ソフトウェア構成特論 第14回

大学院理工学研究科 電気電子情報工学専攻　篠埜 功

2016年7月21日

1　はじめに

Featherweight Java の定義を与える。

2　FJの定義

ここではFJの定義を与える。まず、メタ変数の使い方について示す。クラス名を表すメタ変数として A, B, C, D, E を用いる。フィールド名を表すメタ変数として f, g を用いる。仮引数名を表すメタ変数として x を用いる。term を表すメタ変数として s, t を用いる。値を表すメタ変数として u, v を用いる。クラス宣言を表すメタ変数として CL を用いる。コンストラクタ (constructor) を表すメタ変数として K を用いる。メソッド宣言を表すメタ変数として M を用いる。

FJでは、this は変数とするが、this はメソッドの仮引数名としては使えないものとする。

参考: Java では this は変数ではない。

また、フィールド名等の列を名前の上に線を付けて表すことにする。$\overline{\mathtt{f}}$ は $\mathtt{f}_1, \ldots, \mathtt{f}_n$ を表すとする（$\overline{\mathtt{C}}, \overline{\mathtt{x}}, \overline{\mathtt{t}}$ も同様)。また、$\overline{\mathtt{M}}$ は $\mathtt{M}_1 \ldots \mathtt{M}_n$ を表すとする（コンマ無し)。また、$\overline{\mathtt{C}}\ \overline{\mathtt{f}}$ は $\mathtt{C}_1\ \mathtt{f}_1, \ldots, \mathtt{C}_n\ \mathtt{f}_n$ を表し、$\overline{\mathtt{C}}\ \overline{\mathtt{f}};$ は $\mathtt{C}_1\ \mathtt{f}_1;, \ldots, \mathtt{C}_n\ \mathtt{f}_n;$ を表すとし、$\mathtt{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};$ は $\mathtt{this}.\mathtt{f}_1 = \mathtt{f}_1; \ldots; \mathtt{this}.\mathtt{f}_n = \mathtt{f}_n; \ldots;$ を表すとする。また、フィールド宣言列、仮引数列、メソッド宣言列において、同じ名前は使えない。

2.1　構文

以上の表記法を用いてFJの構文を以下のように定義する。

$$
\begin{array}{lll}
\mathtt{CL} & ::= & \mathtt{class\ C\ extends\ C}\ \{\overline{\mathtt{C}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \\
\mathtt{K} & ::= & \mathtt{C}(\overline{\mathtt{C}}\ \overline{\mathtt{f}})\ \{\mathtt{super}(\overline{\mathtt{f}});\ \mathtt{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};\} \\
\mathtt{M} & ::= & \mathtt{C\ m}(\overline{\mathtt{C}}\ \overline{\mathtt{x}})\ \{\mathtt{return\ t};\} \\
\mathtt{t} & ::= & \mathtt{x} \mid \mathtt{t.f} \mid \mathtt{t.m}(\overline{\mathtt{t}}) \mid \mathtt{new\ C}(\overline{\mathtt{t}}) \mid (\mathtt{C})\ \mathtt{t}
\end{array}
$$

宣言 class C extends D $\{\overline{\mathtt{C}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\}$ は、クラスCをクラスDのサブクラスとして宣言し、クラスCはいくつかのフィールド $\overline{\mathtt{f}}$（それぞれの型は $\overline{\mathtt{C}}$)、一つのコンストラクタK、いくつかのメソッド $\overline{\mathtt{M}}$ を持つ。Cで宣言されるインスタンス変数 $\overline{\mathtt{f}}$、D（やその先祖クラス）で

宣言されるインスタンス変数はすべて異なっていなければならない。一方、クラス C のメソッド名は D（やその先祖クラス）と同じ名前のものがあってもよく、その場合 override される。

参考: Java では親クラスと同じ名前のインスタンス変数を宣言してもよい。

コンストラクタ宣言 $\texttt{C}(\overline{\texttt{D}}\ \overline{\texttt{g}},\ \overline{\texttt{C}}\ \overline{\texttt{f}})\ \{\texttt{super}(\overline{\texttt{g}});\ \texttt{this.}\overline{\texttt{f}} = \overline{\texttt{f}};\}$ はクラス C のインスタンスのフィールドをどのように初期化するかを示す。コンストラクタ宣言の仮引数の個数は、インスタンス変数（先祖クラスで宣言されているものも含む）の個数と同じでなければならず、仮引数の最初の $\overline{\texttt{g}}$ の部分は親クラスのコンストラクタ呼び出しの引数として与えられ、残りの $\overline{\texttt{f}}$ の部分は自分のクラスで宣言されているインスタンス変数と同じ名前の変数が同じ順番で並んでいなければならない。

メソッド宣言 $\texttt{D m}(\overline{\texttt{C}}\ \overline{\texttt{x}})\ \{\texttt{return t;}\}$ はメソッド名が m で結果の型が D、仮引数が $\overline{\texttt{x}}$ でそれらの型が $\overline{\texttt{C}}$ であるようなメソッドを宣言する。メソッドの本体は return 文 `return t;` 一つであり、仮引数 $\overline{\texttt{x}}$ の有効範囲は t である。特別な変数 this も t の中で使える。

2.2 値

FJ の値は以下のように定義する。

$$\texttt{v} \ ::= \ \texttt{new C}(\overline{\texttt{v}})$$

2.3 部分型関係

FJ の部分型関係は以下のように定義する。

$$\frac{}{\texttt{C} \ <: \ \texttt{C}}\ \text{(S-REFL)} \qquad \frac{\texttt{C} \ <: \ \texttt{D} \quad \texttt{D} \ <: \ \texttt{E}}{\texttt{C} \ <: \ \texttt{E}}\ \text{(S-TRANS)}$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D}\ \{\ldots\}}{\texttt{C} \ <: \ \texttt{D}}\ \text{(S-CLASS)}$$

CT はクラステーブル（クラス名からクラスの宣言への写像）である。FJ のプログラムは、クラステーブル CT と評価対象の 1 つの term からなる。全てのクラスは親クラスを持ち、それはクラス宣言中の extends の部分で宣言される。Object クラスは特別で、親クラスを持たないクラスとする。

CT から直接のサブクラス関係を取得するのが S-CLASS 規則であり、S-REFL 規則, S-TRANS 規則により、直接のサブクラス関係の反射的推移的閉包として部分型関係<: が定義される。

クラステーブル CT は以下の 4 つの条件を満たすものとする。

(1) 各クラス $\texttt{C} \in dom(CT)$ に対し、$CT(\texttt{C}) = \texttt{class C}\ldots$ である。

(2) $\texttt{Object} \notin dom(CT)$

(3) CT 中に現れる Object 以外の各クラス名 C に対し $\texttt{C} \in dom(CT)$ である。

(4) CT で定められる部分型関係<: にはサイクルがない（つまり<: は反対称的である。）。

2.4 補助的な定義

これから定義する型付け規則、評価規則を定義するためにいくつか補助的な定義をする。

クラステーブルからクラスのフィールド（とその型）を取得する関数として以下のように定義される *fields* を用いる。

$$\frac{}{\mathit{fields}(\texttt{Object}) = \bullet} \text{ (FIELDS1)}$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \mathit{fields}(\texttt{D}) = \overline{\texttt{D}}\ \overline{\texttt{g}}}{\mathit{fields}(\texttt{C}) = \overline{\texttt{D}}\ \overline{\texttt{g}},\ \overline{\texttt{C}}\ \overline{\texttt{f}}} \text{ (FIELDS2)}$$

fields は Object クラスに対しては空のフィールド（• で表す）を返す。また、Object クラスはメソッドを持たないものとする (Java の Object クラスにはメソッドがあるが)。

クラス C 中のメソッド m の型を取得する関数として以下のように定義される *mtype* を用いる。

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \texttt{B m } (\overline{\texttt{B}}\ \overline{\texttt{x}})\ \{\texttt{return t;}\} \in \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m},\ \texttt{C}) = \overline{\texttt{B}} \to \texttt{B}} \text{ (MTYPE1)}$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \texttt{m is not defined in } \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m},\ \texttt{C}) = \mathit{mtype}(\texttt{m},\ \texttt{D})} \text{ (MTYPE2)}$$

また、クラス C 中のメソッド m の本体を取得する関数として以下のように定義される *mbody* を用いる。

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \texttt{B m } (\overline{\texttt{B}}\ \overline{\texttt{x}})\ \{\texttt{return t;}\} \in \overline{\texttt{M}}}{\mathit{mbody}(\texttt{m},\ \texttt{C}) = (\overline{\texttt{x}}, \texttt{t})} \text{ (MBODY1)}$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \texttt{m is not defined in } \overline{\texttt{M}}}{\mathit{mbody}(\texttt{m},\ \texttt{C}) = \mathit{mbody}(\texttt{m},\ \texttt{D})} \text{ (MBODY2)}$$

mbody の返り値は仮引数列と本体の term との対である。MTYPE2 規則、MBODY2 規則により、メソッド m がクラス C に宣言されておらず、親クラスの宣言が継承されている場合が扱われている。

メソッドをあるクラスのサブクラスで宣言してよいかどうか判定する述語として以下のように定義される *override* を用いる。

$$\frac{\mathit{mtype}(\texttt{m}, \texttt{D}) = \overline{\texttt{D}} \to \texttt{D}_0 \text{ implies } \overline{\texttt{C}} = \overline{\texttt{D}} \text{ and } \texttt{C}_0 = \texttt{D}_0}{\mathit{override}(\texttt{m}, \texttt{D}, \overline{\texttt{C}} \to \texttt{C}_0)} \text{ (OVERRIDE)}$$

$\mathit{override}(\texttt{m}, \texttt{D}, \overline{\texttt{C}} \to \texttt{C}_0)$ は、仮引数の型が $\overline{\texttt{C}}$ で返り値の型が $\texttt{C}_0$ のメソッド m を D のサブクラスで宣言する際、もし D およびその先祖クラスで同じ名前 m のメソッドが宣言されている場合に、その型が同じであることを確認する。

2.5 term の評価規則

FJ の term の評価規則を以下のように定義する。

$$\frac{\mathit{fields}(\texttt{C}) = \overline{\texttt{C}}\ \overline{\texttt{f}}}{(\texttt{new C}(\overline{\texttt{v}})).\texttt{f}_i \to \texttt{v}_i} \text{ (E-PROJNEW)}$$

$$\frac{\mathit{mbody}(\mathtt{m}, \mathtt{C}) = (\overline{\mathtt{x}}, \mathtt{t}_0)}{(\mathtt{new\ C}(\overline{\mathtt{v}})).\mathtt{m}(\overline{\mathtt{u}}) \longrightarrow [\overline{\mathtt{x}} \mapsto \overline{\mathtt{u}}, \mathtt{this} \mapsto \mathtt{new\ C}(\overline{\mathtt{v}})]\mathtt{t}_0} \text{ (E-INVKNEW)}$$

$$\frac{\mathtt{C} <: \mathtt{D}}{(\mathtt{D})(\mathtt{new\ C}(\overline{\mathtt{v}})) \longrightarrow \mathtt{new\ C}(\overline{\mathtt{v}})} \text{ (E-CASTNEW)} \qquad \frac{\mathtt{t}_0 \longrightarrow \mathtt{t}_0'}{\mathtt{t}_0.\mathtt{f} \longrightarrow \mathtt{t}_0'.\mathtt{f}} \text{ (E-FIELD)}$$

$$\frac{\mathtt{t}_0 \longrightarrow \mathtt{t}_0'}{\mathtt{t}_0.\mathtt{m}(\overline{\mathtt{t}}) \longrightarrow \mathtt{t}_0'.\mathtt{m}(\overline{\mathtt{t}})} \text{ (E-INVK-RECV)} \qquad \frac{\mathtt{t}_i \longrightarrow \mathtt{t}_i'}{\mathtt{v}_0.\mathtt{m}(\overline{\mathtt{v}}, \mathtt{t}_i, \overline{\mathtt{t}}) \longrightarrow \mathtt{v}_0.\mathtt{m}(\overline{\mathtt{v}}, \mathtt{t}_i', \overline{\mathtt{t}})} \text{ (E-INVK-ARG)}$$

$$\frac{\mathtt{t}_i \longrightarrow \mathtt{t}_i'}{\mathtt{new\ C}(\overline{\mathtt{v}},\ \mathtt{t}_i,\ \overline{\mathtt{t}}) \longrightarrow \mathtt{new\ C}(\overline{\mathtt{v}},\ \mathtt{t}_i',\ \overline{\mathtt{t}})} \text{ (E-NEW-ARG)} \qquad \frac{\mathtt{t}_0 \longrightarrow \mathtt{t}_0'}{(\mathtt{C})\mathtt{t}_0 \longrightarrow (\mathtt{C})\mathtt{t}_0'} \text{ (E-CAST)}$$

メソッド呼び出しの評価は、メソッドを受け取るオブジェクトの評価が先で、その後引数を左から順に評価し、最後にメソッドの本体の評価が行われる。キャストについては、キャストされるオブジェクトの型がキャスト先の型の部分型であることを確認し、それが成功すればキャストが除かれたオブジェクトへ 1 ステップ評価される。これは Java の意味に対応している。Java ではキャストはオブジェクト自体には一切変更を加えず、単にキャストが成功するか、あるいは失敗して例外を投げるかのいずれかである。FJ では、キャストの評価時に部分型関係の確認が失敗したら、その他に適用できる評価規則がないのでそこで評価が stuck することになる。

2.6 term に対する型付け規則

FJ の term に対する型付け規則を以下のように定義する。

$$\frac{\mathtt{x} : \mathtt{C} \in \Gamma}{\Gamma \vdash \mathtt{x} : \mathtt{C}} \text{ (T-VAR)} \qquad \frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \quad \mathit{fields}(\mathtt{C}_0) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\Gamma \vdash \mathtt{t}_0.\mathtt{f}_i : \mathtt{C}_i} \text{ (T-FIELD)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \qquad \mathit{mtype}(\mathtt{m}, \mathtt{C}_0) = \overline{\mathtt{D}} \to \mathtt{C} \qquad \Gamma \vdash \overline{\mathtt{t}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}}{\Gamma \vdash \mathtt{t}_0.\mathtt{m}(\overline{\mathtt{t}}) : \mathtt{C}} \text{ (T-INVK)}$$

$$\frac{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{D}}\ \overline{\mathtt{f}} \qquad \Gamma \vdash \overline{\mathtt{t}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}}{\Gamma \vdash \mathtt{new\ C}(\overline{\mathtt{t}}) : \mathtt{C}} \text{ (T-NEW)} \qquad \frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{D} <: \mathtt{C}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \text{ (T-UCAST)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} <: \mathtt{D} \qquad \mathtt{C} \neq \mathtt{D}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \text{ (T-DCAST)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D} \qquad \mathtt{D} \not<: \mathtt{C} \qquad \mathit{stupid\ warning}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \text{ (T-SCAST)}$$

T-INVK 規則において、メソッドの実引数の型が仮引数の型の部分型であることが確認される。T-NEW 規則において、コンストラクタの実引数の型が仮引数の型の部分型であることが確認される。

キャストに対する型付け規則が 3 つある。T-UCAST 規則はキャスト先の型がキャスト対象の term の型の supertype の場合 (up cast) であり、その場合、その term の型をキャスト先の型にする。T-DCAST 規則はキャスト先の型がキャスト対象の term の型の部分型の場合 (down cast) であり、T-SCAST はそのどちらでもない場合 (stupid cast) である。Java においては、T-SCAST 規則が使われるような term は型がつかずコンパイル時にエラーになる。FJ においては、型保存定理 (定理 1) を示すために stupid cast を許している。

□

図 1: FJ のプログラム例

2.7 メソッド宣言、クラス宣言に対する型付け規則

FJ のメソッド宣言とクラス宣言に関する型付け規則を以下のように定義する。

$$\frac{\begin{array}{cc}\overline{\mathtt{x}} : \overline{\mathtt{C}},\ \mathtt{this} : \mathtt{C} \vdash \mathtt{t}_0 : \mathtt{E}_0 & \mathtt{E}_0 <: \mathtt{C}_0 \\ CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{...\} & \mathit{override}(\mathtt{m}, \mathtt{D}, \overline{\mathtt{C}} \to \mathtt{C}_0)\end{array}}{\mathtt{C}_0\ \mathtt{m}\ (\overline{\mathtt{C}}\ \overline{\mathtt{x}})\ \{\mathtt{return\ t}_0;\}\ \mathtt{OK\ in\ C}}\ \text{(T-METHOD)}$$

$$\frac{\begin{array}{c}\mathtt{K} = \mathtt{C}(\overline{\mathtt{D}}\ \overline{\mathtt{g}},\ \overline{\mathtt{C}}\ \overline{\mathtt{f}})\{\mathtt{super}(\overline{\mathtt{g}});\ \mathtt{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};\} \\ \mathit{fields}(\mathtt{D}) = \overline{\mathtt{D}}\ \overline{\mathtt{g}} \qquad \overline{\mathtt{M}}\ \mathtt{OK\ in\ C}\end{array}}{\mathtt{class\ C\ extends\ D}\ \{\overline{\mathtt{C}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\}\ \mathtt{OK}}\ \text{(T-CLASS)}$$

2.8 例と練習問題

第 13 回の資料のプログラム例の評価過程を示す。

例 1. 図 2.8 のクラス定義のもとで、term

```
new Pair(new A(), new B()).setfst(new B())
```

を評価する。1 ステップ目は、

$$\frac{\dfrac{CT(\mathtt{Pair}) = \mathtt{class\ Pair\ extends\ Object}\ \{\ldots\overline{\mathtt{M}}\} \quad \mathtt{Pair\ setfst(Object\ newfst)\{return\ new\ Pair(newfst, this.snd)\}} \in \overline{\mathtt{M}}}{\mathit{mbody}(\mathtt{setfst}, \mathtt{Pair}) = (\mathtt{newfst}, \mathtt{new\ Pair(newfst, this.snd)})}\ \text{(MBODY1)}}{\begin{array}{l}\mathtt{new\ Pair(new\ A(), new\ B()).setfst(new\ B())} \\ \to [\mathtt{newfst} \mapsto \mathtt{new\ B(), this} \mapsto \mathtt{new\ Pair(new\ A(), new\ B())}]\mathtt{new\ Pair(newfst,\ this.snd)}\end{array}}\ \text{(E-INVKNEW)}$$

であり、1 ステップ目の結果の置換を行うと

$$\begin{array}{l}\mathtt{new\ Pair(new\ A(),\ new\ B()).setfst(new\ B())} \\ \to [\mathtt{newfst} \mapsto \mathtt{new\ B(), this} \mapsto \mathtt{new\ Pair(new\ A(), new\ B())}]\mathtt{new\ Pair(newfst,\ this.snd)} \\ = \mathtt{new\ Pair(new\ B(), newPair(new\ A(), new\ B()).snd)}\end{array}$$

である。2 ステップ目は、

$$\frac{\dfrac{CT(\mathtt{Pair}) = \mathtt{class\ Pair\ extends\ Object}\{\mathtt{Object\ fst;\ Object\ snd;}\ldots\} \quad \dfrac{}{\mathit{fields}(\mathtt{Object}) = \bullet}\ \text{(FIELDS1)}}{\mathit{fields}(\mathtt{Pair}) = \mathtt{Object\ fst;\ Object\ snd;}}\ \text{(FIELDS2)}}{\mathtt{new\ Pair(new\ A(), new\ B()).snd} \to \mathtt{new\ B()}}\ \text{(E-PROJNEW)}$$
$$\frac{}{\mathtt{new\ Pair(new\ B(), new\ Pair(new\ A(), new\ B()).snd)} \to \mathtt{new\ Pair(new\ B(), new\ B())}}\ \text{(E-NEW-ARG)}$$

である。

例 2. 上記の例は型が付き、型判定の導出木は

$$\frac{(*1) \quad (*2) \quad (*3) \quad (*4)}{\vdash \mathtt{new\ Pair(new\ A(), new\ B()).setfst(new\ B())} : \mathtt{Pair}}\ \text{(T-INVK)}$$

である。(*1) の部分は

$$\frac{\text{(*1-1)} \quad \text{(*1-2)} \quad \text{(*1-3)} \quad \text{(*1-4)} \quad \text{(*1-5)}}{\vdash \texttt{new Pair(new A(), new B())} : \texttt{Pair}}\ \text{(T-NEW)}$$

である。(*1-1) の部分は

$$\frac{CT(\texttt{Pair}) \;=\; \begin{array}{l}\texttt{class Pair extends Object}\\ \{\texttt{Object fst; Object snd;} \ldots\}\end{array} \qquad \frac{}{\mathit{fields}(\texttt{Object}) = \bullet}\ \text{(FIELDS1)}}{\mathit{fields}(\texttt{Pair}) = \texttt{Object fst; Object snd;}}\ \text{(FIELDS2)}$$

であり、(*1-2) の部分は

$$\frac{\dfrac{CT(\texttt{A}) = \texttt{class A extends Object}\{\ldots\} \quad \dfrac{}{\mathit{fields}(\texttt{Object}) = \bullet}\ \text{(FIELDS1)}}{\mathit{fields}(\texttt{A}) = \bullet}\ \text{(FIELDS2)}}{\vdash \texttt{new A()} : \texttt{A}}\ \text{(T-NEW)}$$

であり、(*1-3) の部分は

$$\frac{\dfrac{CT(\texttt{B}) = \texttt{class B extends Object}\{\ldots\} \quad \dfrac{}{\mathit{fields}(\texttt{Object}) = \bullet}\ \text{(FIELDS1)}}{\mathit{fields}(\texttt{B}) = \bullet}\ \text{(FIELDS2)}}{\vdash \texttt{new B()} : \texttt{B}}\ \text{(T-NEW)}$$

であり、(*1-4) の部分は

$$\frac{CT(\texttt{A}) = \texttt{class A extends Object}\{\ldots\}}{\texttt{A} <: \texttt{Object}}\ \text{(S-CLASS)}$$

であり、(*1-5) の部分は

$$\frac{CT(\texttt{B}) = \texttt{class B extends Object}\{\ldots\}}{\texttt{B} <: \texttt{Object}}\ \text{(S-CLASS)}$$

である。(*2) の部分は

$$\frac{\begin{array}{c}CT(\texttt{Pair}) = \texttt{class Pair extends Object}\{\ldots \overline{\texttt{M}}\}\\ \texttt{Pair setfst(Object newfst)}\{\texttt{return} \ldots\} \in \overline{\texttt{M}}\end{array}}{\mathit{mtype}(\texttt{setfst}, \texttt{Pair}) = \texttt{Object} \to \texttt{Pair}}\ \text{(T-MTYPE1)}$$

であり、(*3) の部分は (*1-3) と同じであり、(*4) の部分は (*1-5) と同じである。

練習問題 1. 以下の term は空の型環境で型を持つが、その型判定の導出木を示せ。

```
((Pair) (new Pair(new Pair(new A(), new B()),
                  new A()).fst)).snd
```

練習問題 2. 以下の term を正規形まで評価せよ。その際、各 1 ステップ評価の導出木も示せ。

```
((Pair) (new Pair(new Pair(new A(), new B()),
                  new A()).fst)).snd
```

3 FJの性質

FJが満たす性質 (Preservation と Progress) を示す。

まず、Preservationについては、FJの場合、termは先祖のクラスの型を持てるので、一般には、1ステップ評価後のtermの型は評価前のtermの型の部分型である。

定理 1 (Preservation)**.** $\Gamma \vdash \mathtt{t} : \mathtt{C}$ かつ $\mathtt{t} \to \mathtt{t}'$ ならば何らかの型 $\mathtt{C}'$ に対し、$\mathtt{C}' <: \mathtt{C}$ かつ $\Gamma \vdash \mathtt{t}' : \mathtt{C}'$ である。

証明. 省略する。 □

次にProgressを示すが、まず、以下の補題を示す。

補題 1. t が型の付くtermであるとする。

1. $\mathtt{t} = \mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{t}}).\mathtt{f}$ のとき、$\mathit{fields}(\mathtt{C}_0) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}$ かつ $\mathtt{f} \in \overline{\mathtt{f}}$ である。
2. $\mathtt{t} = \mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{t}}).\mathtt{m}(\overline{\mathtt{s}})$ のとき、$\mathit{mbody}(\mathtt{m}, \mathtt{C}_0) = (\overline{\mathtt{x}}, \mathtt{t}_0)$ かつ $|\overline{\mathtt{x}}| = |\overline{\mathtt{s}}|$ である（$|\overline{\mathtt{x}}|$ はメソッド m の仮引数の個数, $|\overline{\mathtt{s}}|$ は実引数の個数)。

証明. 省略する。 □

FJの場合、downcastが実行できない場合に評価がstuckするので、その状況を表すために評価文脈 (evaluation context) を定義する。

定義 1. FJの評価文脈を以下のように定義する。

$$E ::= [\,] \mid E.\mathtt{f} \mid E.\mathtt{m}(\overline{\mathtt{t}}) \mid \mathtt{v}.\mathtt{m}(\overline{\mathtt{v}}, E, \overline{\mathtt{t}}) \mid \mathtt{new}\ \mathtt{C}(\overline{\mathtt{v}}, E, \overline{\mathtt{t}}) \mid (\overline{\mathtt{C}})E$$

各評価文脈は一つの穴 (hole, [ ]) を持つtermである。評価文脈 E 中の穴をterm tで置き換えて得られるtermを $E[\mathtt{t}]$ と書く。評価文脈は、「次に簡約される部分term」を表すために用いる。$\mathtt{t} \to \mathtt{t}'$ のとき、$\mathtt{t} = E[\mathtt{r}]$、$\mathtt{t}' = E[\mathtt{r}']$ が成り立つような評価文脈E, term t, term r が存在し、E-PROJNEW, E-INVKNEW, E-CASTNEWのいずれかの評価規則により $\mathtt{r} \to \mathtt{r}'$ が成り立つ。

定理 2 (Progress)**.** t が閉じたterm(自由変数の無いterm)で型が付く正規形のtermであるとする。このとき、以下のいずれかが成り立つ。

(1) t は値である。

(2) t は $\mathtt{t} = E[(\mathtt{C})(\mathtt{new}\ \mathtt{D}(\overline{\mathtt{v}})]$ (ただし $\mathtt{D} \not<: \mathtt{C}$) の形で表される。

証明. 省略する。 □

このProgress定理はもう少し明確に述べることもできる。term t中のcastがupcastだけの場合（つまりtに関する型判定の導出木がT-DCAST, T-SCASTを含まない場合）は評価はstuckしない。一般に、あるtermを低い型へcastする場合 (つまり型判定の導出木中でT-DCAST規則が使われる場合)、そのcastは実行時に失敗する（FJだと評価がstuckし、Javaだと例外を投げる）可能性がある。