情報流解析における機密度ワイルドカードの検討

Towards Secrecy Wildcards in Information Flow Analysis

桑原 寛明 *

Summary. This paper proposes secrecy wildcards for improving the flexibility of classes with parameterized secrecy in information flow analysis. It is hard to implement flexible APIs with secrecy-parameterized classes because the secrecy-parameterized classes are invariant, but wildcards for secrecy do not exist. In this paper, we formalize bounded secrecy wildcards used in the type of method parameters and show the candidates of Java annotations for secrecy wildcards.

1 はじめに

プログラムが処理する機密情報がプログラムの外部に流出しないことを静的に検査する手法として，型検査に基づく情報流解析が提案されている [1] [2] [3] [4]．型検査に基づく情報流解析では，データの機密度を型として利用し，型付け可能なプログラムが非干渉性を満たすように型システムを構築する．非干渉性は，機密度の低いデータが機密度の高いデータに直接および間接的に依存しないことを表し，機密データ自体に加え機密データを推測できる情報も流出しないという意味でよい性質である．

型検査に基づく情報流解析では，プログラム中の変数や関数の返り値の型として機密度を指定する必要があり，通常は具体的な機密度を指定する．一方，汎用的なコレクションフレームワークのような扱うデータの機密度を作成時には決められないプログラムも存在し，このようなプログラムを記述するために機密度のパラメータ化と機密度パラメータに対応した情報流解析のための型システムが提案されている [5]．この手法では，Java 言語におけるジェネリックなクラスのように，具体的な機密度の代わりに機密度パラメータを用いてクラスを定義し，当該クラスの利用時に具体的な機密度を機密度パラメータに割り当てる．

さらに，機密度や機密度パラメータなど情報流解析に必要な要素をプログラム中に記述するための手段として Java アノテーションが提案されており [6] [7]，Java プログラム中に注釈として機密度などを記述できる．しかし，従来の研究で提案されている機密度パラメータと Java アノテーションでは，機密度パラメータを持つクラスが不変 (invariant) であるため，柔軟な API の実現が難しい場合がある．Java のジェネリクスでもパラメータ化された型は不変であるため同様の問題があるが，境界ワイルドカードの導入により解決されている．

本稿では，機密度がパラメータ化されたクラスの柔軟性向上のために，既存手法を拡張して機密度のワイルドカードを導入する．特に，機密度がパラメータ化されたクラスがメソッドの引数の型として利用される場合に着目し，上限あるいは下限が存在する境界付きのワイルドカードを提案する．

2 ワイルドカード

Java では，パラメータ化されたクラスは不変である．2 つの異なる型 `T1` と `T2` に対して `T1` と `T2` がサブタイプ関係にあるか否かに関わらず `Collection<T1>`は `Collection<T2>`のサブタイプでもスーパータイプでもない．ここで，`List<T>`の 2 つメソッド，指定された要素を追加する `add`，指定された `Collection` の要素をすべて追加する `addAll` を考える．`U` を `T` のサブタイプとする．`add` の仮引数の型は `T` で

*Hiroaki Kuwabara, 南山大学情報センター

$$
\begin{array}{rl}
T & ::= \mathrm{Bool} \mid C \\
\rho & ::= (\eta, \mathcal{X}) \qquad\qquad \rho_w ::= \rho \mid ?^{\preceq\rho} \mid {}^{\rho\preceq}? \\
\tau & ::= (T\langle\overline{\rho}\rangle, \rho) \qquad\qquad \tau_w ::= (T\langle\overline{\rho_w}\rangle, \rho) \\
CL & ::= \mathrm{class}\, C\langle\overline{X}\rangle\rho\, \mathrm{extends}\, C\langle\overline{\rho}\rangle\, \{\overline{\tau\ f;}\ \overline{M}\} \\
M & ::= \tau\ m(\overline{\tau_w\ x})\ \rho\ B \\
B & ::= \{\overline{\tau\ x;}\ \overline{S;}\} \\
S & ::= x\ =\ e \mid e.f = e \mid x\ =\ e.m(\overline{e}) \mid x\ =\ \mathrm{new}\, C\langle\overline{\rho}\rangle \mid \mathrm{if}\ (e)\ B\ \mathrm{else}\ B \\
e & ::= x \mid \mathrm{true} \mid \mathrm{false} \mid e.f \mid e\ ==\ e
\end{array}
$$

図 1　言語の構文

あり，T のサブタイプである U の値も実引数に指定できる．一方，単純に addAll の仮引数の型を Collection<T>とすると，パラメータ化されたクラスは不変であるため Collection<U>の値を実引数に指定できず，add のような柔軟性に欠ける．この問題は，境界ワイルドカードを用いて仮引数の型を Collection<? extends T>とすることで解決できる．

Java と同様に，情報流解析のための機密度がパラメータ化されたクラス [5] は不変である．すなわち，ある機密度束を構成する 2 つの異なる機密度 L と H に対して List⟨L⟩ は List⟨H⟩ のサブタイプでもスーパータイプでもない [1]．例えば，L を H 以下の機密度（すなわち L は H のサブタイプ），l と h をそれぞれ機密度 L と H の変数，ls と hs をそれぞれ List⟨L⟩ と List⟨H⟩ の変数とすると，h = l は正当であるが，hs = ls は許されない．もし機密度がパラメータ化されたクラスが共変 (covariant) である，すなわち List⟨L⟩ が List⟨H⟩ のサブタイプであるとして hs = ls を許すと，

```
hs = ls; hs.add(h); l = ls.get(...);
```

のようにして List⟨L⟩ の変数を通して機密度 H の値にアクセスできるからである．機密度がパラメータ化されたクラスが不変であるため，X を機密度パラメータとして List⟨X⟩ のメソッド addAll の仮引数の型は Collection⟨X⟩ ではなく X 以下の機密度の要素を持つ Collection としたい．Collection⟨? upperBound X⟩ のように仮引数の型を記述できれば API の柔軟性を維持できるが，機密度に関して Java の境界ワイルドカードに相当する仕組みは存在しない．

3　機密度ワイルドカードの形式化

機密度がパラメータ化されたクラスを持つオブジェクト指向言語と情報流解析のための型システム [5] [7] に機密度ワイルドカードを導入して拡張する．以下では，機密度定数の束 $(\mathcal{H}, \sqsubseteq)$ と機密度パラメータ全体の集合 $\mathcal{Y}$ を仮定する．束の最小元を $\bot_{\mathcal{H}}$ と表す．機密度 ρ を機密度定数 $\eta \in \mathcal{H}$ と機密度パラメータの集合 $\mathcal{X} \subseteq \mathcal{Y}$ の組 $(\eta, \mathcal{X})$ として定義する．直観的には，$(\eta, \mathcal{X})$ は η とすべての $X \in \mathcal{X}$ の結びである．機密度 $(\eta, \mathcal{X})$ について，$\mathcal{X}$ が空の時は単に η と書き，η が $\bot_{\mathcal{H}}$ かつ $\mathcal{X}$ が $\{X\}$ の時は単に X と書く．機密度の集合 $\mathcal{P} = \mathcal{H} \times 2^{\mathcal{Y}}$ 上に機密度の大小関係 $\preceq$ を $(\eta_1, \mathcal{X}_1) \preceq (\eta_2, \mathcal{X}_2)$ **iff** $\eta_1 \sqsubseteq \eta_2 \wedge \mathcal{X}_1 \subseteq \mathcal{X}_2$ と定義する．この時，$(\mathcal{P}, \preceq)$ は束である．

言語の構文を図 1 に示す．型情報 τ はデータ型 $T\langle\overline{\rho}\rangle$ と機密度 ρ の組である．$T\langle\overline{\rho}\rangle$ は T の機密度パラメータ $\overline{X}$ に機密度 $\overline{\rho}$ を割り当てた型である．T が機密度パラメータを持たず $\overline{X}$ の長さが 0 の場合は単に T と書く．τ_w と ρ_w はそれぞれ機密度ワイルドカードを含む型情報と機密度であり，$?^{\preceq\rho}$ は上限が ρ，${}^{\rho\preceq}?$ は下限が ρ の機密度ワイルドカードである．クラス定義 CL は，クラス名 C と，いずれも 0 個以上の機密度パラメータの宣言 $\overline{X}$，フィールドの宣言 $\overline{\tau\ f;}$，メソッド定義 $\overline{M}$ からなる．機密度ワイルドカードは境界付きのみであり，出現位置はメソッドの仮引数の型のみに限られる．

[1] 簡単のために要素のデータ型は省略する．

ワイルドカードを含む機密度の大小関係 $\preceq_?$ と $\preceq_:$ を以下のように定義する．

- $?^{\preceq\rho} \preceq_? \rho^{\preceq}?$
- $\rho \preceq \rho'$ ならば $\rho \preceq_? ?^{\preceq\rho'}$，$?^{\preceq\rho} \preceq_? ?^{\preceq\rho'}$，$\rho' \preceq_? \rho^{\preceq}?$，${\rho'}^{\preceq}? \preceq_? \rho^{\preceq}?$
- $\rho \preceq \rho'$ ならば $\rho \preceq_: \rho'$，$\rho \preceq_: {\rho'}^{\preceq}?$，$?^{\preceq\rho} \preceq_: \rho'$，$?^{\preceq\rho} \preceq_: {\rho'}^{\preceq}?$

直観的には，$\rho_w \preceq_? \rho'_w$ であれば $C\langle\rho'_w\rangle$ の変数に $C\langle\rho_w\rangle$ の値が代入可能，$\rho_w \preceq_: \rho'_w$ であれば ρ'_w の変数に ρ_w の値が代入可能となるように定めている．

データ型 $T\langle\overline{\rho_w}\rangle$ のサブタイプ関係 $\trianglelefteq$ を以下を満たすように定義する．

- $\forall i\,.\,(\rho_{w_i} = \rho'_{w_i} \vee \rho_{w_i} \preceq_? \rho'_{w_i})$ ならば $T\langle\overline{\rho_w}\rangle \trianglelefteq T\langle\overline{\rho'_w}\rangle$
- $C\langle\overline{\rho_w}\rangle \trianglelefteq D\langle\overline{\rho'_w}\rangle$ **iff** C の宣言を class $C\langle\overline{X}\rangle\ \rho$ extends $E\langle\overline{\rho}\rangle\ \{\ldots\}$ として $E\langle[\overline{\rho_w}/\overline{X}]\rho\rangle \trianglelefteq D\langle\overline{\rho'_w}\rangle$

情報流解析のための型付け規則を図 2 に示す．$level(T\langle\overline{\rho_w}\rangle)$ は $T\langle\overline{\rho_w}\rangle$ の機密度，$ftype(f, T\langle\overline{\rho_w}\rangle)$ は $T\langle\overline{\rho_w}\rangle$ が持つフィールド f の型，$mtype(m, T\langle\overline{\rho_w}\rangle)$ は $T\langle\overline{\rho_w}\rangle$ が持つメソッド m の型を表す．文とブロックの型判定式 $\Delta \vdash S : (\rho_s, \rho_h)$ は，S で代入される変数の機密度が ρ_s 以上，フィールドへの代入などで変更されるヒープ上のデータの機密度が ρ_h 以上であることを表す．式の型判定式 $\Delta \vdash e : (T\langle\overline{\rho_w}\rangle, \rho)$ は，e のデータ型が $T\langle\overline{\rho_w}\rangle$ かそのサブタイプ，機密度が ρ 以下であることを表す．図 2 の型付け規則は [5] の拡張であり，機密度に関する条件にワイルドカードに対応した $\preceq_:$ を利用する点と，メソッド呼び出しに対する CALL 規則において仮引数の機密度が上限付き機密度ワイルドカードではないことを求める条件が追加されている点が異なる．

4 例

図 3 に示すプログラム例に対して型付け可能な否か確認する．ここで, L と H は機密度定数であり $\mathtt{L} \sqsubseteq \mathtt{H}$ とする．定義より $\mathtt{L} \preceq \mathtt{H}$ である．$\Delta = \mathtt{u} : (\mathtt{Unit}, \mathtt{H}), \mathtt{h} : (\mathtt{Bool}, \mathtt{H}), \mathtt{ls} : (\mathtt{List}\langle\mathtt{L}\rangle, \mathtt{L}), \mathtt{hs} : (\mathtt{List}\langle\mathtt{H}\rangle, \mathtt{H}), \mathtt{whs} : (\mathtt{List}\langle?^{\preceq\mathtt{H}}\rangle, \mathtt{H})$ とし，$\mathtt{List}\langle\mathtt{X}\rangle$ は

```
class List⟨X⟩ X extends Collection⟨X⟩ {
  (Unit, L) add((Bool, X) e) X {...}
  (Unit, L) addAll((Collection⟨?⪯X⟩, X) c) X {...}
  ...
}
```

のように宣言されているとする．

$\mathtt{List}\langle\mathtt{X}\rangle$ は不変であるため (1) の代入は不正である．型付け不能であることを確認する．ASSIGN1 規則の適用において，$\Delta(\mathtt{hs})$ と $\Delta(\mathtt{ls})$ から $\mathtt{List}\langle\mathtt{L}\rangle \trianglelefteq \mathtt{List}\langle\mathtt{H}\rangle$ でなければならない．しかし，$\mathtt{L} = \mathtt{H}$ でも $\mathtt{L} \preceq_? \mathtt{H}$ でもないため $\trianglelefteq$ の定義から $\mathtt{List}\langle\mathtt{L}\rangle \not\trianglelefteq \mathtt{List}\langle\mathtt{H}\rangle$ であり，(1) の代入は型付け不能である．

```
(Unit, H) u;
(Bool, H) h;
(List⟨L⟩, L) ls;
(List⟨H⟩, H) hs;
(List⟨?⪯H⟩, H) whs;

// (1) NG
hs = ls;

// (2) OK
whs = ls;

// (3) NG
u = whs.add(h);

// (4) OK
u = hs.addAll(ls);
```

図 3　プログラム例

一方，(2) の代入は正当であり，型付け可能であることを確認する．(1) と同様に，ASSIGN1 規則の適用において，$\Delta(\mathtt{whs})$ と $\Delta(\mathtt{ls})$ から $\mathtt{List}\langle\mathtt{L}\rangle \trianglelefteq \mathtt{List}\langle?^{\preceq\mathtt{H}}\rangle$ でなければならない．この時，$\preceq_?$ の定義から $\mathtt{L} \preceq_? ?^{\preceq\mathtt{H}}$ ゆえ $\mathtt{List}\langle\mathtt{L}\rangle \trianglelefteq \mathtt{List}\langle?^{\preceq\mathtt{H}}\rangle$ である．よって，(2) の代入は型付け可能である．

(3) のメソッド呼び出しにおいて，whs のデータ型は $\mathtt{List}\langle?^{\preceq\mathtt{H}}\rangle$ ゆえ，add の型は $(\mathtt{Bool}, ?^{\preceq\mathtt{H}}) \xrightarrow{?^{\preceq\mathtt{H}}} (\mathtt{Unit}, \mathtt{L})$ である．仮引数の機密度が $?^{\preceq\mathtt{H}}$ であるため，機密度が H の変数 h を実引数として渡せるように見える．しかし，この時点で whs が指すリストの型は $\mathtt{List}\langle\mathtt{L}\rangle$ であり，このリストに対する機密度が H の値の追加は不正な情報流となる．これを防ぐためには，レシーバ変数のデータ型が $\mathtt{List}\langle?^{\preceq\mathtt{H}}\rangle$ のように

$$\frac{\begin{array}{l} \mathit{level}(D\langle\overline{\rho}\rangle) \preceq: \rho \quad C\langle\overline{X}\rangle\rho \text{ extends } D\langle\overline{\rho}\rangle \vdash M \text{ for each } M \in \overline{M} \\ \mathit{level}(D\langle\overline{\rho}\rangle) \neq \rho \Rightarrow \\ \qquad \text{every } m \text{ with } \mathit{mtype}(m, D\langle\overline{\rho}\rangle) \text{ defined is overridden in } C \end{array}}{\vdash C\langle\overline{X}\rangle\rho \text{ extends } D\langle\overline{\rho}\rangle\{\overline{\tau\ f};\ \overline{M}\}} \text{[CDEC]}$$

$$\frac{\begin{array}{l} \overline{x : \tau_x},\ \text{this} : (C\langle\overline{X}\rangle, \rho),\ \mathit{result} : \tau_r \vdash B : (\rho_s, \rho'_h) \\ \mathit{mtype}(m, D\langle\overline{\rho}\rangle) \text{ is defined } \Rightarrow \mathit{mtype}(m, D\langle\overline{\rho}\rangle) = \overline{x : \tau_x} \xrightarrow{\rho_h} \tau_r \\ \rho_h \preceq: \rho'_h \end{array}}{C\langle\overline{X}\rangle\rho \text{ extends } D\langle\overline{\rho}\rangle \vdash \tau_r\ m(\overline{\tau_x\ x})\ \rho_h\ B} \text{[MDEC]}$$

$$\frac{\Delta,\ \overline{x : (T_x\langle\overline{\rho_x}\rangle, \rho_x)} \vdash S_i : (\rho^i_s, \rho^i_h) \quad \rho_s \preceq: \rho^i_s \quad \rho_h \preceq: \rho^i_h \quad i \in \{1, \ldots, n\}}{\Delta \vdash \{\overline{(T_x\langle\overline{\rho_x}\rangle, \rho_x)\ x};\ S_1; \ldots S_n;\} : (\rho_s, \rho_h)} \text{[BLOCK]}$$

$$\frac{\begin{array}{l} \Delta \vdash x : (T_x\langle\overline{\rho_x}\rangle, \rho_x) \quad \Delta \vdash e : (T_e\langle\overline{\rho_e}\rangle, \rho_e) \\ T_e\langle\overline{\rho_e}\rangle \trianglelefteq T_x\langle\overline{\rho_x}\rangle \quad \rho_e \preceq: \rho_x \quad \rho_s \preceq: \rho_x \end{array}}{\Delta \vdash x = e : (\rho_s, \rho_h)} \text{[ASSIGN1]}$$

$$\frac{\begin{array}{l} \Delta \vdash e_1 : (T_1\langle\overline{\rho_1}\rangle, \rho_1) \quad \Delta \vdash e_2 : (T_2\langle\overline{\rho_2}\rangle, \rho_2) \\ (T_f\langle\overline{\rho_f}\rangle, \rho_f) = \mathit{ftype}(f, T_1\langle\overline{\rho_1}\rangle) \\ T_2\langle\overline{\rho_2}\rangle \trianglelefteq T_f\langle\overline{\rho_f}\rangle \quad \rho_1 \preceq: \rho_f \quad \rho_2 \preceq: \rho_f \quad \rho_h \preceq: \rho_f \end{array}}{\Delta \vdash e_1.f = e_2 : (\rho_s, \rho_h)} \text{[ASSIGN2]}$$

$$\frac{\begin{array}{l} \Delta \vdash x : (T_x\langle\overline{\rho_x}\rangle, \rho_x) \quad \Delta \vdash e : (T_e\langle\overline{\rho_e}\rangle, \rho_e) \\ (T'_1\langle\overline{\rho'_1}\rangle, \rho'_1), \ldots, (T'_n\langle\overline{\rho'_n}\rangle, \rho'_n) \xrightarrow{\rho'_h} (T_r\langle\overline{\rho_r}\rangle, \rho_r) = \mathit{mtype}(m, T_e\langle\overline{\rho_e}\rangle) \\ \Delta \vdash e_i : (T_i\langle\overline{\rho_i}\rangle, \rho_i) \quad T_i\langle\overline{\rho_i}\rangle \trianglelefteq T'_i\langle\overline{\rho'_i}\rangle \quad \rho_i \preceq: \rho'_i \quad \forall \rho\,.\, \rho'_i \neq ?^{\preceq \rho} \quad i \in \{1, \ldots, n\} \\ T_r\langle\overline{\rho_r}\rangle \trianglelefteq T_x\langle\overline{\rho_x}\rangle \quad \rho_e \preceq: \rho_x \quad \rho_r \preceq: \rho_x \quad \rho_s \preceq: \rho_x \quad \rho_e \preceq: \rho'_h \quad \rho_h \preceq: \rho'_h \end{array}}{\Delta \vdash x = e.m(e_1, \ldots, e_n) : (\rho_s, \rho_h)} \text{[CALL]}$$

$$\frac{\begin{array}{l} \Delta \vdash x : (T_x\langle\overline{\rho_x}\rangle, \rho_x) \quad C\langle\overline{\rho_C}\rangle \trianglelefteq T_x\langle\overline{\rho_x}\rangle \\ \mathit{level}(C\langle\overline{\rho_C}\rangle) \preceq: \rho_x \quad \rho_s \preceq: \rho_x \quad \rho_h \preceq: \mathit{level}(C\langle\overline{\rho_C}\rangle) \end{array}}{\Delta \vdash x = \text{new } C\langle\overline{\rho_C}\rangle : (\rho_s, \rho_h)} \text{[NEW]}$$

$$\frac{\begin{array}{l} \Delta \vdash e : (\text{Bool}, \rho_e) \quad \Delta \vdash B_t : (\rho^t_s, \rho^t_h) \quad \Delta \vdash B_f : (\rho^f_s, \rho^f_h) \\ \rho_e \preceq: \rho_s \quad \rho_s \preceq: \rho^t_s \quad \rho_s \preceq: \rho^f_s \quad \rho_e \preceq: \rho_h \quad \rho_h \preceq: \rho^t_h \quad \rho_h \preceq: \rho^f_h \end{array}}{\Delta \vdash \text{if } (e)\ B_t \text{ else } B_f : (\rho_s, \rho_h)} \text{[IF]}$$

$$\frac{}{\Delta \vdash x : \Delta(x)} \text{[VAR]} \qquad \frac{\begin{array}{l} \Delta \vdash e : (T_e\langle\overline{\rho_e}\rangle, \rho_e) \\ (T_f\langle\overline{\rho_f}\rangle, \rho_f) = \mathit{ftype}(f, T_e\langle\overline{\rho_e}\rangle) \\ \rho_e \preceq: \rho \quad \rho_f \preceq: \rho \end{array}}{\Delta \vdash e.f : (T_f\langle\overline{\rho_f}\rangle, \rho)} \text{[FIELD]}$$

$$\frac{c \in \{\text{true}, \text{false}\}}{\Delta \vdash c : (\text{Bool}, \bot_{\mathcal{H}})} \text{[BOOL]} \qquad \frac{\Delta \vdash e_i : (T\langle\overline{\rho}\rangle, \rho_i) \quad \rho_i \preceq: \rho \quad i \in \{1, 2\}}{\Delta \vdash e_1\ ==\ e_2 : (\text{Bool}, \rho)} \text{[EQ]}$$

図 2　型付け規則

```
@Target({ElementType.TYPE,
  ElementType.TYPE_USE})
public @interface Secrecy {
  SecLattice value() default BOT;
  String[] params() default {};
}
```

```
@Target(ElementType.TYPE_USE)
public @interface Assign {
  String param();
  Secrecy arg();
}
```

図 4　既存の Java アノテーション（抜粋）

```
@Target({ElementType.TYPE,
  ElementType.TYPE_USE})
public @interface Secrecy {
  SecLattice value() default BOT;
  String[] params() default {};
  Secrecy upperBound; // added
  Secrecy lowerBound; // added
}
```

図 5　アノテーションの拡張その 1

```
public @interface Wildcard {
  Secrecy upperBound;
  Secrecy lowerBound;
}

@Target(ElementType.TYPE_USE)
public @interface Assign {
  String param();
  Secrecy arg();
  Wildcard wildcard(); // added
}
```

図 6　アノテーションの拡張その 2

機密度パラメータに上限付き機密度ワイルドカードが指定された型である場合，宣言においてその機密度パラメータが仮引数の機密度に出現するメソッドの呼び出しを禁止すれば十分である．CALL 規則に追加された条件がこの禁止を実現する．

(3) のメソッド呼び出しが型付け不能であることを確認する．CALL 規則の適用において，add の仮引数の型は $(\texttt{Bool}, ?^{\preceq \texttt{H}})$ であり，実引数 h の型は $\Delta(\texttt{h})$ より $(\texttt{Bool}, \texttt{H})$ である．ここで，CALL 規則には仮引数の機密度が $?^{\preceq \rho}$，すなわち上限付き機密度ワイルドカードではないという条件があるため，型付けできない．

(4) のメソッド呼び出しは型付け可能であることを確認する．hs のデータ型が $\texttt{List}\langle\texttt{H}\rangle$ であるので，addAll の仮引数の型は $(\texttt{Collection}\langle ?^{\preceq \texttt{H}}\rangle, \texttt{H})$ である．実引数 ls のデータ型は $\texttt{List}\langle\texttt{L}\rangle$ ゆえ $\texttt{List}\langle\texttt{L}\rangle \trianglelefteq \texttt{Collection}\langle ?^{\preceq \texttt{H}}\rangle$ であればよい．ここで，$\preceq_?$ の定義より $\texttt{L} \preceq_? ?^{\preceq \texttt{H}}$ であり，$\trianglelefteq$ の定義より $\texttt{Collection}\langle\texttt{L}\rangle \trianglelefteq \texttt{Collection}\langle ?^{\preceq \texttt{H}}\rangle$ である．さらに，$\texttt{List}\langle\texttt{X}\rangle$ が $\texttt{Collection}\langle\texttt{X}\rangle$ を継承していることと $\trianglelefteq$ の定義より $\texttt{List}\langle\texttt{L}\rangle \trianglelefteq \texttt{Collection}\langle ?^{\preceq \texttt{H}}\rangle$ である．

5　Java アノテーション

機密度ワイルドカードを記述できるように情報流解析のための Java アノテーション [5] [7] を拡張する．機密度ワイルドカードは機密度パラメータへの割り当てにおいて出現するが，関係するアノテーションとして [7] では図 4 に示す 2 つのアノテーションが定義されている．@Secrecy は機密度を表し，@Assign は機密度パラメータに対する機密度の割り当てを表す．

機密度ワイルドカードへの対応として 2 通りの拡張方法が考えられる．一つは，図 5 のように機密度ワイルドカードも@Secrecy を用いて表す方法であり，機密度ワイルドカードの上限と下限を表すための要素を@Secrecy に追加する．この場合，変更は@Secrecy 内に閉じているが，@Secrecy が通常の機密度と機密度ワイルドカードのいずれを表すか値が設定される要素から判断する規則を定める必要がある．

もう一つは，図 6 のように機密度ワイルドカードを表すための新しいアノテーション（図 6 では@Wildcard）を定義する方法である．新しいアノテーションには機密度ワイルドカードの上限と下限を表す要素を持たせる．この方法では，機密度パラメータに対する割り当てを表す@Assign に機密度ワイルドカードを指定するための

要素を追加し，通常の機密度と機密度ワイルドカードが共に指定された場合の解決手段を決める必要がある．一方，通常の機密度と機密度ワイルドカードが異なるアノテーションで表されるため記述はわかりやすくなる．

本稿で提案した機密度ワイルドカードに限れば変更点の少ない 1 つ目の方法がよいと思われる．一方，将来的に上下限のない機密度ワイルドカードを導入するのであれば 2 つ目の方法を採る必要がある．情報流解析における上下限のない機密度ワイルドカードの必要性の検討は今後の課題であり，必要性に応じてアノテーションの定義方法を選択する．

6 おわりに

本稿では，オブジェクト指向プログラムを対象とする情報流解析において，機密度がパラメータ化されたクラスの柔軟性向上のために機密度ワイルドカードを導入した．特に，機密度がパラメータ化されたクラスがメソッドの仮引数の型に出現する場合に着目して，境界付きの機密度ワイルドカードを書くための構文を定義した上で情報流解析のための型システムを提案した．形式化にあたり，Java のワイルドカードを参考とし，機密度ワイルドカードを含む機密度の大小関係とデータ型のサブタイプ関係を定義した．

関連研究として，Java におけるワイルドカードの導入と形式化の研究が挙げられる [8] [9] [10]．これらの研究は存在型 (existential types) を用いた形式化を提案している．本稿では，対象をメソッドの仮引数に出現する境界付きのワイルドカードに制限し，機密度の大小関係とデータ型のサブタイプ関係の拡張により形式化した．

今後の課題として，メソッドの引数の使用法解析の追加による仮引数の機密度に対する条件の緩和，フィールドやローカル変数などメソッドの仮引数以外の型に出現する機密度ワイルドカードへの対応，非干渉性に対する健全性の証明，機密度ワイルドカードのための Java アノテーションの実装が挙げられる．

謝辞 本研究の一部は JSPS 科研費 JP17K12666，JP18K11241 および 2020 年度南山大学パッヘ研究奨励金 I-A-2 の助成による．

参考文献

[1] Anindya Banerjee and David A. Naumann. Secure Information Flow and Pointer Confinement in a Java-like Language. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pp. 253–267, 2002.

[2] 黒川翔, 桑原寛明, 山本晋一郎, 坂部俊樹, 酒井正彦, 草刈圭一朗, 西田直樹. 例外処理付きオブジェクト指向プログラムにおける情報流の安全性解析のための型システム. 電子情報通信学会論文誌 D, Vol. J91-D, No. 3, pp. 757–770, 2008.

[3] Andrei Sabelfeld and Andrew C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communications*, Vol. 21, No. 1, pp. 5–19, 2003.

[4] Dennis Volpano, Geoffrey Smith, and Cynthia Irvine. A Sound Type System for Secure Flow Analysis. *Journal of Computer Security*, Vol. 4, No. 2, pp. 167–187, 1996.

[5] 吉田真也, 桑原寛明, 國枝義敏. オブジェクト指向言語の情報流解析における機密度のパラメータ化. コンピュータ ソフトウェア, Vol. 36, No. 1, pp. 48–65, 2019.

[6] 吉田真也, 桑原寛明, 國枝義敏. 情報流解析のための Java アノテーション. コンピュータ ソフトウェア, Vol. 34, No. 4, pp. 47–53, 2017.

[7] 桑原寛明, 國枝義敏. 機密度パラメータ付き情報流解析のための型検査アルゴリズムと Java アノテーション. ソフトウェア工学の基礎 XXVI(FOSE2019), pp. 109–114, 2019.

[8] Mads Torgersen, Christian Plesner Hansen, Erik Ernst, Peter von der Ahé, Gilad Bracha, and Neal Gafter. Adding Wildcards to the Java Programming Language. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pp. 1289–1296, 2004.

[9] Mads Torgersen, Erik Ernst, and Christian Plesner Hansen. Wild FJ. In *FOOL 2005*, 2005.

[10] Nicholas Cameron, Sophia Drossopoulou, and Erik Ernst. A Model for Java with Wildcards. In *ECOOP 2008*, pp. 2–26, 2008.