1 Lexical Issues

1.1 Identifiers

An identifier is a sequence of letters, digits, understacores (\_) and primes (\') starting with a letter. Uppercase letters are distinguished from lowercaes ones. In the following presentation, we use \textit{id} to ranges over identifiers and \textit{lab} to ranges over labels, which are either identifiers or sequences of digits.

1.2 Comments

There three froms of comments:

- A comment may start with (* and end with *).
- A comment may start with // and end with the end-of-line (EOL) symbol.
- A comment may start with //// and end with the end-of-file (EOF) symbol.

2 Types

The grammar for types is given in Figure 1.

2.1 Built-in Types

The named types \textit{bool}, \textit{char}, \textit{int}, and \textit{string} are predefined.

2.2 Tuple Types

A tuple type is described as a sequence of types enclosed in ’(). For instance, the following is an example of tuple type:

’(bool, char, ’(int, string))

2.3 Record Types

A record type is described as a sequence of fields enclosed in ’{}{}, where each field is described as \textit{lab} : \textit{ty}. For instance, the following is an example of record type:

’{1:bool, 2:char, 3:’(int, string)}
2.4 Reference Types

We use the symbol `&` followed by a type to denote a reference type. For instance, here is an example of reference type: `& '(int, int)`

2.5 Array Types

We use the symbol `@` followed by a type to denote an array type. For instance, here is an example of array type: `@ '(int, *int)`

2.6 Function Types

As usual, we use `->` to form function types. For instance, `int -> int` is a function type for functions from integers to integers; `(string, string) -> bool` is for functions that take two strings to produce a boolean value.

3 Expressions

4 Declarations

5 Standard Library

6 Sample Programs