}(a)$ oppure $\mathbf{inr}(b)$.

(i) $+$ - Formazione

$$\frac{A\ set \qquad B\ set}{A{+}B\ set}$$

(ii) $+$ - Introduzione

$$\frac{a \in A}{\mathbf{inl}(a) \in A{+}B} \qquad \frac{b \in B}{\mathbf{inr}(b) \in A{+}B}$$

(iii) $+$ - Eliminazione

$$\frac{c \in A{+}B \qquad \begin{array}{c}(x \in A)\\ d(x) \in C(\mathbf{inl}(x))\end{array} \qquad \begin{array}{c}(y \in B)\\ e(y) \in C(\mathbf{inr}(y))\end{array}}{\mathbf{D}(c, d, e) \in C(c)}$$

(iv) + - Uguaglianza

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ d(x) \in C(\mathbf{inl}(x))\end{array} \qquad \begin{array}{c}(y \in B)\\ e(y) \in C(\mathbf{inr}(y))\end{array}}{\mathbf{D}(\mathbf{inl}(a), d, e) = d(a) \in C(\mathbf{inl}(a))}$$

$$\frac{b \in A \qquad \begin{array}{c}(x \in A)\\ d(x) \in C(\mathbf{inl}(x))\end{array} \qquad \begin{array}{c}(y \in B)\\ e(y) \in C(\mathbf{inr}(y))\end{array}}{\mathbf{D}(\mathbf{inr}(b), d, e) = e(b) \in C(\mathbf{inr}(b))}$$

Nelle ultime regole C è una famiglia di insiemi dipendente da $A{+}B$.
Spiegazione di $\mathbf{D}(c, d, e)$:

(i) si calcoli $c \in A{+}B$, ottenendo come valore un elemento canonico di $A{+}B$ della forma $\mathbf{inl}(a)$ per $a \in A$ oppure $\mathbf{inr}(b)$ per $b \in B$;
(ii) nel primo caso si sostituisca a ad x in $d(x)$, così da ottenere $d(a)$ e lo si calcoli. La seconda premessa consente di concludere che $d(a) \in C(\mathbf{inl}(a))$. Calcolando $d(a)$ si perviene dunque ad un elemento canonico di $C(\mathbf{inl}(a))$;
(iii) analogamente, nel secondo caso, si sostituisca b ad y in $e(y)$ ottenendo $e(b) \in C(\mathbf{inr}(b))$. Se ne calcoli il valore, producendo così un elemento canonico di $C(\mathbf{inr}(b))$;
(iv) chiaramente in ogni caso otteniamo un elemento canonico di $C(c)$, poiché se $c = \mathbf{inl}(a) \in A{+}B$, allora $C(c) = C(\mathbf{inl}(a))$, mentre se $c = \mathbf{inr}(b) \in A{+}B$, allora $C(c) = C(\mathbf{inr}(b))$.

5.6.1. *Disgiunzione.* L'interpretazione proposizionale di + è la disgiunzione. Ne riportiamo di seguito le regole.

$\vee$ - Formazione

$$\frac{A\ prop \qquad B\ prop}{A \vee B\ prop}$$

$\vee$ - Introduzione

$$\frac{A\ true}{A \vee B\ true} \qquad\qquad \frac{B\ true}{A \vee B\ true}$$

$\vee$ - Eliminazione

$$\frac{A \vee B\ true \qquad \begin{array}{c}A\ true\\ C\ true\end{array} \qquad \begin{array}{c}B\ true\\ C\ true\end{array}}{C\ true}$$

5.6.2. *Esercizi.* Si giustifichino i seguenti giudizi:
(i) $B \supset A \vee B$ *true*.
(ii) $(A \supset C) \supset ((B \supset C) \supset (A \vee B \supset C))$ *true*.

5.7. **Uguaglianza proposizionale.** Come già accennato vi sono tre distinte nozioni di uguaglianza nella teoria dei tipi: judgemental equality, definitional equality e propositional equality. Quest'ultima è introdotta nella teoria dei tipi per riflettere a livello proposizionale i giudizi asserenti l'uguaglianza tra due elementi di un insieme.

Si osservi che le regole dell'uguaglianza proposizionale hanno dato luogo a numerose discussioni, in particolare in relazione alla distinzione tra teorie dei tipi intensionali ed estensionali. La questione è di rilievo anche perchè ha un notevole impatto sull'implementabilità della teoria. Martin Löf ha presentato sia versioni intensionali (si veda per esempio [21]) che estensionali. Presentiamo qui la teoria dei tipi estensionale di [23], ma evidenziamo che non è adatta ad un'implementazione, perchè la presenza dell'uguaglianza estensionale compromette la decidibilità della relazione di appartenenza della teoria.[14]

(i) **I** - Formazione

$$\frac{A\ set \qquad a \in A \qquad b \in A}{\mathbf{I}(A, a, b)\ set}$$

(ii) **I** - Introduzione

$$\frac{a = b \in A}{r \in \mathbf{I}(A, a, b)}$$

(iii) **I** - Eliminazione

$$\frac{c \in \mathbf{I}(A, a, b)}{a = b \in A}$$

(iv) **I** - Uguaglianza

$$\frac{c \in \mathbf{I}(A, a, b)}{c = r \in \mathbf{I}(A, a, b)}$$

Si osservi che nella regola di **I**-introduzione l'elemento r non dipende da a, b e A. È questa una caratteristica dell'uguaglianza proposizionale estensionale. Regole di uguaglianza intensionali introducono invece elementi canonici dipendenti da a, b e A.

La regola di eliminazione consente inoltre di passare dall'uguaglianza proposizionale al giudizio corrispondente, permette dunque di tornare al livello

[14]Si veda per esempio [25] per chiarimenti e per un richiamo delle regole di uguaglianza proposizionale in un contesto intensionale.

dei giudizi. Anche in questo caso, come in quello del prodotto generalizzato, è possibile sostituire a questa regola un'eliminazione che costituisca un principio di induzione strutturale, per poi derivare da quest'ultima la regola qui assunta come primitiva. Si veda per es. [25].

5.7.1. *Esercizi.*

(i) *Unicità dell'elemento* 0_1 *di* $\mathbf{N_k}$. Si dimostri la derivabilità della seguente regola:

$$\frac{c \in \mathbf{N_1}}{c = 0_1 \in \mathbf{N_1}}$$

(ii) *Assioma di introduzione dell'identità proposizionale.* Si assumano le premesse: A *set* e $x \in A$ e si derivi il seguente giudizio:

$$(\forall x \in A)\, \mathbf{I}(A, x, x)\ true.$$

(iii) *Assioma di eliminazione dell'identità proposizionale.* Si assuma A *set* e $B(x)$ *prop* $(x \in A)$. Si derivi il giudizio:

$$(\forall x \in A)(\forall y \in A)(\mathbf{I}(A, x, y) \supset (B(x) \supset B(y)))\ true.$$

(iv) *Converso della proiezione.* Si derivi la regoa di inferenza:

$$\frac{c \in \Sigma(A, B)}{c = (\pi_0(c), \pi_1(c)) \in \Sigma(A, B)}$$

5.7.2. *Successore.* J. Smith [30] ha dimostrato come non sia possibile, senza postulare un tipo universo, derivare la validità del quarto assioma di Peano. Se ci si propone di esprimere l'aritmetica nella teoria senza universi, si deve dunque assumere la regola:

$$\frac{x \in \mathbf{N}}{\mathbf{I}(\mathbf{N}, suc(x), 0) = \mathbf{N}_0}$$

6. Esempi

6.1. **L'assioma di scelta.** L'assioma di scelta, in breve **AC** (Axiom of Choice), è uno tra i più discussi principi della teoria degli insiemi.

Nel caso della teoria classica degli insiemi, **ZF**, l'assioma di scelta è stato dimostrato consistente ma indipendente dalla teoria.[15]

Nel caso della teoria costruttiva degli insiemi, così come nel caso di altre teorie intuizionistiche puramente estensionali, l'assioma di scelta risulta essere incompatibile con la logica intuizionistica, in quanto la sua assunzione produce istanze del principio del terzo escluso.

[15] La consistenza é stata dimostrata da Gödel, mediante la gerarchia dei costruibili, un modello interno della teoria degli insiemi che verifica **AC**. L'indipendenza é stata dimostrata da Choen, mediante la tecnica del forcing.

A prima vista può risultare dunque sorprendente che l'assioma di scelta sia invece valido nella teoria costruttiva dei tipi. Ciò è da ascrivere al peculiare significato che i quantificatori assumono in essa. La teoria conserva infatti in ogni caso, anche nelle sue versioni estensionali, un carattere di intensionalità: in altri termini, l'estensionalità è qui imposta al di sopra di una teoria intensionale. L'intensionalità residua è rivelata dal fatto che gli elementi di un insieme non possono essere caratterizzati indipendentemente dall'insieme di cui fanno parte e portano sempre con sè anche informazione relativa all'uguaglianza caratterizzante l'insieme.

Si suggerisce come esercizio la derivazione dell'assioma di scelta, ovvero del seguente giudizio:

$$\begin{array}{l}(\forall x \in A)(\exists y \in B(x))C(x,y) \\ \supset (\exists f \in \Pi(A,B))(\forall x \in A)C(x,\mathbf{Ap}(f,x))\ \mathit{true}.\end{array}$$

6.2. **I reali di Cauchy.** Una delle principali operazioni su insiemi disponibili nella teoria assiomatica degli insiemi è quella di separazione: dato un insieme A formiamo un sottinsieme di A costituito esattamente da quegli elementi dell'insieme dato che godono di una certa proprietà B. L'insieme risultante viene in genere indicato con $\{x \in A : B(x)\}$.

Nella teoria dei tipi, possiamo rappresentare la separazione per mezzo di $\Sigma(A,B)$, dove A è un insieme e B una funzione proposizionale dipendente da A. La rappresentazione di $\{x \in A : B(x)\}$ è dunque un insieme di coppie la prima componente delle quali è un elemento, a, di A, la seconda è una dimostrazione, $b(a)$ che $B(a)$ vale.

Un esempio paradigmatico è dato dalla definizione dei reali di Cauchy attraverso il seguente tipo:

$$R \ := \ (\Sigma x \in \mathbf{N} \to \mathbf{Q})\mathit{Cauchy}(x)$$

dove per $a \in \mathbf{N} \to \mathbf{Q}$,

$$\mathit{Cauchy}(a) \ := \ (\forall r \in \mathbf{Q})(r > 0 \supset (\exists m \in \mathbf{N})(\forall n \in \mathbf{N})(|a_{m+n} - a_m| \leq r)).$$

Riferimenti bibliografici

[1] P. Aczel: *The Type Theoretic Interpretation of Constructive Set Theory*, in: A. MacIntyre, L. Pacholski, J. Paris (eds.), *Logic Colloquium '77* (North–Holland, Amsterdam-New York, 1978) pp. 55–66.

[2] P. Aczel, M. Rathjen: *Notes on Constructive Set Theory*, Draft available from the Internet at the address: http://www.cs.man.ac.uk/ petera/.

[3] L. Augustsson, T. Coquand, B. Nordström: *A short description of Another Logical Framework*, in*Proceedings of the First Workshop on Logical Framekorks*, Antibes, 1990, pp. 39–42.

[4] M. Beeson: *Foundations of Constructive Mathematics*, (Springer Verlag, Berlin 1985).

[5] E. Bishop: *Foundations of constructive analysis* (McGraw-Hill, New York, 1967).

[6] R. L. Constable: *Naïve computational type theory*, in: H. Schwichtenberg, R. Steinbrüggen (eds.), *Proof and System-Reliabilitiy* (Kluwer Academic Publisher, Dordrecht, 2002), pp. 213–259.

[7] R. L. Constable et al.: *Implementing Mathematics with the NuPRL Proof Development System*, (Prentice-Hall, Englewood Cliffs, NJ, 1986).

[8] T. Coquand, B. Nordström, J. Smith, B. Sydow: *Type Theory and Programming*, `www.cs.chalmers.se/Cs/Research/Logic`.

[9] T. Coquand, G. Huet: *The Calculus of Constructions*, *Information and Computation*, 76(2/3) (1988), pp. 95–120.

[10] N. G. de Brujin: *A survey of the project AUTOMATH* in J. P. Seldin and J. R. Hindley (eds.), *To H. B. Curry: Essays on combinatory Logic, Lambda Calculus, and Formalism* (Academinc Press, New York, 1980), pp. 589–606 .

[11] M. Dummett: *Elements of intuitionism* (Clarendon Press, Oxford, 1977).

[12] S. Feferman: *A language and axioms for explicit mathematics* in: J. Crossley (ed.), *Algebra and Logic*, *Lecture Notes in Mathematics*, vol 450, (Springer, Berlin 1975), pp. 87–139.

[13] H. Friedman: *Set theoretic foundations for constructive analysis*, Annals of Mathematics 105 (1977), pp. 1–28.

[14] J.-Y. Girard: *Interprétation fonctionnelle et élimination des coupures da l'arithmétique d'ordre supérieur* (Thèse, Université Paris VII, 1972).

[15] J.-Y. Girard, P. Taylor, Y. Lafont: *Proofs and types* Cambridge Tractacts in Computer Science, vol. 7 (Cambridge University Press 1989).

[16] N. D. Goodman: *A theory of constructions equivalent to arithmetic*, in J. Myhill, A. Kino and R. E. Vesley (eds.), Intuitionism and Proof Theory (North-Holland, Amsterdam, 1970), pp. 101-120.

[17] J. van Heijenoort, ed.: *From Frege to Gödel. A Source Book in Mathematical Logic 1879-1931*, (Harvard University Press, Cambridge Mass., 1967).

[18] A. Heyting: *Intuitionism, An Introduction*, (North-Holland, Amsterdam, 1956).

[19] G. Kreisel: *Foundations of intuitionistic logic* in E. Nagel, P. Suppes and A. Tarski (eds.), *Logic, Methodology and philosophy of science* III (Stanford Univeristy Press, Stanford California 1962), pp. 198-210.

[20] G. Kreisel: *Mathematical logic*, in T. L. Saaty, (ed.), *Lectures on Modern Mathematics*, Vol III (Wiley, New York, 1965), pp. 95-195.

[21] P. Martin-Löf: *An intuitionistic Theory of types: predicative part*, in H. E. Rose and J. C Shepherdson (eds.), *Logic Colloquium 1973* (North Holland, Amsterdam, 1975), pp. 73–118.

[22] P. Martin-Löf: *Constructive Mathematics and Computer Programming*, in L. J. Choen (ed.), *Logic, Methodology, and Philosophy of Science VI* (North Holland, Amsterdam, 1982), pp. 153–175.

[23] P. Martin-Löf: *Intuitionistic Type Theory*, (Bibliopolis, Naples, 1984).

[24] J. Myhill: *Constructive Set Theory*, Journal of Symbolic Logic, 40 (1975), pp. 347–382.

[25] B. Nordström, K. Petersson, J. Smith: *Programming in Martin Löf's Type Theory. An introduction*, (Oxford University Press, 1990).

[26] D. Prawitz: *Natural Deduction. A proof theoretical study*, (Almquist and Wiskell, Stockholm).

[27] D. Scott: *Constructive validity* in: Symposium on Automatic Demonstration, Lecture Notes in Mathematics, Vol. 125 (Springer, Berlin, 1970) pp. 237–275.

[28] T. Streicher: *Semantics of Type Theory* (Birkhaeuser Verlag. XII, Basel, 1991).)

[29] G. Sambin, J. Smith: *Twenty-Five Years of Constructive Type Theory* of Oxford Logic Guides, vol. 36, (Oxford, 1998).

[30] J. Smith: *The independence of Peano's fourth axiom from Martin-Löf's type theory without Universes*, Journal fo Symbolic Logic, 53(3), 1988.

[31] G. Sommaruga: *History and philosophy of constructive type theory* (Kluwer Academic Publisher, Dordrecht, 2000).

[32] A. S. Troelstra: *From constructivism to computer science*, Theoretical Computer Science, 211, (1999), pp. 233–252.
[33] A. S. Troelstra and D. van Dalen: *Constructivism in Mathematics: an Introduction*, volumes I and II (North–Holland, Amsterdam 1988).

Dipartimento di Filosofia, Università degli Studi di Firenze, Via Bolognese, 52, Firenze, Italia, E-mail: Laura.Crosilla@unifi.it