任意の機密度束を用いた情報流解析における非機密化プリミティブの配置

Declassifiers Placement for General Secrecy Lattices in Information Flow Analysis

桑原 寛明*　國枝 義敏†

Summary. In this paper, we propose a method to decide the secrecy of declassifiers to correct type errors in information flow analysis with an arbitrary secrecy lattice. The declassification which decreases the secrecy of some program constructs is an useful method to resolve illegal information flow. We have to decide the position and secrecy for declassifiers placement. Our previous method [1] can decide the position but not the secrecy of each declassifier since we assumed the secrecy lattice has only two elements. This paper shows how to derive the secrecy of declassifiers and proves that the derived secrecy always corrects a type error and is the best secrecy.

1　はじめに

型検査に基づく情報流解析は，プログラムが機密情報を外部に漏らさないことをプログラムを静的に解析して検査する手法であり，型システムとして実現される [2] [3] [4] [5]．データの型としてデータの機密度が利用され，機密度の低いデータが機密度の高いデータに依存しないことを表す非干渉性を満たすように型システムが構築される．非干渉性は，機密度の高いデータを用いて機密度の低いデータを計算するといった直接的な依存だけでなく，機密度の高いデータに基づく分岐先で機密度の低いデータを変更するといった間接的な依存も認めない．機密データ自体に加えて機密データを推測できる情報も一切漏らさないという意味で非干渉性はよい性質であるが，機密データの内容によって外部から観察可能な動作を変更することが禁止されるため，非干渉性を満たしながら実用的なプログラムを作成することは困難である．

この問題に対し，開発者が明示した場合に限り，本来は不正であると判断される機密度の高いデータから低いデータへの情報流を認める Declassification（非機密化）と呼ばれるアプローチが多く提案されている [6] [7] [8] [9]．例えば，Sabelfeld らは式 e の機密度を強制的に η とみなすことを意味する式レベルのプリミティブ `declassify`(e, η) を持つプログラミング言語とその型システムを提案している [7]．開発者は，`declassify` 式を適切に記述することで，機密度の高いデータから低いデータへの情報流を容認していることを型システムに示し，型システムは開発者の意図を踏まえて型検査を行う．Declassification は非干渉性の厳しい制約を緩和しながら安全なプログラムを構築するための有益な手法である．

しかし，一般的に，プログラムに含まれる不正な情報流は一通りではなく，複数の不正な情報流それぞれに対処する必要がある．そのため，Sabelfeld らの `declassify` 式のような非機密化プリミティブ (Declassifier) を手作業で適切に記述することはそれほど容易ではなく，非機密化プリミティブの記述位置を列挙する手法が提案されている [10] [11] [1]．これらの手法を用いると，不正な情報流を解消するためにはプログラム中のどこに非機密化プリミティブを記述すればよいか容易に知ることができる．我々が提案した手法 [1] では，情報流解析のための型システムを制約充足問題に帰着させ，充足不能な制約集合の Minimal Correction Subset(MCS) に含まれる

*Hiroaki Kuwabara, 立命館大学情報理工学部

†Yoshitoshi Kunieda, 立命館大学情報理工学部

$$
\begin{array}{rcl}
P & ::= & \overline{F} \\
F & ::= & \eta\ f(\overline{\eta\ x})\ B \\
B & ::= & \{\overline{\eta\ x;}\ \overline{S;}\} \\
S & ::= & x = e^l \mid \text{if } (e^l)\ B \text{ else } B \\
e & ::= & x \mid \text{true} \mid \text{false} \mid e^l == e^l \mid f(\overline{e^l}) \mid \texttt{declassify}(e, \eta)
\end{array}
$$

図 1　対象言語の文法

制約が由来するプログラムの構成要素を非機密化プリミティブの記述位置とする．MCS とは，取り除くことで元の制約集合が充足可能となる要素の極小部分集合を指す．

本稿では，我々の手法 [1] を拡張し，任意の機密度束を用いた情報流解析に対応させる．従来手法では，機密度は L と H の 2 段階であり，$L \sqsubseteq H$ を満たす機密度束 $(\{L, H\}, \sqsubseteq)$ を前提としたため，非機密化プリミティブが適用された式の機密度を常に低い機密度 L に設定していた．それにより不正な情報流も解消された．しかし，任意の機密度束を前提とする場合，非機密化プリミティブによって機密度をどこまで低下させるか決定する必要がある．機密度束の中で最小の値を選択すれば不正な情報流は確実に解消されるが，できるだけ機密度は低下させず，高い機密度を選択して不正な情報流を解消したい．本稿では，非機密化プリミティブが適用される式の新しい機密度を求める方法を示し，従来手法における機密度束に関する制約を緩和する．

本稿の構成は以下の通りである．2 節で対象とする言語と情報流解析のための型システムについて述べる．3 節で非機密化プリミティブが適用される式の新しい機密度を求める方法を示し，4 節で提案手法の実装と簡単な適用例について述べる．5 節で関連研究を挙げ，6 節でまとめる．

2　対象言語

本稿では，非機密化プリミティブとして Sabelfeld らの提案と類似した `declassify` 式を持つ簡単な手続き型言語と型システム [1] を対象とする．ただし，`declassify` 式では適用対象の式に加えて新しい機密度を指定する．機密度として，束 $(\mathcal{H}, \sqsubseteq)$ を構成する機密度の集合 $\mathcal{H}$ と機密度間の半順序関係 $\sqsubseteq$ を仮定する．この束を機密度束，束の要素を機密度定数とも呼ぶ．束の最小の要素を $\eta_\perp$ と書く．任意の $\eta \in \mathcal{H}$ に対し $\eta_\perp \sqsubseteq \eta$ である．

対象言語の文法を図 1 に示す．η は機密度定数である．データ型を bool 型に限定することでデータ型の記述を省略し，変数や関数の返り値の型として機密度のみを記述する．プログラム P は関数定義の並びである．$\overline{A}$ は長さ 0 以上の有限リストを表す略記である．F は関数定義であり，f が関数名を表す．B はブロック，S は文，e は式である．ブロックの先頭でローカル変数を宣言できる．関数の返り値は予約変数 $result$ への代入によって設定する．`declassify`(e, η) は e の機密度を η とみなしたいという開発者の意図を表し，e の機密度によらず `declassify`(e, η) の機密度は必ず η である．`declassify`(e, η) の意味は e と等価である．l は `declassify` を適用可能な式のプログラム中の位置を表すラベルであり，プログラム中のラベルはすべて異なるものとする．以下ではラベルが重要でない場合は省略する．

型付け規則を図 2 に示す．図中ではラベルは省略されている．PROG 規則がプログラム全体，FDEC 規則が関数定義，BLOCK 規則がブロック，ASSIGN 規則，IF 規則が文，その他が式の型付け規則である．$result$ は関数の返り値を表す変数，$ftype$は関数の型を取得する関数である．Δ は型環境であり，変数名からその機密度への関数である．プログラムを構成するすべての関数定義が型付け可能であればプログラムは型付け可能であり，引数と返り値に関する型環境の下で関数本体が型付

$$\frac{\vdash F_i \quad i \in \{1,\ldots,n\}}{\vdash F_1 \ \ldots \ F_n}\ [\text{PROG}] \qquad \frac{\overline{x : \eta_x}, result : \eta_r \vdash B : \eta_B}{\vdash \eta_r \ f(\overline{\eta_x \ x}) \ B}\ [\text{FDEC}]$$

$$\frac{\Delta, \overline{x : \eta_x} \vdash S_i : \eta_i \quad \eta \sqsubseteq \eta_i \quad i \in \{1,\ldots,n\}}{\Delta \vdash \{\overline{\eta_x \ x};\ S_1; \ldots S_n;\} : \eta}\ [\text{BLOCK}]$$

$$\frac{\Delta \vdash e : \eta_e \quad \eta_x = \Delta(x) \quad \eta_e \sqsubseteq \eta_x \quad \eta \sqsubseteq \eta_x}{\Delta \vdash x = e : \eta}\ [\text{ASSIGN}]$$

$$\frac{\Delta \vdash e : \eta_e \quad \Delta \vdash B_t : \eta_t \quad \Delta \vdash B_f : \eta_f \quad \eta_e \sqsubseteq \eta \quad \eta \sqsubseteq \eta_t \quad \eta \sqsubseteq \eta_f}{\Delta \vdash \text{if } (e) \ B_t \text{ else } B_f : \eta}\ [\text{IF}]$$

$$\frac{\eta = \Delta(x)}{\Delta \vdash x : \eta}\ [\text{VAR}] \qquad \frac{\Delta \vdash e_1 : \eta_1 \quad \Delta \vdash e_2 : \eta_2 \quad \eta_1 \sqsubseteq \eta \quad \eta_2 \sqsubseteq \eta}{\Delta \vdash e_1 == e_2 : \eta}\ [\text{COMP}]$$

$$\frac{c \in \{\text{true,false}\}}{\Delta \vdash c : \eta_\bot}\ [\text{CONST}] \qquad \frac{\Delta \vdash e : \eta_e}{\Delta \vdash \texttt{declassify}(e, \eta) : \eta}\ [\text{DECL}]$$

$$\frac{\Delta \vdash e_i : \eta_{e_i} \quad \eta_{e_i} \sqsubseteq \eta_i \quad i \in \{1,\ldots,n\} \quad \eta_1, \ldots, \eta_n \to \eta_r = \mathit{ftype}(f) \quad \eta_r \sqsubseteq \eta}{\Delta \vdash f(e_1, \ldots, e_n) : \eta}\ [\text{CALL}]$$

図 2　型付け規則

け可能であれば関数定義は型付け可能である．プログラムと関数定義は型付けできるか否かが重要であり，プログラムと関数定義の具体的な型は定義しない．ブロックの型判定式 $\Delta \vdash B : \eta$ および文の型判定式 $\Delta \vdash S : \eta$ は，型環境 Δ のもとで B あるいは S の実行によって変更される変数の機密度が η 以上であることを表す．式の型判定式 $\Delta \vdash e : \eta$ は，型環境 Δ のもとで式 e の機密度が η 以下であることを表す．

型検査は制約充足問題に帰着させることができる．型検査が成功するために満たされるべき制約集合を生成するアルゴリズムを図 3 に示す．図中の κ や添字付きの κ_i, κ_e などは機密度を表す変数である．図中の機密度変数 κ はすべてフレッシュである．つまり，それぞれの文や式に対して新しい機密度変数が用意される．その上で，図 2 の規則に従って機密度に関する制約集合を生成する．$\Delta \Vdash e : \kappa \parallel \mathsf{C}$ は型環境 Δ のもとで式 e が型付け可能であるために満たされるべき制約集合が C であることを表す．文やブロックについても同様である．一部の制約には由来する式を示すラベルを付ける．式 e^l に関する制約に $\kappa_1 \sqsubseteq^l \kappa_2$ のようにラベル l を付けることで，この制約がラベル l が付けられた式 e^l に由来することがわかる．

3　`declassify` 式における新しい機密度

型検査に失敗したプログラムに対して，`declassify` 式を適切に記述することで型検査を成功させることができる．`declassify` 式を記述するためには，記述位置と新しい機密度を求める必要がある．記述位置については従来手法 [1] に基づいて求める．

3.1　準備

以下，η, η' や添字付きの η_0 などは機密度定数を表すメタ変数，κ, κ' や添字付きの κ_i, κ_e は機密度変数を表すメタ変数，α, α' や添字付きの α_i, α_e などは機密度定数

$$\frac{\Vdash F_i \parallel \mathsf{C}_i \quad i \in \{1,\ldots,n\}}{\Vdash F_1 \ \ldots \ F_n \parallel \bigcup_i \mathsf{C}_i}\ [\text{C-PROG}] \qquad \frac{\overline{x : \eta_x}, result : \eta_r \Vdash B : \kappa_B \parallel \mathsf{C}}{\Vdash \eta_r \ f(\overline{\eta_x \ x}) \ B \parallel \mathsf{C}}\ [\text{C-FDEC}]$$

$$\frac{\Delta, \overline{x : \eta_x} \Vdash S_i : \kappa_i \parallel \mathsf{C}_i \quad i \in \{1,\ldots,n\}}{\Delta \Vdash \{\overline{\eta_x \ x};\ S_1; \ldots S_n;\} : \kappa \parallel \bigcup_i (\mathsf{C}_i \cup \{\kappa \sqsubseteq \kappa_i\})}\ [\text{C-BLOCK}]$$

$$\frac{\Delta \Vdash e : \kappa_e \parallel \mathsf{C_e} \quad \eta_x = \Delta(x)}{\Delta \Vdash x = e^l : \kappa \parallel \mathsf{C_e} \cup \{\kappa_e \sqsubseteq^l \eta_x, \kappa \sqsubseteq \eta_x\}}\ [\text{C-ASSIGN}]$$

$$\frac{\Delta \Vdash e : \kappa_e \parallel \mathsf{C_e} \quad \Delta \Vdash B_t : \kappa_t \parallel \mathsf{C_t} \quad \Delta \Vdash B_f : \kappa_f \parallel \mathsf{C_f}}{\Delta \Vdash \text{if}(e^l) \ B_t \text{ else } B_f : \kappa \parallel \mathsf{C_e} \cup \mathsf{C_t} \cup \mathsf{C_f} \cup \{\kappa_e \sqsubseteq^l \kappa, \kappa \sqsubseteq \kappa_t, \kappa \sqsubseteq \kappa_f\}}\ [\text{C-IF}]$$

$$\frac{\eta = \Delta(x)}{\Delta \Vdash x : \eta \parallel \emptyset}\ [\text{C-VAR}] \qquad \frac{c \in \{\text{true}, \text{false}\}}{\Delta \Vdash c : \eta_\perp \parallel \emptyset}\ [\text{C-CONST}]$$

$$\frac{\Delta \Vdash e_1 : \kappa_1 \parallel \mathsf{C_1} \quad \Delta \Vdash e_2 : \kappa_2 \parallel \mathsf{C_2}}{\Delta \Vdash e_1^{l_1} == e_2^{l_2} : \kappa \parallel \mathsf{C_1} \cup \mathsf{C_2} \cup \{\kappa_1 \sqsubseteq^{l_1} \kappa, \kappa_2 \sqsubseteq^{l_2} \kappa\}}\ [\text{C-COMP}]$$

$$\frac{\Delta \Vdash e_i : \kappa_i \parallel \mathsf{C_i} \quad i \in \{1,\ldots,n\} \quad \eta_1, \ldots, \eta_n \to \eta_r = \mathit{ftype}(f)}{\Delta \Vdash f(e_1^{l_1}, \ldots, e_n^{l_n}) : \kappa \parallel \bigcup_i (\mathsf{C_i} \cup \{\kappa_i \sqsubseteq^{l_i} \eta_i\}) \cup \{\eta_r \sqsubseteq \kappa\}}\ [\text{C-CALL}]$$

$$\frac{\Delta \Vdash e : \kappa \parallel \mathsf{C}}{\Delta \Vdash \texttt{declassify}(e, \eta) : \eta \parallel \mathsf{C}}\ [\text{C-DECL}]$$

図 3　制約集合生成アルゴリズム

あるいは機密度変数を表すメタ変数とする．

充足不能な制約集合には，以下に定義する 2 種類の部分集合が存在する．一般的に，充足不能な制約集合に対しいずれの部分集合も複数存在する．

定義 1. 制約集合 C の部分集合 $M \subseteq C$ について，$C \setminus M$ が充足可能であり，かつ，M に含まれる任意の制約 $c \in M$ に対して $C \setminus (M \setminus \{c\})$ が充足不能である時，M は Minimal Correction Subset(MCS) である．

定義 2. 制約集合 C の部分集合 $U \subseteq C$ について，U が充足不能であり，かつ，U に含まれる任意の制約 $c \in U$ に対して $U \setminus \{c\}$ が充足可能である時，U は充足不能な極小部分集合 (Minimal Unsatisfiable Subset, MUS) である．

制約集合 C のすべての MUS の集合を $\mathit{MUS}(C)$ と表し，$\mathit{MUS}(C)$ の中でラベル l が付けられた制約を含む MUS のみ集めた集合を $\mathit{MUS}(C, l)$ と表す．すなわち，$\mathit{MUS}(C, l) = \{M \mid M \in \mathit{MUS}(C), \exists \alpha, \alpha' \,.\, \alpha \sqsubseteq^l \alpha' \in M\}$ である．

次に，制約の列を定義する．

定義 3. 制約の列は以下のように定義される制約の並びである．

1. $\alpha \sqsubseteq \alpha'$ は制約の列である．
2. 制約の列 $\alpha_1 \sqsubseteq \kappa_2, \ldots, \kappa_{n-1} \sqsubseteq \alpha_n$ に対し，α_1 が機密度変数ならば $\alpha_0 \sqsubseteq \alpha_1, \alpha_1 \sqsubseteq \kappa_2, \ldots, \kappa_{n-1} \sqsubseteq \alpha_n$ は制約の列であり，α_n が機密度変数ならば $\alpha_1 \sqsubseteq \kappa_2, \ldots, \kappa_{n-1} \sqsubseteq \alpha_n, \alpha_n \sqsubseteq \alpha_{n+1}$ は制約の列である．
3. 1. および 2. によるものだけが制約の列である．

以下では，制約の列を単に列とも呼び，$\alpha_1 \sqsubseteq \cdots \sqsubseteq \alpha_n$ を $\alpha_1 \sqsubseteq \kappa_2, \kappa_2 \sqsubseteq \kappa_3, \ldots, \kappa_{n-1} \sqsubseteq \alpha_n$ の略記とする．定義より，列 $\alpha_1 \sqsubseteq \cdots \sqsubseteq \alpha_n$ において両端の

α_1 と α_n のみ機密度定数となり得る．列 $\alpha \sqsubseteq \cdots \sqsubseteq \alpha'$ に対し，α を列の下限，α' を列の上限と呼び，$\sup(\alpha \sqsubseteq \cdots \sqsubseteq \alpha') = \alpha'$ とする．

定義 4. $Qs(C, l, L)$ は，制約集合 C に含まれる制約から構成される列のうち，$L \setminus \{l\}$ に含まれるラベルが付けられた制約を含まない $\alpha \sqsubseteq^l \alpha' \sqsubseteq \cdots \sqsubseteq \eta'$ なる列の集合である．

例えば，制約集合 $C = \{L \sqsubseteq \kappa, \kappa \sqsubseteq^l \kappa', \kappa' \sqsubseteq M_1, \kappa' \sqsubseteq M_2, \kappa' \sqsubseteq^{l'} M_3\}$ に対し，$Qs(C, l, \{l'\}) = \{\kappa \sqsubseteq^l \kappa' \sqsubseteq M_1, \kappa \sqsubseteq^l \kappa' \sqsubseteq M_2\}$ である．

3.2 手順

`declassify` 式の記述は以下の手順によって行う．以下の手順は従来手法に基づいているが，`declassify` 式の記述位置それぞれについて新しい機密度を求める点が異なる．

1. 図 3 のアルゴリズムで生成された制約集合が充足可能であれば不正な情報流は存在しないと判断する．充足不能であれば制約集合のすべての MCS を求める．
2. ラベル付き制約のみを含む MCS を抽出する．
3. 抽出された各 MCS について，含まれる各制約に付けられたラベルに対応する式 e それぞれを，その MCS を解消するための `declassify` 式の記述位置とする．
4. それぞれの記述位置について新しい機密度を求めて `declassify` 式を記述する．ここで，式 e の位置に機密度 η で `declassify` 式を記述するとは，式 e を $\texttt{declassify}(e, \eta)$ に変更することを指す．

3.3 充足可能性判定

制約 $\kappa_1 \sqsubseteq \kappa_2$ の両辺が取り得る値は束の要素であるため，図 3 のアルゴリズムで得られた制約集合の充足可能性は標準的な制約解消アルゴリズムを用いて判定できる [12]．一部の制約にはラベルが付けられているが，ラベルはプログラム中の式とその式に由来する制約を関連付けるためのものであり機密度の順序には影響しない．そのため，充足可能性の判定はラベルを無視して行える．

3.4 ラベル付き制約のみを含む MCS と記述位置

制約集合が充足不能であれば，制約集合のすべての MCS を求める．MCS を求めるために， [13] [14] [15] などで提案されているアルゴリズムを利用できる．

ラベル付き制約は `declassify` 式の記述位置となり得るラベルが付けられた式に由来するため，MCS に含まれる各制約からラベルを抽出すれば，同じラベルを持つそれぞれの式が `declassify` 式の記述位置として得られる．ただし，MCS に含まれるすべての制約にラベルが付けられている必要がある．図 3 から明らかなように制約集合にはラベルのない制約も含まれるため，MCS がラベルのない制約を含む可能性もあるが， [1] の定理 2 と同様にしてラベル付き制約のみを含む MCS が 1 つ以上存在することを保証できる．MCS はすべての MUS の hitting set である [13] [15] [16] ので，すべての MUS がラベル付き制約を 1 つ以上含むことを示せばよい．MUS は充足不能な極小集合であるため，MUS に含まれるすべての制約を用いて $\eta \sqsubseteq \ldots \sqsubseteq \eta'$ なる列を構成できて，かつ $\eta \not\sqsubseteq \eta'$ である．よって，上限と下限がともに機密度定数である列にはラベル付き制約が含まれることを示せばよい．紙数の制約から，ここでは式から生成される制約集合についてのみ示すが，文やブロックから生成される制約集合についても同様である．

補題 1. 式 e に対し $\Delta \Vdash e : \alpha \parallel \mathsf{C}$ とする．この時，$\mathsf{C} = \emptyset$ であるか，あるいは C に含まれる制約は $\eta \sqsubseteq \cdots \sqsubseteq \alpha$ なる列を構成できる．また，C に含まれる制約が $\eta' \sqsubseteq \cdots \sqsubseteq \eta''$ なる列を構成できる場合，その列はラベル付き制約を含む．

証明： e の構造に関する帰納法による．

- $x, \mathrm{true}, \mathrm{false}$ の場合，$\mathsf{C} = \emptyset$.
- $e_1^{l_1} == e_2^{l_2}$ の場合，$\Delta \Vdash e_1 : \alpha_1 \parallel \mathsf{C_1}$，$\Delta \Vdash e_2 : \alpha_2 \parallel \mathsf{C_2}$ とすると $\mathsf{C} = \mathsf{C_1} \cup \mathsf{C_2} \cup$

$\{\alpha_1 \sqsubseteq^{l_1} \alpha, \alpha_2 \sqsubseteq^{l_2} \alpha\}$．帰納法の仮定より，$\mathsf{C_1} = \emptyset$ の場合，図 3 より α_1 は機密度定数であるため $\eta \sqsubseteq^{l_1} \alpha \in \mathsf{C}$ である．一方，$\mathsf{C_1}$ に含まれる制約が $\eta \sqsubseteq \ldots \sqsubseteq \alpha_1$ なる列を構成できる場合，α_1 が機密度変数ならば，$\mathsf{C_1} \subseteq \mathsf{C}$ かつ $\alpha_1 \sqsubseteq^{l_1} \alpha \in \mathsf{C}$ であるため C に含まれる制約は $\eta \sqsubseteq \cdots \sqsubseteq \alpha$ なる列を構成できる．α_1 が機密度定数ならば単に $\eta \sqsubseteq^{l_1} \alpha \in \mathsf{C}$ である．$\mathsf{C_2}$ についても同様である．C に含まれる制約が $\eta' \sqsubseteq \cdots \sqsubseteq \eta''$ なる列を構成できる場合，$\mathsf{C_1}$ に含まれる制約を構成する機密度変数の集合と $\mathsf{C_2}$ に含まれる制約を構成する機密度変数の集合は互いに素であり，α はいずれにも含まれないため，$\eta' \sqsubseteq \cdots \sqsubseteq \eta''$ なる列は $\mathsf{C_1}$ か $\mathsf{C_2}$ のいずれか一方に含まれる制約のみで構成される．この時，帰納法の仮定よりそのような列はラベル付き制約を含む．

他の場合も同様である． □

ラベル付き制約のみを含む MCS は複数存在する可能性があるため，そのような MCS それぞれについて，各制約のラベルを抽出しそれらのラベルが付けられた式の集合を `declassify` 式の記述位置候補とする．例えば，ラベル付き制約のみを含む MCS が 2 つあり，それぞれから抽出されたラベルの集合が $\{l_1, l_2\}$ および $\{l_3, l_4, l_5\}$ とすると，`declassify` 式の記述位置候補として $\{e_1^{l_1}, e_2^{l_2}\}$ および $\{e_3^{l_3}, e_4^{l_4}, e_5^{l_5}\}$ の 2 種類の式の集合が得られる．

3.5 機密度

`declassify` 式を記述する際には，対象の式の新しい機密度を決める必要がある．そのためには，式 e を $\texttt{declassify}(e, \eta)$ に変更することで生成される制約集合がどのように変化するか考えればよい．図 3 のアルゴリズムにおいて，ラベル l が付けられた式 e^l の機密度を表す変数を κ_e とすると，ラベル l を付けて生成される制約の形は $\kappa_e \sqsubseteq^l \alpha$ である．生成された制約集合が充足不能であるならば，制約集合の MUS U を $U = U_0 \cup \{\kappa_e \sqsubseteq^l \alpha\}$ とすると，U に含まれるすべての制約を用いて $\eta'' \sqsubseteq \cdots \sqsubseteq \kappa_e \sqsubseteq^l \alpha \sqsubseteq \cdots \sqsubseteq \eta'$ なる列を構成できる．ここで，U は充足不能であるため，機密度束において $\eta'' \not\sqsubseteq \eta'$ である．この時，式 e^l が $\texttt{declassify}(e, \eta)^l$ に変更されると，C-DECL 規則により $\kappa_e \sqsubseteq^l \alpha$ の代わりに $\eta \sqsubseteq^l \alpha$ が生成され，U は $U' = U_0 \cup \{\eta \sqsubseteq^l \alpha\}$ に変化する．U' に含まれる制約は 2 つの列 $\eta'' \sqsubseteq \cdots \sqsubseteq \kappa_e$ および $\eta \sqsubseteq^l \alpha \sqsubseteq \cdots \sqsubseteq \eta'$ を構成する．この時，前者の列を構成する制約の集合は充足可能であり，後者の列を構成する制約の集合は η として η' あるいは η' より小さい機密度を選べば充足できる．ただし，実際には $\kappa_e \sqsubseteq^l \alpha$ が複数の MUS や充足可能な部分集合に含まれる可能性を考慮する必要がある．例えば，$\{\kappa_e \sqsubseteq^l \alpha, \alpha \sqsubseteq \eta_1, \alpha \sqsubseteq \eta_2\}$ が制約集合の部分集合であるならば，η として $\eta \sqsubseteq \eta_1$ および $\eta \sqsubseteq \eta_2$ を同時に満たす機密度を選ばなければならない．

以上を踏まえ，新しい機密度を次のように求める．充足不能な制約集合を C，C のラベル付き制約のみを含む MCS に含まれる制約のラベルの集合を L とし，$l \in L$ とする．この時，ラベル l が付けられた式 e^l の新しい機密度を $\bigsqcap_{s \in Qs(C,l,L)} \sup(s)$ とする．ここで，$\sqcap$ は束の要素の交わりを求める演算である．制約集合 C に含まれる制約から構成される列の中で，ラベル l が付けられた制約を含み，MCS に含まれる l 以外のラベルが付けられた制約を含まず，上限が機密度定数であるような列すべての上限の交わりが式 e^l の新しい機密度である．これにより，新しい機密度を下限，機密度定数を上限とする列を構成する制約の集合は常に充足可能である．

新しい機密度を用いて `declassify` 式を記述することで，制約集合の充足不能性，すなわち不正な情報流を解消できる．

補題 2. 式, 文, ブロック D と D 中に出現する任意のラベル l に対し $\Delta \Vdash D : \alpha \parallel \mathsf{C} \cup \{\alpha_e \sqsubseteq^l \alpha'_e\}$ とする．この時，D 中の式 e^l を $\texttt{declassify}(e, \eta)^l$ に変更したものを D' とすると，$\Delta \Vdash D' : \alpha \parallel \mathsf{C} \cup \{\eta \sqsubseteq^l \alpha'_e\}$ である．

証明： D の構造に関する帰納法による． □

補題 3. 充足不能な制約集合 $C = C_0 \cup M$ について，M は C のラベル付き制約のみを含む MCS であって $M = \bigcup_{1 \le i \le n} \{\alpha_i \sqsubseteq^{l_i} \alpha_i'\}$ とし，M に含まれる制約のラベルの集合 $\{l_1, \ldots, l_n\}$ を L とする．この時，$\eta_i = \bigsqcap_{s \in Qs(C, l_i, L)} \sup(s)$ $(1 \le i \le n)$，$M' = \bigcup_{1 \le i \le n} \{\eta_i \sqsubseteq^{l_i} \alpha_i'\}$，$C' = C_0 \cup M'$ とすると，C' は充足可能である．

証明： C' が充足不能であると仮定すると，M が C の MCS であることから C_0 は充足可能であるため，$MUS(C', l_i) \neq \emptyset$ なる $l_i \in L$ が存在する．この時，ある列 $s' \in Qs(C', l_i, L)$，$s' = \eta_i \sqsubseteq^{l_i} \alpha_i' \sqsubseteq \ldots \sqsubseteq \eta_i'$ に対して $\eta_i \not\sqsubseteq \eta_i'$ でなければならない．しかし，$\eta_i = \bigsqcap_{s' \in Qs(C', l_i, L)} \sup(s')$ および $\eta_i' = \sup(s')$ であるため，$\eta_i = \eta_i' \sqcap \eta_i''$ なる η_i'' が存在し，必ず $\eta_i \sqsubseteq \eta_i'$ である．よって，任意の $l_i \in L$ について $MUS(C', l_i) = \emptyset$ であり，C' は充足可能である． □

定理 1. 関数定義 F に対し $\Vdash F \parallel \mathsf{C}$ とする．C が充足不能の場合，C のラベル付き制約のみを含む MCS に含まれる制約のラベルの集合を $L = \{l_1, \ldots, l_n\}$ とし，F 中の $e_i^{l_i}$ を $\mathtt{declassify}(e_i, \eta_i)^{l_i}$ に変更した関数定義 F' について $\Vdash F' \parallel \mathsf{C}'$ とすると，C' は充足可能である．ここで，$\eta_i = \bigsqcap_{s \in Qs(\mathsf{C}, l_i, L)} \sup(s)$ である．

証明： F の本体ブロックを B とすると，$\Vdash F \parallel \mathsf{C}$ より適当な Δ と κ が存在して $\Delta \Vdash B : \kappa \parallel \mathsf{C}$ である．C のラベル付き制約のみを含む MCS を $M = \{\alpha_1 \sqsubseteq^{l_1} \alpha_1', \ldots, \alpha_n \sqsubseteq^{l_n} \alpha_n'\}$ とおくと，適当な $\mathsf{C_0}$ が存在して $\mathsf{C} = \mathsf{C_0} \cup M$ であり，$\mathsf{C_0}$ は充足可能である．ここで，$M' = \{\eta_1 \sqsubseteq^{l_1} \alpha_1', \ldots, \eta_n \sqsubseteq^{l_n} \alpha_n'\}$ とすると，補題 2 より $\mathsf{C}' = \mathsf{C_0} \cup M'$ であり，補題 3 より C' は充足可能である． □

新しい機密度は制約集合の充足不能性を解消できる最も大きな機密度である．さらに大きな機密度を選択すると充足不能性は解消されない．

定理 2. 充足不能な制約集合 $C = C_0 \cup \{\alpha \sqsubseteq^{l} \alpha'\}$ と C のラベル付き制約のみを含む MCS に含まれるラベルの集合 L に対し，$l \in L$ および $\eta = \bigsqcap_{s \in Qs(C, l, L)} \sup(s)$ とする．さらに，$\eta' \not\sqsubseteq \eta$ を満たす η' に対し $C' = C_0 \cup \{\eta' \sqsubseteq^{l} \alpha'\}$ とする．この時，$MUS(C', l) \neq \emptyset$ である．

証明： $MUS(C', l) = \emptyset$ とすると，任意の列 $s' \in Qs(C', l, L)$ について $\eta' \sqsubseteq \sup(s')$ でなければならない．しかし，$\eta' \not\sqsubseteq \eta = \bigsqcap_{s' \in Qs(C', l, L)} \sup(s')$ ゆえ $\eta' \not\sqsubseteq \sup(s')$ が成り立つ列 $s' \in Qs(C', l, L)$ が存在する．よって，$MUS(C', l) \neq \emptyset$ である． □

4 実装と例

提案手法を [1] で作成したツールを拡張する形で実装した．実装には Java 言語を利用し，充足可能性の判定には制約解消系 Cream [17] を用いる．MCS と MUS の集合は [13] のアルゴリズム daa_min_unsat によって求める．プログラム中の式と生成される制約を対応付けるラベルは構文解析時に自動的に与えられる．

図 3 のアルゴリズムで生成された制約集合の解を Cream が発見できれば制約集合は充足可能であると判定する．解が発見できなければ充足不能であると判定し，制約集合の MCS をすべて求めてからラベル付き制約のみを含む MCS を抽出する．抽出された MCS に含まれる制約に付けられたラベルから得られる `declassify` 式の記述位置それぞれについて提案手法により新しい機密度を求める．

例として，$L \sqsubseteq UL, UL \sqsubseteq M, UL \sqsubseteq LM, LM \sqsubseteq UM_1, LM \sqsubseteq UM_2, M \sqsubseteq H, UM_1 \sqsubseteq H, UM_2 \sqsubseteq H$ を満たす機密度束 $(\{L, UL, M, LM, UM_1, UM_2, H\}, \sqsubseteq)$ のもとで，以下に示す関数 `f` の定義に対して提案手法を適用する．機密度束を図 4 に示す．図中の $\eta \leftarrow \eta'$ は $\eta \sqsubseteq \eta'$ の意味である．

```
L f() {
  H x; UM1 y; UM2 z; M w;
  if (w^6 ==^8 true^7) {
```

```
    x = true11;
    y = true14;
  } else {
    z = true18;
    w = true21;
  }
}
```

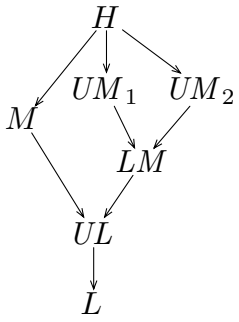

図 4　機密度束

ここで，6, 7, 8, 11, 14, 18, 21 といった数はラベルである[1]．このプログラムでは，機密度 M の変数 w の値を条件とする if 文内で 4 種類の変数への代入が行われており，このうち機密度が UM_1 の変数 y および UM_2 の変数 z への代入が不正な情報流となっている．

このプログラムに対し，図 3 のアルゴリズムによって以下の制約集合 C が生成される．

$$\{M \sqsubseteq^6 \kappa_8, L \sqsubseteq^7 \kappa_8, L \sqsubseteq^{11} H, \kappa_{12} \sqsubseteq H, \kappa_9 \sqsubseteq \kappa_{12}, L \sqsubseteq^{14} UM_1,$$
$$\kappa_{15} \sqsubseteq UM_1, \kappa_9 \sqsubseteq \kappa_{15}, L \sqsubseteq^{18} UM_2, \kappa_{19} \sqsubseteq UM_2, \kappa_{16} \sqsubseteq \kappa_{19}, L \sqsubseteq^{21} M,$$
$$\kappa_{22} \sqsubseteq M, \kappa_{16} \sqsubseteq \kappa_{22}, \kappa_8 \sqsubseteq^8 \kappa_{23}, \kappa_{23} \sqsubseteq \kappa_9, \kappa_{23} \sqsubseteq \kappa_{16}, \kappa_1 \sqsubseteq \kappa_{23}\}$$

制約集合 C は充足不能であり，2 つの MUS

$$\{M \sqsubseteq^6 \kappa_8, \kappa_8 \sqsubseteq^8 \kappa_{23}, \kappa_{23} \sqsubseteq \kappa_9, \kappa_9 \sqsubseteq \kappa_{15}, \kappa_{15} \sqsubseteq UM_1\}$$
$$\{M \sqsubseteq^6 \kappa_8, \kappa_8 \sqsubseteq^8 \kappa_{23}, \kappa_{23} \sqsubseteq \kappa_{16}, \kappa_{16} \sqsubseteq \kappa_{19}, \kappa_{19} \sqsubseteq UM_2\}$$

が存在する．ラベル付き制約のみを含む MCS を求めると $\{M \sqsubseteq^6 \kappa_8\}$ および $\{\kappa_8 \sqsubseteq^8 \kappa_{23}\}$ の 2 つの MCS が得られる．それぞれの MCS について含まれているラベルの集合は $\{6\}$ および $\{8\}$ であるので，ラベル 6 が付けられた if 文の条件に出現する変数 w の参照式，あるいはラベル 8 が付けられた if 文の条件に出現する比較式を declassify 式の記述位置とすればよいことがわかる．

declassify 式を記述するためには記述位置に加えて新しい機密度を求める必要がある．ここでは，ラベルが 8 の比較式を記述位置として選択した場合を考える．$Qs(C, 8, \{8\})$ を求めると

$$\{\kappa_8 \sqsubseteq^8 \kappa_{23} \sqsubseteq \kappa_9 \sqsubseteq \kappa_{12} \sqsubseteq H,$$
$$\kappa_8 \sqsubseteq^8 \kappa_{23} \sqsubseteq \kappa_9 \sqsubseteq \kappa_{15} \sqsubseteq UM_1,$$
$$\kappa_8 \sqsubseteq^8 \kappa_{23} \sqsubseteq \kappa_{16} \sqsubseteq \kappa_{19} \sqsubseteq UM_2,$$
$$\kappa_8 \sqsubseteq^8 \kappa_{23} \sqsubseteq \kappa_{16} \sqsubseteq \kappa_{22} \sqsubseteq M\}$$

が得られる．それぞれの列の上限は H, UM_1, UM_2, M であるので，新しい機密度は $H \sqcap UM_1 \sqcap UM_2 \sqcap M = UL$ と決まる．

ラベルが 8 の比較式に新しい機密度 UL で declassify 式を記述すると，関数 f の定義は以下のように変更される．

```
L f() {
  H x; UM1 y; UM2 z; M w;
  if (declassify(w6 == true7, UL)8) {
    x = true11;
    y = true14;
  } else {
    z = true18;
    w = true21;
  }
```

[1] 8 は比較式全体に対するラベルである．

```
}
```

この関数定義に対して，以下の制約集合 C' が得られる．

$$\{M \sqsubseteq^{6} \kappa_8, L \sqsubseteq^{7} \kappa_8, L \sqsubseteq^{11} H, \kappa_{12} \sqsubseteq H, \kappa_9 \sqsubseteq \kappa_{12}, L \sqsubseteq^{14} UM_1,$$
$$\kappa_{15} \sqsubseteq UM_1, \kappa_9 \sqsubseteq \kappa_{15}, L \sqsubseteq^{18} UM_2, \kappa_{19} \sqsubseteq UM_2, \kappa_{16} \sqsubseteq \kappa_{19}, L \sqsubseteq^{21} M,$$
$$\kappa_{22} \sqsubseteq M, \kappa_{16} \sqsubseteq \kappa_{22}, \boxed{UL \sqsubseteq^{8} \kappa_{23}}, \kappa_{23} \sqsubseteq \kappa_9, \kappa_{23} \sqsubseteq \kappa_{16}, \kappa_1 \sqsubseteq \kappa_{23}\}$$

四角形で囲まれた制約は，`declassify` 式を記述する前の関数定義に対する制約集合 C から変更があった制約を表している．C' は充足可能である．$Qs(C', 8, \{8\})$ を求めると

$$\{UL \sqsubseteq^{8} \kappa_{23} \sqsubseteq \kappa_9 \sqsubseteq \kappa_{12} \sqsubseteq H,$$
$$UL \sqsubseteq^{8} \kappa_{23} \sqsubseteq \kappa_9 \sqsubseteq \kappa_{15} \sqsubseteq UM_1,$$
$$UL \sqsubseteq^{8} \kappa_{23} \sqsubseteq \kappa_{16} \sqsubseteq \kappa_{19} \sqsubseteq UM_2,$$
$$UL \sqsubseteq^{8} \kappa_{23} \sqsubseteq \kappa_{16} \sqsubseteq \kappa_{22} \sqsubseteq M\}$$

であり，いずれの列についても，その列を構成する制約の集合は充足可能であることがわかる．さらに，新しい機密度として UL より大きな機密度を選択すると，いずれかの列についてその列を構成する制約の集合が充足できなくなることもわかる．

以上で挙げた適用例は，プロトタイプ実装を実際に動作させた結果を示している．Mac OS X 10.8.5，Core i7 1.8GHz，メモリ 4GB，Java 1.7.0_67 の環境において，制約集合の生成に約 4.6 ミリ秒，充足可能性判定に約 16 ミリ秒，MCS の列挙に約 150 ミリ秒，新しい機密度の計算に機密度 1 つあたり約 0.5 ミリ秒を要した．新しい機密度の計算は MCS の列挙など他の処理に比べ負荷が小さい．

5 関連研究

King らも不正な情報流が存在するプログラムに対して非機密化プリミティブの記述位置を求める手法を提案している [10] [11]．彼らの手法では，情報流解析のための型システムを制約充足問題に帰着させ，制約中に出現する変数や定数を節として制約の左辺から右辺への有向辺をもつ有向グラフを制約集合から構築する．有向グラフに高い機密度から低い機密度へのパスが存在すればそれが不正な情報流に対応するため，そのようなパスの切断点に対応する式に対して非機密化プリミティブを記述する．彼らの手法でも 2 段階の機密度束が仮定されている．任意の機密度束への対応についても [11] で述べられているが，束における最小の機密度への変更のみが考慮されており，本稿で提案したような不正な情報流を解消可能なできるだけ高い機密度を求めることは行われていない．

6 おわりに

本稿では，不正な情報流を含むプログラムに対し非機密化プリミティブを記述することで不正な情報流を解消する Declassification を，任意の機密度束のもとで実施する手法を提案した．本手法ではプログラム中の式に対して適用される非機密化プリミティブを採用したため，非機密化プリミティブの記述には，記述位置となる適用対象の式と，適用対象の式の新しい機密度を決定する必要がある．記述位置は従来手法により求められるため，本稿では新しい機密度を求める方法を示した．本手法では，情報流解析のための型検査を制約充足問題に帰着させ，非機密化プリミティブの適用対象である式に由来する制約を含む充足不能な極小部分集合および充足可能な極大部分集合それぞれから上限である機密度を抽出し，抽出された機密度の交わりを新しい機密度とする．提案手法で得られた機密度を用いて不正な情報流が解消できること，および提案手法で得られた機密度が不正な情報流を解消可能な最大の機密度であることを示した．

一般に，プログラム中の不正な情報流を解消するために必要な非機密化プリミティ

ブの記述位置は一意ではなく，複数の候補が存在し得る．候補に挙げられるすべての記述位置に対し，本稿の提案手法によって新しい機密度を求めることができるが，候補の選択は開発者に委ねられている．記述位置や新しい機密度といった情報から候補のフィルタリングや順位付けを行うことは今後の課題である．

本稿では，関数を利用可能な簡単な手続き型言語を対象としたが，対象言語を拡張してオブジェクトや例外処理といったプログラム構成要素に対応することも今後の課題である．非機密化プリミティブの適用対象を式以外の構成要素に拡張することを含め，非機密化プリミティブの記述位置や新しい機密度の求め方を検討する必要がある．

謝辞 本研究の一部は JSPS 科研費 24700036 の助成による．

参考文献

[1] 桑原寛明, 國枝義敏. 情報流解析における Declassifier の配置手法. ソフトウェア工学の基礎 XX(FOSE2013), pp. 15–24, 2013.

[2] 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. IEEE Computer Society Press, 2002.

[3] 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, pp. 757–770, 2008.

[4] 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.

[5] 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.

[6] Andrei Sabelfeld and David Sands. Dimensions and Principles of Declassification. In *Proceedings of the 18th IEEE Computer Security Foundations Workshop*, pp. 255–269. IEEE Computer Society Press, 2005.

[7] Andrei Sabelfeld and Andrew C. Myers. A Model for Delimited Information Release. In *Software Security - Theories and Systems*, Vol. 3233 of *Lecture Notes in Computer Science*, pp. 174–191. Springer Berlin Heidelberg, 2004.

[8] Gilles Barthe, Salvador Cavadini, and Tamara Rezk. Tractable Enforcement of Declassification Policies. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium*, pp. 83–97. IEEE Computer Society Press, 2008.

[9] Aslan Askarov and Andrei Sabelfeld. Localized Delimited Release: Combining the What and Where Dimensions of Information Release. In *Proceedings of the 2007 workshop on Programming languages and analysis for security*, PLAS '07, pp. 53–60. ACM, 2007.

[10] Dave King, Susmit Jha, Trent Jaeger, Somesh Jha, and Sanjit A. Seshia. On Automatic Placement of Declassifiers for Information-Flow Security. Technical Report NAS-TR-0083-2007, Network and Security Research Center, 2007.

[11] Dave King, Susmit Jha, Divya Muthukumaran, Trent Jaeger, Somesh Jha, and Sanjit A. Seshia. Automating Security Mediation Placement. In *Programming Languages and Systems*, Vol. 6012 of *Lecture Notes in Computer Science*, pp. 327–344. Springer Berlin Heidelberg, 2010.

[12] Jakob Rehof and Torben Æ. Mogensen. Tractable constraints in finite semilattices. *Science of Computer Programming*, Vol. 35, No. 2–3, pp. 191 – 221, 1999.

[13] James Bailey and Peter J. Stuckey. Discovery of Minimal Unsatisfiable Subsets of Constraints Using Hitting Set Dualization. In *Practical Aspects of Declarative Languages*, Vol. 3350 of *Lecture Notes in Computer Science*, pp. 174–186. Springer Berlin Heidelberg, 2005.

[14] Maria Garcia de la Banda, Peter J. Stuckey, and Jeremy Wazny. Finding All Minimal Unsatisfiable Subsets. In *Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming*, pp. 32–43. ACM, 2003.

[15] Mark H. Liffiton and Karem A. Sakallah. Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints. *Journal of Automated Reasoning*, Vol. 40, No. 1, pp. 1–33, 2008.

[16] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, Vol. 32, No. 1, pp. 57–95, 1987.

[17] Naoyuki Tamura. Cream: Class Library for Constraint Programming in Java. `http://bach.istc.kobe-u.ac.jp/cream/`.