11 Cool Syntax

Figure 1 provides a specification of Cool syntax. The specification is not in pure Backus-Naur Form (BNF); for convenience, we also use some regular expression notation. Specifically, A^* means zero or more A's in succession; A^+ means one or more A's. Items in square brackets [...] are optional. Double brackets $[\![\,]\!]$ are not part of Cool; they are used in the grammar as a meta-symbol to show association of grammar symbols (e.g. $a[\![bc]\!]^+$ means a followed by one or more bc pairs).

11.1 Precedence

The precedence of infix binary and prefix unary operations, from highest to lowest, is given by the following table:

```
.
@
~
isvoid
* /
+ -
<=  <  =
not
<-
```

All binary operations are left-associative, with the exception of assignment, which is right-associative, and the three comparison operations, which do not associate.

$$
\begin{array}{rcl}
\mathit{program} & ::= & \llbracket \mathit{class}; \rrbracket^{+} \\
\mathit{class} & ::= & \textbf{class}\ \mathrm{TYPE}\ [\textbf{inherits}\ \mathrm{TYPE}]\ \{\ \llbracket \mathit{feature}; \rrbracket^{*} \} \\
\mathit{feature} & ::= & \mathrm{ID}(\ [\ \mathit{formal}\ \llbracket, \mathit{formal} \rrbracket^{*}\ ]\ )\ :\ \mathrm{TYPE}\ \{\ \mathit{expr}\ \} \\
 & | & \mathrm{ID} : \mathrm{TYPE}\ [\ \texttt{<-}\ \mathit{expr}\ ] \\
\mathit{formal} & ::= & \mathrm{ID} : \mathrm{TYPE} \\
\mathit{expr} & ::= & \mathrm{ID}\ \texttt{<-}\ \mathit{expr} \\
 & | & \mathit{expr}[@\mathrm{TYPE}].\mathrm{ID}(\ [\ \mathit{expr}\ \llbracket, \mathit{expr} \rrbracket^{*}\ ]\ ) \\
 & | & \mathrm{ID}(\ [\ \mathit{expr}\ \llbracket, \mathit{expr} \rrbracket^{*}\ ]\ ) \\
 & | & \textbf{if}\ \mathit{expr}\ \textbf{then}\ \mathit{expr}\ \textbf{else}\ \mathit{expr}\ \textbf{fi} \\
 & | & \textbf{while}\ \mathit{expr}\ \textbf{loop}\ \mathit{expr}\ \textbf{pool} \\
 & | & \{\ \llbracket \mathit{expr}; \rrbracket^{+} \} \\
 & | & \textbf{let}\ \mathrm{ID} : \mathrm{TYPE}\ [\ \texttt{<-}\ \mathit{expr}\ ]\ \llbracket, \mathrm{ID} : \mathrm{TYPE}\ [\ \texttt{<-}\ \mathit{expr}\ ] \rrbracket^{*}\ \textbf{in}\ \mathit{expr} \\
 & | & \textbf{case}\ \mathit{expr}\ \textbf{of}\ \llbracket \mathrm{ID} : \mathrm{TYPE} => \mathit{expr}; \rrbracket^{+} \textbf{esac} \\
 & | & \textbf{new}\ \mathrm{TYPE} \\
 & | & \textbf{isvoid}\ \mathit{expr} \\
 & | & \mathit{expr} + \mathit{expr} \\
 & | & \mathit{expr} - \mathit{expr} \\
 & | & \mathit{expr} * \mathit{expr} \\
 & | & \mathit{expr} / \mathit{expr} \\
 & | & \tilde{}\,\mathit{expr} \\
 & | & \mathit{expr} < \mathit{expr} \\
 & | & \mathit{expr} <= \mathit{expr} \\
 & | & \mathit{expr} = \mathit{expr} \\
 & | & \textbf{not}\ \mathit{expr} \\
 & | & (\mathit{expr}) \\
 & | & \mathrm{ID} \\
 & | & \mathrm{integer} \\
 & | & \mathrm{string} \\
 & | & \textbf{true} \\
 & | & \textbf{false}
\end{array}
$$

Figure 1: Cool syntax.