ソフトウェア基礎論配布資料 (11)
継続

五十嵐 淳
京都大学 大学院情報学研究科知能情報学専攻
e-mail: igarashi@kuis.kyoto-u.ac.jp

平成 23 年 9 月 14 日

本章では，MLi の評価のための EvalMLi とは異なる規則を持つ導出システムを与える．この新しい導出システムの特徴は，全ての評価規則の前提がひとつ (厳密には，評価判断の形をした前提はひとつ) であることである．そのため，このシステムでの評価判断の導出は，枝分かれを持つ木構造ではなく，(ほぼ) 線形の構造を持つことになり，与えられた結論から導出を構成する場合，いわば「上に向かって一直線に」導出を記述することができる．

この,「上に向かって一直線に導出を記述する」という過程を，状態機械の遷移だと思うことで，与えられた式の値を計算する (抽象) 機械の定義を与えることができる．この時，状態の表現として (評価) 判断を使うことができる．

この新しい導出システムは,「残りの作業」「計算の続き」を表現する**継続**(continuation) を形式化して明示的に扱うことで得ることができる．この継続をプログラム内部で扱うことができるような構文を導入することで，例外処理を行うプログラムを元のプログラムの構造を大きく変更することなく記述できることを見る．

1 評価判断の導出を作る，ということについて

EvalMLi では，式 $\boldsymbol{e}$ (と環境 $\boldsymbol{\mathcal{E}}$) さえ与えられれば，$\boldsymbol{e} \Downarrow \boldsymbol{v}$ (もしくは $\boldsymbol{\mathcal{E}} \vdash \boldsymbol{e} \Downarrow \boldsymbol{v}$) の導出 (があれば，それ) を手続的・機械的に構成することができる．その手続きは大体以下のようにまとめられる．

- 与えられた式の形で，どの規則を使うか決定する．
- 規則の一番左の前提にある判断の導出を (再帰呼び出しによって) 得る．
- 次の前提の判断の導出を (再帰呼び出しによって) 得る．
- 全体の結論となる判断を得る．

(このように左の前提から順に処理をすすめていけばいいのは，実は，意図的に規則がそのように—上の手続きで導出が構成できるように—書かれているからである．規則の前提の順番が違えば，必ずしも左の前提から導出を得ていくのが得策とは限らない．) ただし，各ステップの間でも，ちょっとした作業を伴なう場合がある．例えば，EvalML2 での E-Let では，$\boldsymbol{e_2}$ の値を得るための環境を ($\boldsymbol{e_1}$ の値を使って) 構成する必要がある．また，E-Plus では，$\boldsymbol{e_1}, \boldsymbol{e_2}$ の値を得てから，その和を計算する必要がある．

2 継続—残りの作業リスト

さて，**継続**(continuation) とは，計算プロセス・手続き的作業の (各時点における) 残りの計算，残りの作業のことである．TODO リスト (これからやることのリスト) といってもよいかもしれない．この TODO リストは，作業を完了することで縮んだり，作業をさらに細かい作業列に分割することで伸びたりもするものである．

例えば，EvalML1 で $(\boldsymbol{e_1}$ + $\boldsymbol{e_2}) * \boldsymbol{e_3} \Downarrow \boldsymbol{v}$ の導出を作ることを考える．

$(\boldsymbol{e}_1$ + $\boldsymbol{e}_2) * \boldsymbol{e}_3$ evalto $\boldsymbol{v}$ by E-Times {

ここまで書いたところでの継続は

1. $\boldsymbol{e}_1$ + $\boldsymbol{e}_2$ evalto $\boldsymbol{v_1}$ の導出を作る
2. $\boldsymbol{e_3}$ evalto $\boldsymbol{v_3}$ の導出を作る
3. $\boldsymbol{v_1}$ times $\boldsymbol{v_3}$ is $\boldsymbol{v}$ の導出を作る

である．もう少し進めて,

$(\boldsymbol{e}_1$ + $\boldsymbol{e}_2) * \boldsymbol{e}_3$ evalto $\boldsymbol{v}$ by E-Times {
  $\boldsymbol{e}_1$ + $\boldsymbol{e}_2$ evalto $\boldsymbol{v_1}$ by E-Plus {

まで書いたところでの継続は,

1. $\boldsymbol{e}_1$ + $\boldsymbol{e}_2$ evalto $\boldsymbol{v_1}$ の導出の残りを作る，すなわち,
   (a) $\boldsymbol{e}_1$ evalto $\boldsymbol{v_{11}}$ の導出を作る
   (b) $\boldsymbol{e_2}$ evalto $\boldsymbol{v_{12}}$ の導出を作る
   (c) $\boldsymbol{v_{11}}$ plus $\boldsymbol{v_{12}}$ is $\boldsymbol{v_1}$ の導出を作る
2. $\boldsymbol{e_3}$ evalto $\boldsymbol{v_3}$ の導出を作る
3. $\boldsymbol{v_1}$ times $\boldsymbol{v_3}$ is $\boldsymbol{v}$ の導出を作る

となる．さらに，もう少し進めて，$\boldsymbol{e}_1$ evalto $\boldsymbol{v_{11}}$ の導出が完成し,

(e_1 + e_2) * e_3 evalto v by E-Times {

  e_1 + e_2 evalto v_1 by E-Plus {

    e_1 evalto v_{11} by ... { ... };

まで書いた時点での継続は,

1. e_1 + e_2 evalto v_1 の導出の残りを作る，すなわち,
   (a) e_2 evalto v_{12} の導出を作る
   (b) v_{11} plus v_{12} is v_1 の導出を作る
2. e_3 evalto v_3 の導出を作る
3. v_1 times v_3 is v の導出を作る

である．このように，導出を書き進める度に継続が変化してゆく．

この作業の中から純粋に，式から値を得ることに関係ある作業を抽出すると，最後の継続は

1. 式 e_1 + e_2 の値を得る作業の残り，すなわち,
   (a) 式 e_2 の値 v_{12} を得る,
   (b) v_{11} と前ステップで得られた値 (つまり v_{12}) の和 v_1 を計算する．
2. 式 e_3 の値 v_3 を得る,
3. v_1 と前ステップで得られた値 (つまり v_3) の積 v を計算する．

という計算作業リストである，と捉えることもできる．

これから与える導出システム EvalContML1 では，このような継続を明示的に表示した

$$e \gg k \Downarrow v$$

という評価判断を考える．この判断の内容は,

> e の値に対して残りの計算 k を行うと値 v が得られる

であるとともに，EvalML1 で導出を書いている作業途中の

> 現在 e evalto v' の導出を書こうとしており，その後の作業として k が残っている．(そして，導出全体が書きあがった時，その結論の値は v である．)

という状態を表している．

継続 k は，残りの計算であるが，上の例で箇条書きで与えたような細かい作業単位の列でもある．EvalML1 に関しては，作業単位として,

- 二項演算式 $e_1\ op\ e_2$ の左の引数式 e_1 を評価している時の残りである，

  二項演算の右の引数式 e_2 を評価して，現在評価している式の値との演算を行う，

  という作業

- 二項演算式の右の引数式を評価している時の残りである，

  左の引数式の値 v_1 と現在評価している式の値を演算する

  という作業

- `if` 式 (`if` e_1 `then` e_2 `else` e_3) の条件式 e_1 を評価している時の残りである，

  現在評価している式の値に従って `then` 節 (e_2) または `else` 節 (e_3) の式のどちらかを評価する

  という作業

を考えれば充分である．これらを，EvalContML1 では，それぞれ以下のように表記する．

- $\{\_\ op\ e_2\}$
- $\{v_1\ op\ \_\}$
- $\{\texttt{if}\ \_\ \texttt{then}\ e_2\ \texttt{else}\ e_3\}$

“_” は「現在評価中の式の値」を示している記号である．これを $\gg$ で区切って並べたものを継続の記号表現として用いる．

3 導出システム EvalContML1

以下が，導出システム EvalContML1 で扱うものの BNF 定義である．

$i \in \mathbf{int}$
$b \in \mathbf{bool}$
$v \in \mathbf{Value} ::= i \mid b$
$e \in \mathbf{Exp} ::= i \mid b \mid e\ op\ e \mid \texttt{if}\ e\ \texttt{then}\ e\ \texttt{else}\ e$
$op \in \mathbf{Prim} ::= \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{<}$
$k \in \mathbf{Cont} ::= \_ \mid \{\_\ op\ e\} \gg k \mid \{v\ op\ \_\} \gg k \mid \{\texttt{if}\ \_\ \texttt{then}\ e\ \texttt{else}\ e\} \gg k$

継続の定義の最初にある “_” のみの継続は，TODO リストの末尾，すなわち，これ以上残りの作業がないこと，もしくは，空の計算を示すものである．

この導出システムの主な判断は以下のふたつである．

- $e \gg k \Downarrow v$
- $v_1 \Rightarrow k \Downarrow v_2$

それぞれ，演習システム上では e `>>` k `evalto` v と v_1 `=>` k `evalto` v_2 と書かれる．なお，継続の最後にある “`>> _`” は省略してよく，e `evalto` v や，e `>>` $\{\cdots\}$ `evalto` v などと略記してよい．

前者の判断の意味は既に説明した通りである．後者は，

> 値 v_1 に対して残りの計算 k を行うと値 v_2 が得られる

という意味であり，ひとつの (部分) 導出が完成し，値が得られた瞬間の状態を示している．

3.1 EvalContML1 の推論規則

推論規則を見ていこう．これらの規則は，通常通り，前提の判断が成立する (導出できる) ならば結論が成立する，と読むこともできるが，判断を「EvalML1 で導出を記述している最中の状態」を示すものと解釈すると，規則は作業の単位であり，結論が作業前，前提が作業後の状態を示している，とも捉えられる．

まず，E-Int，E-Bool は以下のように与えられる．

$$\frac{i \Rightarrow k \Downarrow v}{i \gg k \Downarrow v} \quad \text{(E-Int)}$$

$$\frac{b \Rightarrow k \Downarrow v}{b \gg k \Downarrow v} \quad \text{(E-Bool)}$$

i, b は，結論では式，前提では値であることに注意すること．規則の通常の読み方をすれば，

> 値 i に対して残りの計算をすると v が得られる時，式 i に対して同じ残りの計算をしても同じ値が得られる

と読める．別の解釈では，式 i を評価して，値 i を得るという作業に相当している．

これらは EvalML1 では前提のない規則，すなわち，これ以上の作業をすることなしに導出が完成する規則であった．このような，それが完了すると値が得られる作業を表す規則の前提は，一般に $v_1 \Rightarrow k \Downarrow v_2$ の形である．

次に，二項演算式の評価は以下の式で与えられる．

$$\frac{e_1 \gg \{\_\ op\ e_2\} \gg k \Downarrow v}{e_1\ op\ e_2 \gg k \Downarrow v} \quad \text{(E-BinOp)}$$

これは，作業を分割して，右側の引数式の評価は継続に追加して左側の引数式の評価に入ることを示している．E-If も同様である．

$$\frac{e_1 \gg \{\texttt{if \_ then } e_2 \texttt{ else } e_3\} \gg k \Downarrow v}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \gg k \Downarrow v} \quad \text{(E-If)}$$

残りの規則は，判断 $v_1 \Rightarrow k \Downarrow v_2$ を結論とする規則である．これらは，継続 k の形に応じて規則が与えられている．まず，最初の規則は「もうこれ以上する作業がない」場合の規則である．

$$\frac{}{v \Rightarrow \_ \Downarrow v} \quad \text{(C-Ret)}$$

この場合は，今与えられた値が作業全体を通じて得られるべき値となる．

次の規則 C-EvalR

$$\frac{e \gg \{v_1\ op\ \_\} \gg k \Downarrow v_2}{v_1 \Rightarrow \{\_\ op\ e\} \gg k \Downarrow v_2} \quad \text{(C-EvalR)}$$

は，二項演算子の左の引数の評価が終わったところから，右の引数の評価に移るために，左の引数の値と継続にあった式を交換する作業を示している．

次の 4 つの規則は，二項演算子の右の引数式の評価が終了し，実際の演算を行う作業を表している．

$$\frac{i_1 \texttt{ plus } i_2 \texttt{ is } i_3 \qquad i_3 \Rightarrow k \Downarrow v}{i_2 \Rightarrow \{i_1 \texttt{ + } \_\} \gg k \Downarrow v} \quad \text{(C-Plus)}$$

$$\frac{i_1 \texttt{ minus } i_2 \texttt{ is } i_3 \qquad i_3 \Rightarrow k \Downarrow v}{i_2 \Rightarrow \{i_1 \texttt{ - } \_\} \gg k \Downarrow v} \quad \text{(C-Minus)}$$

$$\frac{i_1 \texttt{ times } i_2 \texttt{ is } i_3 \qquad i_3 \Rightarrow k \Downarrow v}{i_2 \Rightarrow \{i_1 \texttt{ * } \_\} \gg k \Downarrow v} \quad \text{(C-Times)}$$

$$\frac{i_1 \texttt{ less than } i_2 \texttt{ is } b_3 \qquad b_3 \Rightarrow k \Downarrow v}{i_2 \Rightarrow \{i_1 \texttt{ < } \_\} \gg k \Downarrow v} \quad \text{(C-Lt)}$$

これらの規則のみ前提がふたつあるが，最初の前提の導出は使用する規則がそれぞれ以下の規則のいずれかに決まっており，ふたつの規則を結合して前提を付帯条件で与えることもできたので，本質的にはひとつと考えてよい．

$$\frac{(i_3 = i_1 + i_2)}{i_1 \texttt{ plus } i_2 \texttt{ is } i_3} \quad \text{(B-Plus)}$$

$$\frac{(i_3 = i_1 - i_2)}{i_1 \texttt{ minus } i_2 \texttt{ is } i_3} \quad \text{(B-Minus)}$$

$$\frac{(\boldsymbol{i_3} = \boldsymbol{i_1} * \boldsymbol{i_2})}{\boldsymbol{i_1}\ \texttt{times}\ \boldsymbol{i_2}\ \texttt{is}\ \boldsymbol{i_3}} \quad \text{(B-TIMES)}$$

$$\frac{(\boldsymbol{b_3} = (\boldsymbol{i_1} < \boldsymbol{i_2}))}{\boldsymbol{i_1}\ \texttt{less than}\ \boldsymbol{i_2}\ \texttt{is}\ \boldsymbol{b_3}} \quad \text{(B-LT)}$$

最後の 2 つの規則は，if 式の条件式の評価が終了し，その値に応じての分岐を表現している．

$$\frac{\boldsymbol{e_1} \gg \boldsymbol{k} \Downarrow \boldsymbol{v}}{\texttt{true} \Rightarrow \{\texttt{if \_ then}\ \boldsymbol{e_1}\ \texttt{else}\ \boldsymbol{e_2}\} \gg \boldsymbol{k} \Downarrow \boldsymbol{v}} \quad \text{(C-IFT)}$$

$$\frac{\boldsymbol{e_2} \gg \boldsymbol{k} \Downarrow \boldsymbol{v}}{\texttt{false} \Rightarrow \{\texttt{if \_ then}\ \boldsymbol{e_1}\ \texttt{else}\ \boldsymbol{e_2}\} \gg \boldsymbol{k} \Downarrow \boldsymbol{v}} \quad \text{(C-IFF)}$$

3.2 EvalContML1 の特徴と式の評価のための抽象機械

この導出システムの特徴は,

- 各規則の前提が (実質的に) ひとつである．
- 前提と結論に現れる評価判断に現れる値が常に同じである．

ということである．このふたつの特徴から，与えられた結論に対しては，その導出は「一直線に」作成できることがわかる．また，導出を書く時の，判断の推移過程を状態遷移だと思うと，式を評価するための抽象的な機械の定義を得ることができる．

図 1 にそのような機械の定義を与える．途中の状態は $\langle \boldsymbol{e}, \boldsymbol{k} \rangle$ と $\langle \boldsymbol{k}, \boldsymbol{v} \rangle$ の二種類で，それぞれ第一要素によって遷移の種類が決まっている．

$$\boldsymbol{e} \implies_{\mathrm{init}} \langle \boldsymbol{e}, \_ \rangle$$

$$\begin{array}{rcl}
\langle \boldsymbol{i}, \boldsymbol{k} \rangle & \Longrightarrow & \langle \boldsymbol{k}, \boldsymbol{i} \rangle \\
\langle \boldsymbol{b}, \boldsymbol{k} \rangle & \Longrightarrow & \langle \boldsymbol{k}, \boldsymbol{b} \rangle \\
\langle \boldsymbol{e_1}\ \boldsymbol{op}\ \boldsymbol{e_2}, \boldsymbol{k} \rangle & \Longrightarrow & \langle \boldsymbol{e_1}, \{_\ \boldsymbol{op}\ \boldsymbol{e_2}\} \gg \boldsymbol{k} \rangle \\
\langle \texttt{if}\ \boldsymbol{e_1}\ \texttt{then}\ \boldsymbol{e_2}\ \texttt{else}\ \boldsymbol{e_3}, \boldsymbol{k} \rangle & \Longrightarrow & \langle \boldsymbol{e_1}, \{\texttt{if}\ _\ \texttt{then}\ \boldsymbol{e_2}\ \texttt{else}\ \boldsymbol{e_3}\} \gg \boldsymbol{k} \rangle \\
\\
\langle \{_\ \boldsymbol{op}\ \boldsymbol{e}\} \gg \boldsymbol{k}, \boldsymbol{v_1} \rangle & \Longrightarrow & \langle \boldsymbol{e}, \{\boldsymbol{v_1}\ \boldsymbol{op}\ _\} \gg \boldsymbol{k} \rangle \\
\langle \{\boldsymbol{i_1} + _\} \gg \boldsymbol{k}, \boldsymbol{i_2} \rangle & \Longrightarrow & \langle \boldsymbol{k}, \boldsymbol{i_1} + \boldsymbol{i_2} \rangle \\
\langle \{\boldsymbol{i_1} + _\} \gg \boldsymbol{k}, \boldsymbol{i_2} \rangle & \Longrightarrow & \langle \boldsymbol{k}, \boldsymbol{i_1} - \boldsymbol{i_2} \rangle \\
\langle \{\boldsymbol{i_1} + _\} \gg \boldsymbol{k}, \boldsymbol{i_2} \rangle & \Longrightarrow & \langle \boldsymbol{k}, \boldsymbol{i_1} * \boldsymbol{i_2} \rangle \\
\langle \{\boldsymbol{i_1} + _\} \gg \boldsymbol{k}, \boldsymbol{i_2} \rangle & \Longrightarrow & \langle \boldsymbol{k}, \boldsymbol{i_1} < \boldsymbol{i_2} \rangle \\
\langle \{\texttt{if}\ _\ \texttt{then}\ \boldsymbol{e_1}\ \texttt{else}\ \boldsymbol{e_2}\} \gg \boldsymbol{k}, \texttt{true} \rangle & \Longrightarrow & \langle \boldsymbol{e_1}, \boldsymbol{k} \rangle \\
\langle \{\texttt{if}\ _\ \texttt{then}\ \boldsymbol{e_1}\ \texttt{else}\ \boldsymbol{e_2}\} \gg \boldsymbol{k}, \texttt{false} \rangle & \Longrightarrow & \langle \boldsymbol{e_2}, \boldsymbol{k} \rangle
\end{array}$$

$$\langle \_, \boldsymbol{v} \rangle \implies_{\mathrm{end}} \boldsymbol{v}$$

図 1: ML1 式の評価のための抽象機械