

# Implementazione in un linguaggio logico con vincoli di ordine superiore della type inference di Haskell

Daniele Polidori

Alma Mater Studiorum - Università di Bologna  
Corso di Laurea in Informatica

III Sessione  
Anno Accademico 2018/2019

**Problema da risolvere:** Implementare in un linguaggio logico con vincoli di ordine superiore la type inference di Haskell.

- Codifica della sintassi di Haskell.
- Implementazione dell'algoritmo di type inference con le type class di Haskell.

Linguaggio di programmazione utilizzato: ELPI.

- 1 Dimostrare che ELPI è più espressivo di  $\lambda$ Prolog.
- 2 Strumento di prova per testare, implementare e studiare nuove estensioni al meccanismo delle type class di Haskell.  
(Sviluppi futuri)

## Frammento:

1. `type typeclass nome_classe -> tipo -> list declaration -> prop.`
2. `type istanza nome_classe -> tipo -> list implementation -> prop.`
- 3.
4. `mode (istanza i i o).`
5. `istanza N (uvar _ as X) L :- !, declare_constraint (istanza N X L) [X].`

## Esempio:

1. `typeclass printable T [fun_decl print (base (arr T string))].`
2. `istanza printable int [fun_impl print (fun print_int)].`

```
goal> pi x \ of x int => of (app (fun print) x) T1.  
%   T1 = string
```

```
goal> pi x \ of x bool => of (app (fun print) x) T2.  
%   Failure
```

```
goal> pi x \ of x T3 => of (app (fun print) x) T4.  
%   T3 = X0  
%   T4 = string  
%   istanza printable X0 [...]
```

- Higher Order constraint Logic Programming language.
- Estensione con vincoli del linguaggio  $\lambda$ Prolog.

### Frammento (STLC):

1. `kind tipo type.`
2. `type arr tipo -> tipo -> tipo.`
- 3.
4. `kind term type.`
5. `type app term -> term -> term.`
6. `type lam (term -> term) -> term.`
- 7.
8. `type of term -> tipo -> prop.`
9. `mode (of i o).`
10. `of (uvar _ as X) T :- !, declare_constraint (of X T) [X].`
11. `of (app X Y) B :- of X (arr A B), of Y A.`
12. `of (lam F) (arr A B) :- pi x \ of x A => of (F x) B.`

# Istanziamento degli schemi

## Frammento:

1. `type for_all (tipo -> schema) -> schema.`
2. `type base tipo -> schema.`
3.
4. `type is_instance nome_classe -> tipo -> holds.`
5. `type implies holds -> schema -> schema.`
6. `type instantiate schema -> tipo -> prop.`
7.
8. `instantiate (base T) T.`
9. `instantiate (for_all S) 0 :- instantiate (S T_) 0.`
10. `instantiate (implies (is_instance N T) S) 0 :-`
11. `istanza N T _, instantiate S 0.`

## Esempio:

```
goal> instantiate (for_all t \ implies (is_instance printable t)
      (base (arr t int))) T.
%   T = arr X0 int
%   istanza printable X0 [...]
```

# Let-in: Regola di generalizzazione

## Frammento:

```
1. type retrieve_constraints tipo -> list prop -> prop.
2.
3. retrieve_constraints X [Y|L] :-
4.   declare_constraint (retrieve X Y) [X], !, retrieve_constraints X L.
5. retrieve_constraints _ [].
6.
7.
8. type retrieve tipo -> prop -> prop.
9.
10. constraint istanza retrieve {
11.   rule \ (retrieve X Y) (istanza N X L) <=> (Y = istanza N X L).
12.   rule \ (retrieve X Y) <=> false.
13. }
```

## Esempio:

```
goal> of (lam x \ let_in [coppia_decl f (lam g \ app g x)] (decl_let f)) T.
%   T = arr X0 (arr (arr X0 X1) X1)
%   associa_schema f (for_all c1 \ base (arr (arr X0 c1) c1))
```

## Stima del lavoro svolto:

- Tempo impiegato: 180 ore.
- Linee di codice prodotte: 600, scritte in linguaggio ELPI.

## Sviluppi futuri:

- 1 Parser
- 2 Testing
- 3 Estensioni



Vi ringrazio per l'attenzione

**Presentata da:** Daniele Polidori

**Relatore:** Professor Claudio Sacerdoti Coen