Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

LA MIA FANTASTICA OTTIMISTICA TESI

Relatore:
Chiar.mo Prof.
Claudio Sacerdoti Coen

Presentata da:
Mattia Girolimetto

I Appello di Laurea
Anno Accademico 2022-2023

Indice

1 Introduzione **2**
1.1 Teoria dei Tipi 2
1.1.1 Il paradosso di Russel 2
1.1.2 La Teoria dei Tipi 2
1.1.3 L'isomorfismo di Curry-Howard 3
1.1.4 La teoria dei tipi nella pratica 4
1.2 Dimostratori Interattivi di Teoremi 4
1.2.1 Matita 4
1.2.2 Dedukti 4
1.3 Interoperabilità DeduktiMatita 4
1.3.1 Interoperabilità tra sistemi 4
1.3.2 Da Matita a Dedukti 5
1.3.3 Da Dedukti a Matita 5

2 Parte tecnica **6**
2.1 Export: da Matita a Dedukti 6

3 Conclusioni **7**

4 Sviluppi futuri **8**

Capitolo 1

Introduzione

1.1 Teoria dei Tipi

1.1.1 Il paradosso di Russel

Quando il matematico inglese Bertrand Russel propose il paradosso oggi chiamato a suo nome, nei primi anni del ′900 scatenò quella che venne definita come la*crisi dei fondamenti matematici*. Il paradosso molto semplicemente evidenziava come la teoria degli insiemi usata fino a quel punto (oggi definita teoria degli insiemi *naive*) rendesse possibile definire un insieme come il seguente

$$X = \{Y | Y \notin Y\}$$

La contraddizione avviene quando ci si domanda se $X \in X$, in quanto se X appartenese a sé stesso allora non apparterrebbe all'insieme degli insiemi che appartengono loro stessi, ovvero X stesso. In formule:

$$X \in X \Leftrightarrow X \notin X$$

Una delle strategie pensate dunque per aggirare questo paradosso fu la *teoria dei tipi*.

1.1.2 La Teoria dei Tipi

La *teoria dei tipi* è una branca della matematica, della logica, dell'informatica teorica il cui obbiettivo è quello di studiare i così detti *type system*, ovvero insiemi di regole che associano una proprietà chiamata *tipo* ad degli oggetti chiamati *termini*. Nonostante siano state proposte molteplici teorie, le principali emerse sono due: il λ-*calcolo tipato* di Alonozo Church e la *teoria dei tipi intuizionistica* di Per Martin-Löf.

Intuitivamente, assegnare un tipo ad un termine significa assegnare al termine un'etichetta che rappresenta la natura del termine stesso. Esempi comuni possono essere:

- 42 è un numero naturale
- -5 è un numero intero
- $falso$ è un valore di verità

Formalmente si usa rappresentare queste espressioni separando il termine dal tipo usando il simbolo ':'. Gli esempi precedenti diventano quindi:

- $42 : \mathbb{N}$
- $-5 : \mathbb{Z}$
- $falso : \mathbb{B}$

Nella teoria dei tipi, anche le funzioni possono sono termini, e quindi possono essere a loro volta tipizzate. Ad esempio, la seguente funzione rappresentata con un λ-termine appartenente al λ-calcolo di Church

$$(\lambda\ x : \mathbb{N}\ .\ x + x)$$

è definita da $\mathbb{N}$ a $\mathbb{N}$, e per tanto ha tipo $\mathbb{N} \to \mathbb{N}$

1.1.3 L'isomorfismo di Curry-Howard

Sempre durante il '900 il matematico e logico Haskell Curry e il logico William Alvin Howard, scoprono una corrispondenza diretta tra prove formali e programmi. In particolare notano che gli operatori logici e le regole usate durante una dimostrazione formale sono equivalenti a tipi e ai costrutti usati nei programmi scritti usando linguaggi di programmazione funzionali. Ne segue anche che il verificare la correttezza di una prova è analogo al verificare la correttezza degli assegnamenti di tipo di un programma (*type checking*). Nella sua formulazione più generale, l'isomorfismo di Curry-Howard può essere riassunto con la seguente tabella:

Logica	**Informatica**
$\top$	Tipo unit
$\bot$	Tipo vuoto/void
$\wedge$	Tipi prodotto
$\vee$	Tipi somma
$\Rightarrow$	Tipi funzione
$\exists$	Tipi Σ
$\forall$	Tipi Π

1.1.4 La teoria dei tipi nella pratica

La teoria dei tipi trova quindi grande applicazione nel campo dell'informatica grazie allo studio e allo sviluppo dei linguaggi di programmazione. Inoltre, grazie all'isomorifismo di Curry-Howard, ha permesso lo sviluppo di dimostratori automatici di teoremi e di dimostratori interattivi di teoremi, i quali sono soggetto di questa tesi.

1.2 Dimostratori Interattivi di Teoremi

Un dimostratore interattivo di teoremi (o *proof assistant*) è un software che permette all'utente di costruire e verificare delle dimostrazioni matematiche formali. Presa in input una prova espressa utilizzando uno specifico linguaggio formale, simile ad un linguaggio di programmazione, il software è in grado di verificarne la correttezza. In questo modo si possono costruire dimostrazioni in modo interattivo, controllando progressivamente la corretezza di ogni passo. Uno dei benefici chiave dell'usare un dimostratore interattivo automatico è l' abilità di eliminare gli errori e le ambiguità che possono comparire nelle dimostrazioni tradizionali.

1.2.1 Matita

Matita è un proof assistant sotto sviluppo nel dipartimento di informatica all' Università di Bologna. E' basato sul *calcolo delle costruzioni coinduttive*. Il software, che è open source, è scritto nel linguaggio di programmazione OCAML ed è rilasciato secondo i termini della GNU General Public Licence.

1.2.2 Dedukti

Dedukti (che significa "dedurre" in esperanto) è un *logical framework* sviluppato da alcuni ricercatori del INRIA, basato sul *calcolo lambdapi*. Il software è open source, anch'esso scritto nel linguaggio di programmazione OCAML e distribuito secondo i termini della CeCILLB License.

1.3 Interoperabilità DeduktiMatita

1.3.1 Interoperabilità tra sistemi

Il numero di proof assistant è aumentato nel tempo. Ciò porta sicuramente un beneficio alla comunità scientifica, in quanto dimostra un crescente interesse

verso lo sviluppo di questi strumenti. Tuttavia, unito alla forte diversità che li caratterizza individualmente, questo fenomeno porta inevitabilmente ad una *frammentazione* della conoscenza. Non è quasi mai possibile infatti per un utente dimostrare la veridicità di un teorema usando un proof assistant e usare la stessa dimostrazione in un altro di questi tool. Il problema è dovuto a fattori facilmente aggirabili, come ad esempio la differenza sintattica dei due linguaggi proprietari, ma anche a fattori non facilmente aggirarabili, come nel caso in cui i due tool usino calcoli con diversi livelli di espressività.

Nasce dunque l'esigenza di favorire l'interoperabilità tra questi sistemi, in modo da arginare questo problema e favorire lo sviluppo scientifico. A tale scopo, nel tempo sono state aggiunte ad alcuni di tool delle funzionalità di export, per permettere all'utente di ottenere la propria dimostrazione in un formato compatibile con un altro software.

1.3.2 Da Matita a Dedukti

Attorno al 2018 un team di sviluppatori del *Institut national de recherche en informatique et en automatique* ha sviluppato un fork di Matita con la possibilità di esportare le dimostrazioni in un formato compatibile con Dedukti. Questo fork è tutt'ora distribuito pubblicamente con il nome di *Krajono* ("matita" in esperanto) anche se non è stato aggiornato con gli ultimi sviluppi del Matita baseline.

1.3.3 Da Dedukti a Matita

Come visto nel paragrafo precedente, usando Krajono è possibile esportare del codice Matita verso Dedukti, tuttavia non è possibile fare il contrario, in quanto ne Krajono, ne Dedukti stesso godono di questa funzionalità. L'export è dunque a senso unico, e qualcosa di esportato non può essere reimportato in Matita. Il lavoro di questa tesi è proprio il seguente: rendere Matita capace di esportare ed importare codice da e verso Dedukti. Con un export a doppio senso gli sviluppatori Matita saranno in grado di usare dimostrazioni Dedukti e vice versa.

Capitolo 2

Parte tecnica

2.1 Export: da Matita a Dedukti

Capitolo 3

Conclusioni

Capitolo 4

Sviluppi futuri