Midi-FreshML Grammar

12/01/2014

No further restriction is enforced on identifiers (such as distinguishing between name and data types) within the grammar itself. However such restrictions do exist; it's just that they are enforced during typechecking rather than during parsing.

The regular expressions present use the following syntax:

[a b]	'a' or 'b'
r*	zero or more occurences of regular expression r
r+	one or more occurences of regular expression r
a?	zero or one occurence of 'a'

⟨*program*⟩ ::=
- | name ⟨*nty*⟩ ;
- | type ⟨*dty*⟩ ⟨*ctor-list*⟩ ;
- | ⟨*exp*⟩ ;

⟨*nty*⟩ ::=
- | ⟨*id*⟩
- | ⟨*nty*⟩ , ⟨*id*⟩

⟨*dty*⟩ ::=
- | ⟨*id*⟩ where
- | ⟨*id*⟩ , ⟨*dty*⟩

⟨*id*⟩ ::= [a-z] [a-z A-Z 0-9 _ ']*

⟨*ctor-list*⟩ ::=
- | ⟨*ctor*⟩
- | ⟨*ctor-list*⟩ , ⟨*ctor*⟩

⟨*ctor*⟩ ::= ⟨*id*⟩ : ⟨*type-name*⟩

⟨*type-name*⟩ ::=
- | int | real | bool | string | unit | ⟨*id*⟩
- | ≪ ⟨*id*⟩ ≫ ⟨*type-name*⟩
- | ⟨*type-name*⟩ ∗ ⟨*type-name*⟩
- | ⟨*type-name*⟩ → ⟨*type-name*⟩

⟨*exp*⟩ ::=
- | ⟨*id*⟩
- | ⟨*id*⟩ ⟨*exp*⟩
- | ⟨*int-literal*⟩ | ⟨*real-literal*⟩ | ⟨*bool-literal*⟩ | ⟨*string-literal*⟩ | ()
- | fresh : ⟨*id*⟩
- | if ⟨*exp*⟩ = ⟨*exp*⟩ then ⟨*exp*⟩ else ⟨*exp*⟩
- | swap (⟨*exp*⟩ , ⟨*exp*⟩) in ⟨*exp*⟩
- | ≪ ⟨*exp*⟩ ≫ ⟨*exp*⟩
- | (⟨*exp*⟩ , ⟨*exp*⟩)
- | fun (⟨*id*⟩ : ⟨*type-name*⟩) → ⟨*exp*⟩
- | ⟨*exp*⟩ ⟨*exp*⟩
- | match ⟨*exp*⟩ with ⟨*branch*⟩
- | let ⟨*dec*⟩ in ⟨*exp*⟩

| let ⟨*dec*⟩
| ⟨*exp*⟩ ⟨*binary-op*⟩ ⟨*exp*⟩
| ⟨*unary-op*⟩ ⟨*exp*⟩
| (⟨*exp*⟩)

⟨*int-literal*⟩ ::= ∼? digit+

⟨*real-literal*⟩ :: = ∼? digit+ . digit*

⟨*bool-literal*⟩ ::= true | false

⟨*string-literal*⟩ ::= “ char* ”

⟨*branch*⟩ ::=
| | ⟨*pattern*⟩ → ⟨*exp*⟩
| ⟨*branch*⟩ | ⟨*pattern*⟩ → ⟨*exp*⟩

⟨*pattern*⟩ ::=
| _
| ⟨*id*⟩
| ⟨*id*⟩ ⟨*pattern*⟩
| ≪ ⟨*pattern*⟩ ≫ ⟨*pattern*⟩
| ()
| (⟨*pattern*⟩ , ⟨*pattern*⟩)
| (⟨*pattern*⟩)

⟨*dec*⟩ ::=
| ⟨*pattern*⟩ = ⟨*exp*⟩
| ⟨*rec_func*⟩ ⟨*exp*⟩

⟨*rec_func*⟩ ::= ⟨*id*⟩ (⟨*id*⟩ : ⟨*type-name*⟩) : ⟨*type-name*⟩ =

⟨*binary-op*⟩ ::= * | / | + | -

⟨*unary-op*⟩ ::= ∼