The Definition of Snail Programming Language

letexpr

2020 年 7 月 11 日

1 Snail の構文定義

EBNF 記法を用いて Snail の具象構文を以下に示す.

$$
\begin{aligned}
\textit{toplevel} &::= \textit{let}\ [\textit{rec}]\ \textit{var}\ \{\textit{var}\ [\ :\ \langle \textit{type} \rangle]\} : \langle \textit{type} \rangle = \langle \textit{term} \rangle\ \{\langle \textit{mutual-recursion-top-let} \rangle\} \\
&\mid\ \textit{typedef}\ \textit{cons}\ \{\textit{var}\} = [\ |\ ]\ \{\langle \textit{type-dec} \rangle\ |\ \}\ \langle \textit{type-dec} \rangle\ \{\langle \textit{mutual-recursion-type} \rangle\} \\
\textit{mutual-recursion-type} &::= \textit{and}\ \textit{cons}\ \{\textit{var}\} = [\ |\ ]\ \{\langle \textit{type-dec} \rangle\ |\ \}\ \langle \textit{type-dec} \rangle \\
\textit{mutual-recursion-top-let} &::= \textit{and}\ \textit{var}\ \{\textit{var}\ [\ :\ \langle \textit{type} \rangle]\}\ :\ \langle \textit{type} \rangle = \langle \textit{term} \rangle \\
\textit{type-dec} &::= \textit{cons}\ [\textit{of}\ \langle \textit{type} \rangle] \\
\textit{type} &::= \langle \textit{type} \rangle \to \langle \textit{type} \rangle \\
&\mid\ !\ '[' \langle \textit{expmod} \rangle\ ']'\ '\{'\ \langle \textit{type} \rangle\ '\}' \\
&\mid\ \langle \textit{simple-type} \rangle \\
&\mid\ \langle \textit{type} \rangle\ \langle \textit{simple-type} \rangle \\
\textit{expmod} &::= \textit{int} \\
&\mid\ \infty \\
\textit{simple-type} &::= '(' \langle \textit{type} \rangle\ ')' \\
&\mid\ \textit{var} \\
&\mid\ \textit{cons} \\
&\mid\ () \\
\textit{pattern} &::= \langle \textit{simple-pattern} \rangle \\
&\mid\ \langle \textit{pattern} \rangle\ \langle \textit{simple-pattern} \rangle \\
&\mid\ \langle \textit{simple-pattern} \rangle\ \textit{binop}\ \langle \textit{simple-pattern} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\textit{simple-pattern} ::= {} & '(' \ \langle \textit{pattern} \rangle \ ')' \\
| \ & \textit{var} \\
| \ & \textit{cons} \ '[' \ \langle \textit{simple-pattern} \rangle \ ']' \\
| \ & [\,] \\
| \ & \_
\end{aligned}
$$

$$
\textit{mutual-recursion-let} ::= \textit{and} \ \textit{var} \ \{\textit{var} \ [\ : \ \langle \textit{type} \rangle]\} \ : \ \langle \textit{type} \rangle = \langle \textit{term} \rangle
$$

$$
\begin{aligned}
\textit{term} ::= {} & \langle \textit{simple-term} \rangle \\
| \ & \langle \textit{term} \rangle \ \langle \textit{simple-term} \rangle \\
| \ & \textit{let} \ [\textit{rec}] \ \textit{var} \ \{\textit{var} \ [\ : \ \langle \textit{type} \rangle]\} : \langle \textit{type} \rangle = \langle \textit{term} \rangle \ \{\langle \textit{mutual-recursion-let} \rangle\} \ \textit{in} \ \langle \textit{term} \rangle \\
| \ & \textit{fun} \ \{\textit{var} \ [\ : \ \langle \textit{type} \rangle]\} \ \rightarrow \langle \textit{term} \rangle \\
| \ & \textit{match} \ \langle \textit{term} \rangle \ \textit{with} \ [\,|\,] \ \{\langle \textit{pattern} \rangle \rightarrow \langle \textit{term} \rangle \ | \ \} \ \langle \textit{pattern} \rangle \rightarrow \langle \textit{term} \rangle \\
| \ & \textit{if} \ \langle \textit{term} \rangle \ \textit{then} \ \langle \textit{term} \rangle \ \textit{else} \ \langle \textit{term} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\textit{simple-term} ::= {} & '(' \ \langle \textit{term} \rangle \ [\ : \langle \textit{type} \rangle] \ ')' \\
| \ & ! \ \langle \textit{term} \rangle \\
| \ & \textit{int} \\
| \ & \textit{float} \\
| \ & \textit{string} \\
| \ & \textit{var} \\
| \ & \textit{cons} \ [\langle \textit{simple-term} \rangle] \\
| \ & () \\
| \ & [\,] \\
| \ & \textit{list}
\end{aligned}
$$

終端記号の意味を以下のように定義する.

- var 先頭が小文字で始まる文字列.
- cons 先頭が大文字で始まる文字列.
- list 組み込みリストの構文糖衣,[1,2,3] など.
- string 文字列リテラル.
- int 整数リテラル.
- float 小数リテラル.
- その他 予約語.

2 Snail の型システム

Snail は次のような型付け規則を持つ.

$$\frac{}{\vdash int : Int}\ (\textsc{Int}) \qquad \frac{}{\vdash float : Float}\ (\textsc{Float}) \qquad \frac{}{\vdash string : String}\ (\textsc{String})$$

$$\frac{}{x : A \vdash x : A}\ (\textsc{id}) \qquad \frac{\Gamma, x : A \vdash e : B}{\Gamma, x : [A]_1 \vdash e : B}\ (\textsc{der}) \qquad \frac{[\Gamma] \vdash e : B}{r * [\Gamma] \vdash\ !e : !_r B}\ (\textsc{pr})$$

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash fun\ x \to e : A \multimap B}\ (\textsc{fun}) \qquad \frac{\Gamma, x : [A]_r \vdash e : B}{\Gamma \vdash fun\ (!x : !_r A) \to e : !_r A \multimap B}\ (\textsc{fun-exp})$$

$$\frac{\Gamma \vdash e : A \multimap B \quad \Delta \vdash e' : A}{\Gamma + \Delta \vdash e\ e' : B}\ (\textsc{app}) \qquad \frac{\Gamma \vdash e : Bool \quad \Delta \vdash e_1 : A \quad \Delta \vdash e_2 : A}{\Gamma + \Delta \vdash if\ e\ then\ e_1\ else\ e_2 : A}\ (\textsc{if})$$

$$\frac{\Gamma \vdash e : A \quad \Delta, x : A \vdash e' : B}{\Gamma + \Delta \vdash let\ x = e\ in\ e' : B}\ (\textsc{let}) \qquad \frac{\Gamma \vdash e : !_r A \quad \Delta, x : [A]_r \vdash e' : B}{\Gamma + \Delta \vdash let\ !x = e\ in\ e' : B}\ (\textsc{let-exp})$$

$$\frac{[\Gamma], x : [A]_p \vdash e : A \quad \Delta, x : !_\infty A \vdash e' : B}{\infty * [\Gamma] + \Delta \vdash let\ rec\ x = e\ in\ e' : B}\ (\textsc{let-rec})$$