

Indice

1	Introduzione	3
1.1	Stato dell'arte	3
1.2	ELPI	5
1.2.1	Introduzione al linguaggio	5
1.2.2	Sistema di propagazione dei vincoli	5
1.3	Haskell	5
1.3.1	BNF utilizzata	5
1.3.2	Type class	5
1.3.3	Let-in	5
2	Implementazione	7
2.1	STLC	7
2.2	Funzioni ricorsive, match	7
2.3	Type class, istanza, schema	7
2.4	Let-in	7
3	Conclusioni	9
3.1	Riassumendo	9
3.2	La mia esperienza con ELPI	9
3.3	Il mio lavoro	9
3.4	Sviluppi futuri	9
3.4.1	Parser	9
3.4.2	Testing	9
3.4.3	Estensioni	9

Capitolo 1

Introduzione

Il lavoro da me svolto consiste nell'implementazione dell'algoritmo di type inference di Haskell in ELPI.

1.1 Stato dell'arte

Iniziamo questa trattazione ponendo alcune basi. Esse faciliteranno la comprensione degli argomenti seguenti.

Haskell è un linguaggio di programmazione che adotta il paradigma di programmazione funzionale. Al suo interno è presente il lambda calcolo e il meccanismo delle type class; è presente inoltre una parte più ampia che corrisponde alle librerie.

La type inference è il rilevamento automatico del tipo di dato di un'espressione in un linguaggio di programmazione. La capacità di dedurre i tipi automaticamente semplifica molte attività di programmazione, lasciando il programmatore libero di omettere le annotazioni sui tipi pur consentendo il type check.

ELPI è un linguaggio di programmazione logico. Esso è un'estensione con vincoli del linguaggio λ Prolog, il quale a sua volta è un'estensione di Prolog a una logica di ordine superiore.

Prolog è un linguaggio di programmazione che adotta il paradigma di programmazione logica. Si basa sul calcolo dei predicati (logica del prim'ordine); la sintassi è composta da formule dette clausole che sono disgiunzioni di letterali del prim'ordine. L'esecuzione di un programma Prolog è comparabile alla dimostrazione di un teorema mediante la regola di inferenza detta risoluzione. I concetti fondamentali di questo linguaggio sono l'unificazione, la ricorsione in coda e il backtracking.

λ Prolog è, come già detto, un'estensione di Prolog. Le caratteristiche principali in aggiunta, rispetto a Prolog, sono il polimorfismo, la programmazione di ordine superiore e il lambda calcolo tipato.

Non essendo λ Prolog un linguaggio di programmazione con vincoli risulta impossibile implementare la type inference mediante esso. I limiti dell'utilizzo di tale linguaggio si riscontrano in particolare nel tentare di codificare il tipaggio per i costrutti del let-in e della type class.

L'unica strategia attuabile sarebbe quella di codificare interamente il sistema punto per punto, il che evidentemente rende tale strategia impraticabile.

Si è reso dunque necessario l'utilizzo di ELPI, la cui maggiore espressività permette di svolgere operazioni impossibili da codificare in λ Prolog. Prendiamo come esempio i due casi indicati precedentemente:

- Per codificare il tipaggio del let-in è necessario l'utilizzo del meccanismo mode di ELPI. Infatti, grazie ad esso, si è in grado di accorgersi se un elemento del codice è una variabile, così da poterla gestire in modo appropriato.
- Per codificare il tipaggio della type class è necessario l'utilizzo dei vincoli. Infatti questi possono sussistere anche non totalmente istanziati e quindi permettono, ad esempio, di fissare l'obbligo di appartenenza di una variabile di tipo ad un'istanza di type class.

Entrambi i requisiti sono caratteristiche presenti in ELPI ma non in λ Prolog. Risulta dunque evidente la necessità di utilizzare ELPI come linguaggio di programmazione per poter raggiungere lo scopo prefissato.

1.2 ELPI

1.2.1 Introduzione al linguaggio

ELPI, così come λ Prolog, è un linguaggio logico di ordine superiore (HOLP language - Higher Order Logic Programming language).

1.2.2 Sistema di propagazione dei vincoli

1.3 Haskell

1.3.1 BNF utilizzata

1.3.2 Type class

1.3.3 Let-in

Capitolo 2

Implementazione

2.1 STLC

2.2 Funzioni ricorsive, match

2.3 Type class, istanza, schema

2.4 Let-in

Capitolo 3

Conclusioni

3.1 Riassumendo

3.2 La mia esperienza con ELPI

3.3 Il mio lavoro

3.4 Sviluppi futuri

3.4.1 Parser

3.4.2 Testing

3.4.3 Estensioni

Ringraziamenti