

Hat 言語の仕様

島 和之

2023 年 11 月 2 日

Hat は Scheme と同様に静的スコープを持つ動的型付けのプログラミング言語である。ただし、その評価戦略は Scheme とは異なり、名前呼びである。つまり、引数として与えられた式の評価が必要な場合、呼び出された関数で明示的に評価する必要がある。一方、使わない引数を実行する無駄を省くことができる。また、プログラマが遅延評価を用いた独自の制御構造を実現できる。

1 形式的構文

この節では、Hat の形式的構文を示す。記述を簡潔にするため、BNF を以下のように拡張する。

- $\langle \text{thing} \rangle^*$ は $\langle \text{thing} \rangle$ の 0 個以上の出現を意味する。
- $\langle \text{thing} \rangle^+$ は $\langle \text{thing} \rangle$ の 1 個以上の出現を意味する。

上記のように拡張した BNF で、Hat プログラムの構文 $\langle \text{program} \rangle$ の生成規則を以下に示す。

```
 $\langle \text{program} \rangle \rightarrow \langle \text{includer} \rangle^* \langle \text{definition} \rangle^*$   
 $\langle \text{includer} \rangle \rightarrow (\text{include } \langle \text{string} \rangle)$   
 $\langle \text{definition} \rangle \rightarrow$   
   $(\text{defun } \langle \text{symbol} \rangle \langle \text{hat function} \rangle)$   
 $\langle \text{hat function} \rangle \rightarrow \wedge \langle \text{head} \rangle \langle \text{body} \rangle$   
   $| \wedge \langle \text{head} \rangle \langle \text{body} \rangle \langle \text{hat function} \rangle$   
 $\langle \text{head} \rangle \rightarrow (\langle \text{symbol} \rangle^*)$   
 $\langle \text{body} \rangle \rightarrow \langle \text{expression} \rangle^+$   
 $\langle \text{hat function} \rangle \rightarrow \wedge \langle \text{head} \rangle \langle \text{body} \rangle$   
 $\langle \text{head} \rangle \rightarrow (\langle \text{symbol} \rangle^*)$   
 $\langle \text{body} \rangle \rightarrow \langle \text{expression} \rangle^+$   
   $| \langle \text{expression} \rangle^+ \langle \text{hat function} \rangle$   
 $\langle \text{hat function} \rangle \rightarrow \wedge \langle \text{head} \rangle \langle \text{body} \rangle$   
 $\langle \text{head} \rangle \rightarrow (\langle \text{symbol} \rangle^*)$   
 $\langle \text{body} \rangle \rightarrow \langle \text{command} \rangle | \langle \text{command} \rangle \langle \text{hat function} \rangle$   
 $\langle \text{command} \rangle \rightarrow \langle \text{expression} \rangle^+$   
 $\langle \text{hat function} \rangle \rightarrow \wedge \langle \text{head} \rangle \langle \text{body} \rangle$   
 $\langle \text{head} \rangle \rightarrow (\langle \text{symbol} \rangle^*)$   
 $\langle \text{body} \rangle \rightarrow \langle \text{command} \rangle | \langle \text{command} \rangle \langle \text{hat function} \rangle$   
 $\langle \text{command} \rangle \rightarrow \langle \text{operator} \rangle \langle \text{operand} \rangle^*$   
 $\langle \text{operator} \rangle \rightarrow \langle \text{expression} \rangle$   
 $\langle \text{operand} \rangle \rightarrow \langle \text{expression} \rangle$   
 $\langle \text{expression} \rangle \rightarrow$   
   $\langle \text{symbol} \rangle | \langle \text{datum} \rangle | (\langle \text{hat function} \rangle)$ 
```

上記で未定義の記号 $\langle \text{variable} \rangle$, $\langle \text{literal} \rangle$, $\langle \text{lambda expression} \rangle$ の定義は R5RS と同じである。 $\langle \text{variable} \rangle$ は変数を示す。 $\langle \text{literal} \rangle$ は真偽値、数値、文字列、quote を付けたシンボルやリストなどを示す。 $\langle \text{lambda expression} \rangle$ はラムダ式を示す。

2 形式的意味論

この節では Hat プログラムに対する形式的な表示の意味論を定める。以下のように記号を定義する。

I	\in	Ide	識別子 (変数)
E	\in	Exp	式
ρ	\in	$U = \text{Ide} \rightarrow L$	環境
κ	\in	$K = E^* \rightarrow C$	式の継続

v, v_1, v_2, \dots	\in	$\langle \text{hat variable} \rangle$
f, f_1, f_2, \dots	\in	$\langle \text{hat function} \rangle$
c, c_1, c_2, \dots	\in	$\langle \text{hat call} \rangle$
e, e_1, e_2, \dots	\in	$\langle \text{hat expression} \rangle$
L, L_1, L_2, \dots	\in	$\langle \text{lambda expression} \rangle$

$(\text{defineCPS } v \ f)$ は $(v, f) \in \rho$ を意味する。ここで、 ρ は変数に対応する関数を示す環境であり、 $\langle \text{hat variable} \rangle$ と $\langle \text{hat function} \rangle$ からなる二項組の集合として定義される。以下、 $\mathcal{H}[e]$ は Hat の式 e に対する数学的意味を返す意味関数、 $\mathcal{E}[e]$ は Scheme の式 e に対する数学的意味を返す意味関数、 $\mathcal{F}[e]$ は式 e に含まれる自由変数の集合とする。

$\langle \text{hat variable} \rangle$ の意味は次のように定義される。

$$\mathcal{H}[v] = \begin{cases} \mathcal{H}[(f)] & \text{if } ((v, \exists f) \in \rho) \\ \text{wrong "undefined variable"} & \text{otherwise} \end{cases}$$

$\langle \text{hat function} \rangle$ の意味は以下のように定義される。

$$\mathcal{H}[(f)] = \begin{cases} \lambda x.x(\lambda v_1.\mathcal{H}[(\wedge (v_2 \ \dots \ v_n) \ c)]) & \text{if } (f = \wedge (v_1 \ v_2 \ \dots \ v_n) \ c) \wedge (n \geq 1) \\ \mathcal{H}[(c)] & \text{if } (f = \wedge () \ c) \\ \lambda x.x(\lambda v_1.\mathcal{H}[(\wedge (v_2 \ \dots \ v_n \ . \ v_{n+1}) \ c)]) & \text{if } (f = \wedge (v_1 \ v_2 \ \dots \ v_n \ . \ v_{n+1}) \ c) \wedge (n \geq 1) \\ \lambda x.x(\lambda v_1.\mathcal{H}[(\wedge \ v_2 \ c)]) & \text{if } (f = \wedge (v_1 \ . \ v_2) \ c) \\ \lambda v.\mathcal{H}[(c)]v & \text{if } (f = \wedge \ v \ c) \end{cases}$$

$\langle \text{hat call} \rangle$ の意味は以下のように定義される.

$$\mathcal{H}[\![c]\!] = \left\{ \begin{array}{l} \lambda x.x(\mathcal{E}[\![L]\!]e_1e_2\cdots e_n) \\ \quad \text{if } (c = L \ e_1 \ e_2 \ \cdots \ e_n) \\ \mathcal{H}[\![(f) \ e_1 \ e_2 \ \cdots e_n]\!] \\ \quad \text{if } (c = v \ e_1 \ e_2 \ \cdots \ e_n) \\ \quad \wedge (n \geq 0) \wedge ((v, \exists f) \in \rho) \\ \mathcal{H}[\![(f) \ e_1 \ e_2 \ \cdots e_{n-1}]\!](\lambda x.xe_n) \\ \quad \text{if } (c = (f) \ e_1 \ e_2 \ \cdots \ e_n) \wedge (n \geq 1) \\ \mathcal{H}[\!(f)\!] \quad \text{if } (c = (f)) \\ \mathcal{H}[\![c_1 \hat{\ } (v) \ v \ e_1 \ e_2 \ \cdots e_n]\!] \\ \quad \text{if } (c = (c_1) \ e_1 \ e_2 \ \cdots \ e_n) \\ \quad \wedge (n \geq 1) \wedge v \notin \mathcal{F}[\![c]\!] \\ \mathcal{H}[\!(c_1)\!] \quad \text{if } (c = (c_1)) \\ \lambda x.\mathcal{H}[\!(\kappa_1 \ e_1 \ e_2 \ \cdots \ e_n)\!]x_1 \\ \quad \text{if } (c = (\mathbf{F}.\mathbf{C} \ \kappa_1 \ . \ x_1) \ e_1 \ e_2 \ \cdots \ e_n) \\ \mathcal{H}[\!(e_1 \ e_2 \ \cdots \ e_n \ . \ (f))\!] \\ \quad \text{if } (c = e_1 \ e_2 \ \cdots \ e_n \ f) \wedge (n \geq 1) \\ \lambda x.\mathcal{H}[\!(e_1 \ e_2 \ \cdots \ e_n)\!](\mathbf{F}.\mathbf{C} \ \kappa \ . \ x) \\ \quad \text{if } (c = e_1 \ e_2 \ \cdots \ e_n \ . \ \kappa) \wedge (n \geq 1) \end{array} \right.$$