The Definition of Snail Programming Language

letexpr

2020 年 8 月 4 日

1 はじめに

Snail は静的型付けの関数型プログラミング言語である.

主な特徴として,

- Bounded Linear Type によるリソースの制御
- Effect System / Coeffect System (未実装)
- 軽量な依存型 (indexed type) (未実装)

が挙げられる.

本文では Snail について Core 言語を定義し,Core 言語への脱糖規則,Core 言語の型付け規則, 操作的意味論を定義することにより Snail に定義を与える.

本文中ではメタ変数として以下のようなものを用いる.

- $\Gamma, \Delta, \Theta\ \cdots$ 型環境上を動くメタ変数.
- $A, B, C \cdots$ 型の上を動くメタ変数.
- $K \cdots$ コンストラクタ上を動くメタ変数.
- $x, y, z \cdots$ 変数上を動くメタ変数.
- $p \cdots$ パターン上を動くメタ変数.
- $i, j, k \cdots$ resource semiring 上を動くメタ変数.
- $e \cdots$ 項の上を動くメタ変数.
- $c\ \cdots$ 定数上を動くメタ変数.
- $n\ \cdots$ 自然数上を動くメタ変数.

2 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{array}{rcl}
\text{toplevel} & ::= & \text{let [rec] x } \{\text{y [: } \langle\text{type}\rangle]\}\text{: } \langle\text{type}\rangle = \langle\text{term}\rangle\ \{\langle\text{mutual-recursion-let}\rangle\} \\
& | & \text{typedef A } = [\ |\ ]\ \{\langle\text{K [of } \langle\text{type}\rangle]\rangle \mid \}\ \langle\text{K [of } \langle\text{type}\rangle]\rangle\ \{\langle\text{mutual-recursion-type}\rangle\} \\
\\
\text{mutual-recursion-type} & ::= & \text{and A } = [\ |\ ]\ \{\langle\text{K [of } \langle\text{type}\rangle]\rangle \mid \}\ \langle\text{K [of } \langle\text{type}\rangle]\rangle \\
\\
\text{type} & ::= & \langle\text{type}\rangle \to \langle\text{type}\rangle \\
& | & \text{! '[' } \langle\text{expmod}\rangle \text{ ']' '\{' } \langle\text{type}\rangle \text{ '\}'} \\
& | & \langle\text{type}\rangle\ \langle\text{type}\rangle \\
& | & \text{'(' } \langle\text{type}\rangle \text{ ')'} \\
& | & \text{A} \\
\\
\text{expmod} & ::= & \text{n} \mid \infty \\
\\
\text{pattern} & ::= & \langle\text{pattern}\rangle \text{ binop } \langle\text{pattern}\rangle \\
& | & \text{'(' } \langle\text{pattern}\rangle \text{ ')'} \\
& | & \text{x} \mid \text{K } (\{\langle\text{pattern}\rangle\ ,\}\langle\text{pattern}\rangle) \\
& | & \{\langle\text{pattern}\rangle\ ,\}\langle\text{pattern}\rangle) \\
& | & \mathit{list} \qquad \text{(組み込みリストの構文糖衣)} \\
& | & \_ \\
\\
\text{mutual-recursion-let} & ::= & \text{and x } \{\text{y [: } \langle\text{type}\rangle]\}\text{ : } \langle\text{type}\rangle = \langle\text{term}\rangle \\
\\
\text{term} & ::= & \langle\text{term}\rangle\ \langle\text{term}\rangle \\
& | & \text{let [rec] x } \{\text{y [: } \langle\text{type}\rangle]\}\text{: } \langle\text{type}\rangle = \langle\text{term}\rangle\ \{\langle\text{mutual-recursion-let}\rangle\} \text{ in } \langle\text{term}\rangle \\
& | & \text{fun } \{\text{x [: } \langle\text{type}\rangle]\} \to \langle\text{term}\rangle \\
& | & \text{match } \langle\text{term}\rangle \text{ with } [\ |\ ]\ \{\langle\text{pattern}\rangle \to \langle\text{term}\rangle \mid \}\ \langle\text{pattern}\rangle \to \langle\text{term}\rangle \\
& | & \text{if } \langle\text{term}\rangle \text{ then } \langle\text{term}\rangle \text{ else } \langle\text{term}\rangle \\
& | & \text{fix x.}\langle\text{term}\rangle \\
& | & \text{'(' } \langle\text{term}\rangle \text{ [: } \langle\text{type}\rangle\text{] ')'} \\
& | & \text{! } \langle\text{term}\rangle \\
& | & \text{K } (\{\langle\text{term}\rangle,\}\langle\text{term}\rangle) \\
& | & (\{\langle\text{term}\rangle,\}\langle\text{term}\rangle) \mid \text{c} \mid \text{x} \mid \text{list}
\end{array}
$$

3 Snail の Core 言語

Snail の Core 言語は Snail のプログラムを脱糖する事により得ることができる.

3.1 Core 言語の構文

Core 言語は次のような構文を持つ.

$$
\begin{aligned}
e ::= &\ \text{let } !x = e_1\ in\ e_2 \\
&\mid\ K\ \mid\ x\ \mid\ !e\ \mid\ (e_1, e_2, \ldots, e_n) \\
&\mid\ \text{match } e \text{ with } \{p_n \to e_n\ \}_{n=1}^{m} \\
&\mid\ e_1\ e_2\ \mid\ \lambda x.e\ \mid\ \text{fix } x.e \\
\\
p ::= &\ K\ (x_1, x_2, \cdots, x_n)\ \mid\ (x_1, x_2, \cdots, x_n)\ \mid\ x \\
\\
\Gamma ::= &\ \emptyset\ \mid\ \Gamma, x\colon [A]_i \\
\\
A ::= &\ K\ \mid\ A \multimap A\ \mid\ !_r A
\end{aligned}
$$

3.2 Core 言語の型システム

Core 言語の型付け規則を次に示す.

3.2.1 Context と演算の定義

Context 間の加算 $+$ を次のように定義する.

$$
\begin{aligned}
\emptyset + \Delta &= \Delta \\
(x : [A]_i, \Gamma) + (x : [A]_j, \Delta) &= x : [A]_{i+j}, (\Gamma + \Delta) \\
(x : [A]_i, \Gamma) + \Delta &= x : [A]_i, (\Gamma + \Delta) \qquad \text{if } x \notin \text{dom}(\Delta)
\end{aligned}
$$

同様に,Context と自然数の乗算 $\star$ を次のように定義する.

$$
\begin{aligned}
i \star \emptyset &= \emptyset \\
i \star (x : [A]_j, [\Gamma]) &= x : [A]_{i \star j}, i \star [\Gamma]
\end{aligned}
$$

$[\Gamma]$ と表記した際,Context Γ 内には線形な変数は含まれない.

3.2.2 部分型付け規則

$$\frac{}{A <: A}\ (\textsc{o-i})$$

$$\frac{A <: B \quad j \preceq i}{[A]_i <: [B]_j}\ (\textsc{o-d})$$

$$\frac{A <: B \quad j \preceq i}{!_i A <: !_j B}\ (\textsc{o-b})$$

$$\frac{}{\Gamma <: \Gamma}\ (\textsc{o-ic})$$

$$\frac{A' <: A \quad B <: B'}{A \multimap B <: A' \multimap B'}\ (\textsc{o-l})$$

$$\frac{\Gamma <: \Delta \quad A <: B}{\Gamma, x : A <: \Delta, x : B}\ (\textsc{o-c})$$

3.2.3 型付け規則

$$\frac{\mathrm{TC}(c) = A}{\vdash c : A}\ (\textsc{const})$$

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x.e : A \multimap B}\ (\textsc{abs})$$

$$\frac{}{x : A \vdash x : A}\ (\textsc{id})$$

$$\frac{\Gamma \vdash e : A \multimap B \quad \Delta \vdash e' : A}{\Gamma + \Delta \vdash e\ e' : B}\ (\textsc{app})$$

$$\frac{[\Gamma] \vdash e : B}{i \star [\Gamma] \vdash !e : !_i B}\ (\textsc{pr})$$

$$\frac{[\Gamma], x : [A]_i \vdash e : A \quad 1 + i \star j \preceq j}{j \star [\Gamma] \vdash \mathrm{fix}\ x.e : A}\ (\textsc{fix})$$

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma, x : [A]_1 \vdash e : B}\ (\textsc{der})$$

$$\frac{\Delta \vdash e : B \quad \Gamma <: \Delta}{\Gamma, \Theta \vdash e : B}\ (\textsc{sub})$$

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

TC は値コンストラクタについての型環境を表している.

3.2.4 アルゴリズム的型付け規則

$$\frac{}{\vdash i \downarrow \mathrm{Int}; \phi} \quad (\textsc{int})$$

$$\frac{}{\vdash f \downarrow \mathrm{Float}; \phi} \quad (\textsc{float})$$

$$\frac{}{\vdash s \downarrow \mathrm{String}; \phi} \quad (\textsc{string})$$

$$\frac{}{\vdash b \downarrow \mathrm{Bool}; \phi} \quad (\textsc{bool})$$

$$\frac{}{x : A \vdash x \uparrow A; x : [A]_1} \quad (\textsc{id})$$

$$\frac{[\Gamma_1] \vdash e \downarrow B; [\Gamma_2]}{r \star [\Gamma_1] \vdash !e \downarrow !_r B; r \star [\Gamma_2]} \quad (\textsc{pr})$$

$$\frac{\Gamma_1, x : A \vdash e \downarrow B; \Gamma_2}{\Gamma_1, x : [A]_1 \vdash e \downarrow B; \Gamma_2} \quad (\textsc{der})$$

$$\frac{\Delta \vdash e \downarrow B; \Gamma_2 \qquad \Gamma_1 <: \Delta}{\Gamma_1, \Theta \vdash e \downarrow B; \Gamma_2} \quad (\textsc{sub})$$

$$\frac{\Gamma_1, x : A \vdash e \downarrow B; \Gamma_2}{\Gamma_1 \vdash \lambda x.e \downarrow A \multimap B; \Gamma_2} \quad (\textsc{abs})$$

$$\frac{\Gamma_1 \vdash e \uparrow A \multimap B; \Gamma_2 \qquad \Gamma_1 - \Gamma_2 \vdash e' \downarrow B; \Gamma_3}{\Gamma_1 \vdash e\ e' \downarrow B; \Gamma_2 + \Gamma_3} \quad (\textsc{app})$$

$$\frac{[\Gamma_1], x : [A]_p \vdash e \downarrow A; [\Gamma_2] \qquad 1 + p \star q \preceq q}{q \star [\Gamma_1] \vdash \mathrm{fix}\ x.e \downarrow A; q \star [\Gamma_2]} \quad (\textsc{fix})$$

$$\frac{\Gamma_1 \vdash e \uparrow !_r A; \Gamma_2 \qquad \Gamma_1 - \Gamma_2, x : [A]_r \vdash e' \downarrow B; \Gamma_3}{\Gamma_1 \vdash \mathrm{let}\ !x = e\ \mathrm{in}\ e' \downarrow B; \Gamma_2 + \Gamma_3} \quad (\textsc{let})$$

4 参考文献