Haskell

un'implementazione in *StandardML*

Carboni Francesco
Cicio Ionuţ
?? Giovanni
Mazzella Marco

Indice

1 Haskell

1.1 Grammatica

$$k ::= 0 \mid 1 \mid ... \mid \pi \mid \sqrt{2} \mid ... \mid a \mid b \mid ... \mid \texttt{true} \mid \texttt{false} \mid ()$$

$$\text{Exp} ::= x \mid k \mid A \implies B \mid M = N \mid M < N \mid M + N \mid M \cdot N \mid M - N \mid$$

$$\text{let } x = M \text{ in } N \mid \text{if } B \text{ then } M \text{ else } N \mid \text{case } Q \text{ of } M_1 \to N_1, ..., M_n \to N_n \mid \text{Haskell } N$$

$$\text{Haskell} ::= f\ x = \text{Exp} \mid \text{module } x(\text{Haskell}_1, ..., \text{Haskell}_n) \text{ where Haskell}_\alpha, ..., \text{Haskell}_\eta \mid$$

$$\text{types?} \mid \text{classes?} \mid \text{instances?}$$

1.2 Semantica operazionale lazy static

$$\text{Env} : \text{Var} \xrightarrow{\text{fin}} \text{Exp} \times \text{Env} \cup \text{Var} \times \text{Exp} \times \text{Env}$$

$$\overset{v}{\rightsquigarrow} \subseteq \text{Env} \times \text{Exp} \times \text{Val}$$

$$\overset{p}{\rightsquigarrow} \subseteq \text{Haskell} \times \text{Val}$$

$$\overset{c}{\rightsquigarrow} \subseteq \text{Haskell} \times \text{Val}$$

In base a "interprete" o "compilato" cambia la semantica ??

Considereremo 2 tipi di semantiche per Haskell:

- quella usata per i programmi (quando si **compila** un file) che richiede la presenza di un `module` che esporta la funzione `main`:

```
module Main (main) where
main = ...
```

- quella usata dall'interprete per valutare le espressioni

1.2.1 Valutazione di un programma (compilatore)

Per valutare (dare un giudizio operazionale) un programma compilato, è necessario avere il modulo `Main`. Con questa semantica sto indicando che un funzione di un modulo può vedere tutte le altre funzioni, indipendentemente da dove sono dichiarate

$$\frac{(f_1, (M_1, \varnothing)) \cdots (f_n, (M_n, \varnothing)) \vdash \text{main } () \overset{v}{\rightsquigarrow} v}{\varnothing \vdash \text{module Main(main) where main}, f_1\ x_1 = M_1, ..., f_n\ x_n = M_n \overset{p}{\rightsquigarrow} v}$$

Sarebbe utile avere una semantica per gli `import` e una per portare un modulo in un ambiente, + una semantica per portare una funzione in un ambiente (specialmente per quello interpretato)

1.2.2 Valutazione di un programma (interprete)

1.2.3 Valutazione di un'espressione

$$E \vdash k \overset{v}{\rightsquigarrow} k$$

$$\frac{E' \vdash M \overset{v}{\rightsquigarrow} v}{E \vdash x \overset{v}{\rightsquigarrow} v} \text{ (se Env}(x) = (M, E'))$$

$$\frac{E(x, N, E') \vdash M' \overset{v}{\rightsquigarrow} v}{E \vdash fN \overset{v}{\rightsquigarrow} v} \text{ (se Env}(f) = (x, M', E'))$$

$$\frac{E \vdash B \overset{v}{\rightsquigarrow} \text{true} \quad E \vdash M \overset{v}{\rightsquigarrow} v}{E \vdash \text{if } B \text{ then } M \text{ else } N \overset{v}{\rightsquigarrow} v}$$

$$\frac{E \vdash B \overset{v}{\rightsquigarrow} \text{false} \quad E \vdash N \overset{v}{\rightsquigarrow} v}{E \vdash \text{if } B \text{ then } M \text{ else } N \overset{v}{\rightsquigarrow} v}$$

2 Monadi

Bibliografia

https://github.com/shwestrick/smlfmt https://smlhelp.github.io/book/docs/ TODO: smlnj TODO: millet