一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SS2019-53(2020-03)

Rust プログラムの情報流解析のための型システム

長谷川健太† 桑原 寛明†† 國枝 義敏†††

† 立命館大学大学院情報理工学研究科
†† 南山大学情報センター
††† 立命館大学情報理工学部

あらまし 本稿では，Rust プログラムの情報流解析のための型システムを提案する．プログラミング言語 Rust の言語機能として危険なメモリ操作が発生しないことを静的に保証するための所有権と借用が存在する．所有権は，プログラムの任意の位置でそれぞれの値の所有者が唯一であることを保証し，データ競合を防ぐ．借用は，値へのアクセスに必要となる所有権の移動の省略する仕組みであり，参照を用いて実現される．実用的な Rust プログラムでは借用の活用が不可欠であるが，借用には参照に伴う情報流が存在する．提案する型システムでは，機密度を持つ参照型と，制御依存する条件の機密度を表すコンテキスト機密度を用いて，参照に伴う情報流を解析する．提案する型システムにより型付け可能なプログラムは機密情報を漏洩しない．

キーワード Rust，型システム，情報流解析，プログラム解析

1. はじめに

プログラミング言語 Rust [1] は，Mozilla が Web ブラウザエンジン [2] を実装するために開発したシステムプログラミング言語である．Rust には，メモリ安全性を静的に保証するための言語機能として所有権と借用がある．所有権は，メモリ中の値に対する所有者が唯一であることを保証する．値には所有者を通してのみアクセスが可能であり，所有権は代入や関数呼び出しの引数渡しに伴って移動する．借用は所有権を移動させずに値に対する一時的なアクセスを可能にする．借用は参照によって実現される．

Web ブラウザのようなパスワードなどの機密データを扱うソフトウェアは，機密データを外部に漏洩しないことが求められる．プログラムが機密データを外部に漏洩しないことを静的解析する手法として，型検査に基づく情報流解析が提案されている [3]〜[5]．型検査に基づく情報流解析では，データの機密度を型の一部として扱い，非干渉性に対し健全な型システムを構築する．機密度の低いデータが機密度の高いデータに依存する時に不正な情報流が存在するといい，非干渉性はプログラムに不正な情報流が存在しないことを表す．プログラムが型付け可能であれば機密データは外部に漏洩しない．

Rust で実用的なプログラムを記述する上で参照の利用は欠かせないが，参照を介した不正な情報流が存在する場合がある．C プログラムに対しポインタを介した不正な情報流を動的解析を用いて検出する手法は提案されているが [6]〜[8]，静的解析による手法は提案されていない．Rust プログラムを対象とする情報流解析の手法もこれまでに提案されていない．

本稿では，Rust プログラムを対象とする情報流解析のための型システムを提案し，参照を介した不正な情報流の静的検出を実現する．提案手法では，Rust のサブセット言語 Oxide [9] から所有権や参照等の言語機能を抽出し，型に対して機密度を加えて拡張したサブセット言語 FlowRust を定義する．型付け可能な FlowRust プログラム中には不正な情報流が含まれないように型システムを構築する．参照を介した不正な情報流が存在するプログラムが型付けできない例を示す．提案する型システムが非干渉性に対して健全であることを証明し，型付け可能なプログラムは不正な情報流を含まないことを保証する．

2. プログラミング言語 Rust

2.1 メモリ安全性

メモリ安全性とは，ダングリングポインタやダブルフリー，未初期化状態のメモリの使用，マルチスレッド時のデータ競合といったメモリを扱う上で危険であるとされるバグをプログラムが含まないことを指す．メモリ安全性を静的に保証するために，Rust は言語機能として所有権と借用を用意している．

2.2 所有権

所有権は，メモリ中の値の所有者が唯一であることを保証する．例えば変数への代入により，代入先の変数は代入された値の所有権を持つ．値に対する所有者が唯一であることを保証するために，代入の右辺や関数呼び出しの実引数に変数を指定すると，指定した変数の値に対する所有権が代入先の変数や関数の仮引数に対して移動する．所有権の移動後，所有権を失った変数を通して元々所有していた値にアクセスすることはできない．関数呼び出しにより仮引数に所有権が移動した値に関数の呼び出し側からアクセスするためには呼び出し側へ所有権を戻す必要がある．そのためには，所有権を戻したい値を関数の戻り値として指定しなければならず，所有権の規則に従うために煩雑なプログラムの記述を強いられる．

ただし，所有権は移動せず値がコピーされる場合もある．構造体オブジェクトやベクタオブジェクトのようなヒープ領域に確保される値の場合，値はコピーされず所有権が移動する．一方，整数値や真偽値などのプリミティブ値の場合，値がコピー

```
struct Point{x:i32, y:i32}
fn foo1(p:Point){ }
fn foo2(p:Point) -> Point { p }
fn foo3(r:&Point){ }
fn main(){
    let p1 = Point{x:1, y:1};
    let p11 = p1;
    let z1 = p1.x; //error

    let p2 = Point{x:1, y:1};
    foo1(p2);
    let z2 = p2.x; //error

    let p3 = Point{x:1,y:1};
    let p3 = foo2(p3);
    let z3 = p3.x; //OK

    let p4 = Point{x:1,y:1};
    foo3(&p4);
    let z4 = p4.x; //OK
}
```

図 1　所有権の移動と借用

され所有権は移動しない．所有権が移動せずに値がコピーされることを所有権のコピーと呼び，所有権が移動しない値の型をコピー型と呼ぶ(注1)．

所有権の移動のプログラム例を図 1 に示す．図 1 中の変数 p1, p2, p3, p4 はそれぞれ代入された構造体 Point のオブジェクトに対する所有権を持つ．変数 p11 に対して p1 を代入すると，p1 が持つ所有権が p11 に移動する．そのため，所有権を失った p1 を用いて構造体のフィールド x にアクセスしようとするとコンパイルエラーとなる．p2 を関数 foo1 に引数として渡すと，p1 が持つ所有権が foo1 の仮引数 p に移動するため，関数の呼び出し後に p2 を用いてオブジェクトにアクセスできない．関数の呼び出し後にアクセスするためには，関数 foo2 のように戻り値として所有権を持つ仮引数を明示的に指定して所有権を戻す必要がある．しかし，所有権を戻したい仮引数が多くなるほど関数の記述が煩雑になる．

2.3 借　　用

借用は，参照を利用して所有権を移動させずに値に対する一時的なアクセスを可能にする．参照はポインタの一種であり，共有参照と可変参照の 2 種類が存在する．

共有参照は参照先の値の読み出しのみが可能な参照である．同一の値に対して複数の共有参照を生成できるが共有参照を介した値の変更は不可能であり，複数の参照を介して同時に値が変更されるデータ競合を防止する．可変参照は参照先の値に対する読み出しと書き込みの両方が可能な参照である．参照先の値を所有する変数は可変である必要がある．可変参照を介した値の書き込みが存在する場合，同一の値に対する他の参照を介

(注1)：厳密には，所有権が移動しない型は Copy トレイトを実装しているためであるが，詳細は省略する．

```
fn main(){
    let h = false;   let l:&mut i32;
    let mut l1 = 1;  let mut l2 = 2;
    if h {l = &mut l1;} else {l = &mut l2;}
    *l = 4;
}
```

図 2　参照による不正な情報流が存在する例

したアクセスは禁止される．

図 1 の関数 foo3 の仮引数 r は構造体 Point の共有参照であり，Point のオブジェクトを借用する．foo3 の引数として p4 の参照を渡しており，所有権は r に移動しない．そのため，所有権を戻すための戻り値の指定も不要であり，関数の呼び出し後にも p4 を用いてフィールドにアクセスできる．

3.　型検査に基づく情報流解析

プログラム中のデータ y からデータ x が推測できる時，x から y への情報流が存在するという．例えば代入文 $y = x$; では，y の値は x の値から直接計算されるため，x から y への明示的な情報流が存在する．一方，if 文 if($x \leq 4$) $\{y = 1\}$ else $\{y = 3\}$ では，明示的な情報流は存在しないが，y の値から x の値を推測できるため，x から y への暗黙的な情報流が存在する．

情報流解析では，データの機密度に基づき不正な情報流の有無を検査する．機密度束 $(\mathcal{H}, \sqsubseteq)$ が機密度の種類と大小関係を定義し [10]，機密なデータに対して高い機密度，機密でないデータに対して低い機密度を指定する．例えば，機密度が $\eta_x \in \mathcal{H}$ のデータ x と $\eta_y \in \mathcal{H}$ のデータ y に対し代入文 $y = x$; がある時，$\eta_x \not\sqsubseteq \eta_y$ ならば x から y への情報流は不正である．情報流解析では，プログラム中のすべての情報流が不正でないことを検査する．型検査に基づく情報流解析は，データの機密度が型の一部を構成し，情報流が不正でないことが機密度に関する条件として表わされた型システムにより静的検査する手法である．

Rust プログラムには参照を介した情報流が存在する場合がある．例えば，図 2 のプログラムでは，変数 h の値に応じて変数 l1 の可変参照 &mut l1 あるいは変数 l2 の可変参照 &mut l2 を l に代入し，l を介して l1 あるいは l2 に 4 を代入する．もし変数 h の機密度が高く，変数 l1，l2 の機密度が低いとすると，l1 と l2 の値は外部に対して秘密ではないため l を介して l1 と l2 のどちらに値が代入されたかわかる．その結果，外部に対して秘密であるはずの if 文の条件 h の値が推測できるため，図 2 のプログラムには参照を介した不正な情報流が存在している．

同様な情報流はポインタなど間接参照の仕組みを有するプログラミング言語にも存在し，C プログラムにおけるポインタを介した不正な情報流を動的解析により検出する手法が提案されている [6]〜[8]．Rust の所有権システムに従いながら実用的なプログラムを記述するためには参照の利用が必須であるが，Rust プログラムを対象とする情報流解析の手法はこれまでに提案されていない．

$$
\begin{aligned}
p &::= \overline{S}\ \overline{F}\ \overline{I}\ \text{let}\ \overline{x\colon \tau};\ s;\\
S &::= \text{struct}\ sn\ \eta\ \{\overline{fld\colon T^{\eta}}\}\\
F &::= \text{fn}\ f\ (\overline{x\colon \tau})\colon \tau\ B\\
I &::= \text{impl}\ sn\ \{\overline{M}\}\\
M &::= \text{fn}\ m\ (\overline{x\colon \tau})\colon \tau\ \eta\ B\\
\tau &::= T^{\eta}_{\omega} \mid \&^{\eta}\omega\ T^{\eta}_{\omega}\\
T &::= \text{int} \mid \text{bool} \mid sn\\
\eta &::= L \mid H\\
\omega &::= \text{mut} \mid \text{shrd}\\
B &::= \{\text{let}\ \overline{x\colon \tau};\ s;\}\\
s &::= s; s \mid \pi \coloneqq e \mid x \coloneqq sn\{\overline{e}\} \mid x \coloneqq \&^{\eta}\ \omega\ \pi \mid x \coloneqq f(\overline{e})\\
&\quad \mid\ x \coloneqq e.m(\overline{e}) \mid B \mid \text{if}\ e\ B\ \text{else}\ B\\
e &::= \pi \mid i \mid \text{true} \mid \text{false} \mid e\ bop\ e \mid e == e\\
\pi &::= x \mid *\pi \mid \pi.fld
\end{aligned}
$$

図 3　FlowRust の構文

4. Rust プログラムの情報流解析のための型システム

Rust プログラムの情報流解析のための型システムを提案する．所有権や参照といった Rust の主要な言語機能に関する情報流解析に焦点を絞り，Rust のサブセット言語 Oxide [9] からタプルや配列，ジェネリクスなどを除き，機密度を追加した言語 FlowRust を定義する．型付け可能な FlowRust プログラム中に不正な情報流が存在しないように型システムを構築する．

4.1 構　　文

FlowRust の構文を図 3 に示す．図中の $\overline{S}$ は $S_1 S_2 \ldots S_n$ の略記であり，$\overline{F}$，$\overline{I}$，$\overline{M}$ も同様である．$\overline{e}$ は $e_1, e_2, \ldots, e_n$，$\overline{fld\colon T^{\eta}}$ は $fld_1\colon T_1{}^{\eta_1}, fld_2\colon T_2{}^{\eta_2}, \ldots, fld_n\colon T_n{}^{\eta_n}$，$\overline{x\colon \tau}$ は $x_1\colon \tau_1, x_2\colon \tau_2, \ldots, x_n\colon \tau_n$ の略記である．

S は機密度 η の構造体 sn の宣言であり，フィールドも定義する．メソッドは impl ブロック I で定義される．機密度 η は束 $(\{L, H\}, \sqsubseteq)$ の元であり，$L \sqsubseteq H$ を満たす．型 τ は，値の型 T^{η}_{ω} と，T 型の値を指す参照の型 $\&^{\eta}\omega\ T^{\eta}_{\omega}$ である．ω は可変性であり，mut は可変，shrd は不変を表す．$x \coloneqq \&^{\eta}\omega\ \pi$ は参照生成文であり，π の値を指す参照を生成して変数 x に代入する．η は参照の機密度の指定であり，ω が mut ならば可変参照，shrd ならば共有参照を生成する．π は代入の左辺に出現できる式である．$*\pi$ は参照解決式であり，π の参照先の値を取り出す．FlowRust プログラム p に対して，構造体名から構造体宣言への構造体テーブル ST を仮定する．

4.2 型付け規則

補助関数 $placestyp$ の定義を図 4，FlowRust の式の型付け規則を図 5 に示す．図中の $fields(sn)$ は構造体 sn のフィールドの列である．$placestyp(\pi, \tau)$ は，式 π が型 τ の時に π を通してアクセスできる式とその型を返す．τ が int あるいは bool の参照ならば参照解決式とその型を返す．τ が構造体の参照の場合，π の参照解決式および構造体のフィールド参照式とそれらの型を返す．τ が構造体の場合，π をレシーバとするフィールド参照式とその型を返す．FlowRust では，値の所有権を失った変数を型環境から削除することで所有権の移動を表現する．所有権が移動する状況を表すために，左辺式と右辺式を区別して型付け規則を定める．左辺式の型判定式は $\Psi \vdash_{le} \pi\colon \tau$，右辺式の型判定式は $\Psi \vdash_{re} e\colon \tau \Rightarrow \Psi'$ であり，所有権が移動した場合に型環境 Ψ が更新される．変数が持つ所有権は，変数の値が構造体オブジェクトあるいは可変参照の場合に移動する．所有権が移動すると構造体オブジェクトのフィールドへのアクセスや参照の解決もできなくなるため，フィールド参照式や参照解決式も型環境から削除する．参照解決式 $*\pi$ には参照先から参照元への情報流が存在する．そのため参照解決式の機密度は，参照の機密度と参照先の値の機密度以上でなければならない．

$$
\frac{}{placestyp(\pi, \text{int}^{\eta}_{\omega}) = \emptyset} \qquad \frac{}{placestyp(\pi, \text{bool}^{\eta}_{\omega}) = \emptyset}
$$

$$
\frac{T \in \{\text{int}, \text{bool}\}}{placestyp(\pi, \&^{\eta'}\omega\ T^{\eta}_{\omega}) = *\pi\colon T^{\eta}_{\omega}}
$$

$$
\frac{ST(sn) = \text{struct}\ sn\ \eta_{sn}\ \{\overline{fld\colon T^{\eta}}\}}{placestyp(\pi, \&^{\eta'}\omega\ sn^{\eta}_{\omega}) = *\pi\colon sn^{\eta_{sn}}_{\omega}, (*\pi).\overline{fld}\colon \overline{T^{\eta}_{\omega}}}
$$

$$
\frac{ST(sn) = \text{struct}\ sn\ \eta_{sn}\ \{\overline{fld\colon T^{\eta}}\}}{placestyp(\pi, sn^{\eta}_{\omega}) = \pi.\overline{fld}\colon \overline{T^{\eta}_{\omega}}}
$$

図 4　補助関数 $placestyp$

FlowRust の文の型付け規則を図 6 に示す．図中の $level(sn)$ は構造体 sn の機密度，$ftype$ と $mtype$ はそれぞれ関数とメソッドのシグネチャである．$\Psi_1 \oplus \Psi_2$ は，$x \in dom(\Psi_1)$ ならば $(\Psi_1 \oplus \Psi_2)(x) = \Psi_1(x)$，$x \in dom(\Psi_2)$ ならば $(\Psi_1 \oplus \Psi_2)(x) = \Psi_2(x)$ となるように定義する．x がいずれにも含まれない場合は未定義である．$\Theta_1 \oplus \Theta_2$ も同様である．文 s の型判定式は $\Theta; \Psi; \Sigma \vdash_s s\colon (\eta_s, \eta_h) \Rightarrow \Theta'; \Psi'$ であり，Θ，Ψ，Σ の下で文 s の型が (η_s, η_h) であることを表す．s の型付けに伴い Θ と Ψ が更新される．借用環境 Θ は参照生成時に参照先の式 π の可変参照が既に存在するか確認するために必要である．コンテキスト機密度 [11] のスタック Σ は，文 s を囲む if 文のネストに対応して各 if 文の条件式の機密度の結びを記録しており，スタックのトップが s のコンテキスト機密度である．文の型 (η_s, η_h) の η_s は文で代入される変数や引数の機密度の下限，η_h は文の実行によって変更されるヒープ上の値の機密度の下限を表す．規則 T-Ref は参照生成文の型付け規則である．参照先が同じ可変参照が既に存在する場合は新たに生成できない．右辺 $\&^{\eta'_r}\omega\ \pi$ から左辺 x への情報流があるため，右辺の参照の機密度 η'_r は x の参照の機密度 η_r 以下，右辺の参照先の機密度 η_π は x の参照先の機密度 η_x 以下である必要がある．図 2 のような参照を介した不正な情報流を防ぐため，参照生成文のコンテキスト機密度 η は右辺の参照の機密度 η'_r 以下であるだけでなく，右辺の参照先の機密度 η_e 以下でもなければならない．

FlowRust のプログラム, 関数宣言, impl ブロック, メソッド宣言の型付け規則を図 7 に示す．図中の struct $sn\ \eta\ \{\overline{fld\colon T^{\eta}}\} \vdash \overline{M}$ は, struct $sn\ \eta\ \{\overline{fld\colon T^{\eta}}\} \vdash M_1 \ldots$ struct $sn\ \eta\ \{\overline{fld\colon T^{\eta}}\} \vdash$

$$\frac{x\colon \tau \in \Psi}{\Psi \vdash_{le} x\colon \tau}\ \text{[T-Var]} \qquad \frac{\Psi \vdash_{le} \pi\colon \&^{\eta'}\omega\ T_{\omega}^{\eta''} \quad \eta' \sqsubseteq \eta \quad \eta'' \sqsubseteq \eta}{\Psi \vdash_{le} *\pi\colon T_{\omega}^{\eta}}\ \text{[T-DrfL]} \qquad \frac{\Psi \vdash_{le} \pi\colon sn_{\omega}^{\eta_{sn}} \quad \eta_{sn} \sqsubseteq \eta \quad fld\colon T^{\eta_f} \in fields(sn) \quad \eta_f \sqsubseteq \eta}{\Psi \vdash_{le} \pi.fld\colon T_{\omega}^{\eta}}\ \text{[T-FldL]}$$

$$\frac{x\colon sn_{\omega}^{\eta} \in \Psi \quad \overline{\pi}\colon\ \overline{T_{\omega}^{\eta}} = placestyp(x, sn_{\omega}^{\eta}) \quad \Psi' = \Psi \backslash \{x, \overline{\pi}\}}{\Psi \vdash_{re} x\colon T_{\omega}^{\eta} \Rightarrow \Psi'}\ \text{[T-Move]} \qquad \frac{x\colon T_{\omega}^{\eta} \in \Psi \quad T \in \{\mathrm{int}, \mathrm{bool}\}}{\Psi \vdash_{re} x\colon T_{\omega}^{\eta} \Rightarrow \Psi}\ \text{[T-Copy]}$$

$$\frac{x\colon \&^{\eta_r}\mathrm{mut}\ T_{\mathrm{mut}}^{\eta_x} \in \Psi \quad \overline{\pi}\colon\ \overline{T_{\mathrm{mut}}^{\eta}} = placestyp(x, \&^{\eta_r}\mathrm{mut}\ T_{\omega}^{\eta_x}) \quad \Psi' = \Psi \backslash \{x, \overline{\pi}\}}{\Psi \vdash_{re} x\colon \&^{\eta_r}\mathrm{mut}\ T_{\mathrm{mut}}^{\eta_x} \Rightarrow \Psi'}\ \text{[T-MoveRef]}$$

$$\frac{x\colon \&^{\eta_r}\mathrm{shrd}\ T_{\mathrm{shrd}}^{\eta} \in \Psi}{\Psi \vdash_{re} x\colon \&^{\eta_r}\mathrm{shrd}\ T_{\mathrm{shrd}}^{\eta} \Rightarrow \Psi}\ \text{[T-CopyRef]} \qquad \frac{\Psi \vdash_{le} \pi\colon \&^{\eta'}\omega\ T_{\omega}^{\eta''} \quad \eta'' \sqsubseteq \eta \quad \eta' \sqsubseteq \eta}{\Psi \vdash_{re} *\pi\colon T_{\omega}^{\eta} \Rightarrow \Psi}\ \text{[T-DrfR]}$$

$$\frac{\Psi \vdash_{le} \pi\colon sn_{\omega}^{\eta_{sn}} \quad \eta_{sn} \sqsubseteq \eta \quad fld\colon T^{\eta_f} \in fields(sn) \quad \eta_f \sqsubseteq \eta}{\Psi \vdash_{re} \pi.fld\colon T_{\omega}^{\eta} \Rightarrow \Psi}\ \text{[T-FldR]} \qquad \frac{}{\Psi \vdash_{re} i\colon \mathrm{int}_{\mathrm{shrd}}^{L} \Rightarrow \Psi}\ \text{[T-Int]} \qquad \frac{b \in \{\mathrm{true}, \mathrm{false}\}}{\Psi \vdash_{re} b\colon \mathrm{bool}_{\mathrm{shrd}}^{L} \Rightarrow \Psi}\ \text{[T-Bool]}$$

$$\frac{\Psi \vdash_{le} e_1\colon \mathrm{int}_{\omega_1}^{\eta_1} \quad \Psi \vdash_{le} e_2\colon \mathrm{int}_{\omega_2}^{\eta_2} \quad \eta_1 \sqsubseteq \eta \quad \eta_2 \sqsubseteq \eta}{\Psi \vdash_{re} e_1\ bop\ e_2\colon \mathrm{int}_{\mathrm{shrd}}^{\eta} \Rightarrow \Psi}\ \text{[T-BOp]} \qquad \frac{\Psi \vdash_{le} e_1\colon T_{\omega_1}^{\eta_1} \quad \Psi \vdash_{le} e_2\colon T_{\omega_2}^{\eta_2} \quad \eta_1 \sqsubseteq \eta \quad \eta_2 \sqsubseteq \eta}{\Psi \vdash_{re} e_1 == e_2\colon \mathrm{bool}_{\mathrm{shrd}}^{\eta} \Rightarrow \Psi}\ \text{[T-Eq]}$$

図 5　式の型付け規則

$$\frac{\Theta; \Psi; \Sigma \vdash_s s_1\colon (\eta_{s_1}, \eta_{h_1}) \Rightarrow \Theta'; \Psi' \qquad \Theta'; \Psi'; \Sigma \vdash_s s_2\colon (\eta_{s_2}, \eta_{h_2}) \Rightarrow \Theta''; \Psi'' \qquad \eta_s \sqsubseteq \eta_{s_1} \sqcap \eta_{s_2} \qquad \eta_h \sqsubseteq \eta_{h_1} \sqcap \eta_{h_2}}{\Theta; \Psi; \Sigma \vdash_s s_1; s_2\colon (\eta_s, \eta_h) \Rightarrow \Theta''; \Psi''}\ \text{[T-Seq]}$$

$$\frac{\Psi \vdash_{le} x\colon T_{\mathrm{mut}}^{\eta_x} \quad \Psi \vdash_{re} e\colon T_{\omega}^{\eta_e} \Rightarrow \Psi' \quad \eta_e \sqsubseteq \eta_x \quad \eta_s \sqsubseteq \eta_x \quad \overline{\pi}\colon\ \overline{\tau} = placestyp(x, T_{\omega}^{\eta_e}) \quad \Psi'' = \Psi' \cup \{x, \overline{\pi}\}}{\Theta; \Psi; \Sigma \vdash_s x \coloneqq e\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi''}\ \text{[T-Ass]}$$

$$\frac{\Psi \vdash_{le} x\colon \&^{\eta_r}\mathrm{mut}\ T_{\mathrm{mut}}^{\eta_x} \quad \eta'_r \sqsubseteq \eta_r \quad \eta_s \sqsubseteq \eta_x \sqcap \eta_r \quad \Psi \vdash_{re} e\colon \&^{\eta'_r}\omega\ T_{\omega}^{\eta_e} \Rightarrow \Psi' \quad \eta_e \sqsubseteq \eta_x \quad \eta \sqsubseteq \eta'_r \sqcap \eta_e \quad \overline{\pi}\colon\ \overline{\tau} = placestyp(x, \&^{\eta'_r}\omega\ T_{\omega}^{\eta_e}) \quad \Psi'' = \Psi' \cup \{x, \overline{\pi}\}}{\Theta; \Psi; \Sigma, \eta \vdash_s x \coloneqq e\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi''}\ \text{[T-RefAss]}$$

$$\frac{\Psi \vdash_{le} \pi\colon \&^{\eta_r}\mathrm{mut}\ T_{\mathrm{mut}}^{\eta_\pi} \quad \Psi \vdash_{re} e\colon T_{\omega}^{\eta_e} \Rightarrow \Psi' \quad \eta_e \sqsubseteq \eta_\pi \sqcap \eta_r \quad \eta_h \sqsubseteq \eta_e \quad \overline{\pi}\colon\ \overline{\tau} = placestyp(\pi, T_{\omega}^{\eta_e}) \quad \Psi'' = \Psi' \cup \{\pi, \overline{\pi}\}}{\Theta; \Psi; \Sigma \vdash_s *\pi \coloneqq e\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi''}\ \text{[T-DrfAss]}$$

$$\frac{\Psi \vdash_{le} \pi\colon sn_{\mathrm{mut}}^{\eta} \quad \Psi \vdash_{re} e\colon T_{\omega}^{\eta_e} \Rightarrow \Psi' \quad fld\colon T^{\eta_f} \in fields(sn) \quad \eta \sqsubseteq \eta_f \quad \eta_e \sqsubseteq \eta_f \quad \eta_h \sqsubseteq \eta_f \quad \overline{\pi}\colon\ \overline{\tau} = placestyp(\pi.fld, T_{\omega}^{\eta_e}) \quad \Psi'' = \Psi' \cup \{\pi.fld, \overline{\pi}\}}{\Theta; \Psi; \Sigma \vdash_s \pi.fld \coloneqq e\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi''}\ \text{[T-FldAss]}$$

$$\frac{\Psi \vdash_{le} x\colon sn_{\mathrm{mut}}^{\eta} \quad \Psi \vdash_{re} e_1\colon T_{1\omega_1}^{\eta_1} \Rightarrow \Psi_1\ \ldots\ \Psi_{n-1} \vdash_{re} e_n\colon T_{n\omega_n}^{\eta_n} \Rightarrow \Psi_n \quad fields(sn) = \overline{fld\colon T^{\eta_f}} \quad \forall i \in \{1, \ldots, n\}.\eta_i \sqsubseteq \eta_{f_i} \quad level(sn) \sqsubseteq \eta \quad \eta_s \sqsubseteq \eta \quad \eta_h \sqsubseteq level(sn) \quad \overline{\pi}\colon\ \overline{\tau} = placestyp(x, sn) \quad \Psi' = \Psi_n \cup \{x, \overline{\pi}\}}{\Theta; \Psi; \Sigma \vdash_s x \coloneqq sn\{\overline{e}\}\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi'}\ \text{[T-Struct]}$$

$$\frac{\Psi \vdash_{le} x\colon \&^{\eta_r}\mathrm{mut}\ T_{\mathrm{mut}}^{\eta_x} \quad \Psi \vdash_{re} \pi\colon T_{\omega}^{\eta_\pi} \Rightarrow \Psi' \quad \Theta' = \Theta \cup \{{}^{\omega}\pi\} \quad \eta_s \sqsubseteq \eta_r \quad \eta'_r \sqsubseteq \eta_r \quad \eta_\pi \sqsubseteq \eta_x \quad \eta \sqsubseteq \eta'_r \sqcap \eta_\pi \quad \eta_h \sqsubseteq \eta'_r \quad {}^{\mathrm{mut}}\pi \notin \Theta \quad \overline{\pi}\colon\ \overline{\tau} = placestyp(x, \&^{\eta'_r}\omega\ T_{\omega}^{\eta_\pi}) \quad \Psi'' = \Psi' \cup \{x, \overline{\pi}\}}{\Theta; \Psi; \Sigma, \eta \vdash_s x \coloneqq \&^{\eta'_r}\omega\ \pi\colon (\eta_s, \eta_h) \Rightarrow \Theta'; \Psi''}\ \text{[T-Ref]}$$

$$\frac{\Theta; \Psi, \overline{x\colon \tau}; \Sigma \vdash_s s\colon (\eta_s, \eta_{h_s}) \Rightarrow \Theta'; \Psi' \quad \eta_s \sqsubseteq \eta_{s_s} \quad \eta_h \sqsubseteq \eta_{h_s}}{\Theta; \Psi; \Sigma \vdash_s \{\mathrm{let}\ \overline{x\colon \tau};\ s;\}\colon (\eta_s, \eta_h) \Rightarrow \Theta'; \Psi'}\ \text{[T-Blk]}$$

$$\frac{\Psi \vdash_{le} x\colon T_{\mathrm{mut}}^{\eta_x} \quad ftype(f) = \overline{x\colon T_{\omega}^{\eta}} \to T_{\omega}^{\eta_r} \quad \Psi_{i-1} \vdash_{re} e_i\colon T_{i\omega_i}^{\eta_{e_i}} \Rightarrow \Psi_i \quad \Psi_0 = \Psi \quad i \in \{1, \ldots, n\} \quad \eta_{e_i} \sqsubseteq \eta_i \quad \eta_r \sqsubseteq \eta_x \quad \eta_s \sqsubseteq \eta_x}{\Theta; \Psi; \Sigma \vdash_s x \coloneqq f(\overline{e})\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi_n}\ \text{[T-FCall]}$$

$$\frac{\Psi \vdash_{le} e\colon sn_{\omega}^{\eta_{sn}} \quad mtype(m, sn) = \overline{x\colon T_{\omega}^{\eta}} \xrightarrow{\eta'_h} T_{\omega}^{\eta_r} \quad \Psi \vdash_{le} x\colon T_{\mathrm{mut}}^{\eta_x} \quad \Psi_{i-1} \vdash_{re} e_i\colon T_{i\omega_i}^{\eta_{e_i}} \Rightarrow \Psi_i \quad \Psi_0 = \Psi \quad \forall i \in \{1, \ldots, n\} \quad \eta_{e_i} \sqsubseteq \eta_i \quad \eta_{sn} \sqsubseteq \eta'_h \quad \eta_h \sqsubseteq \eta'_h \quad \eta_r \sqsubseteq \eta_x \quad \eta_{sn} \sqsubseteq \eta_x \quad \eta_s \sqsubseteq \eta_x}{\Theta; \Psi; \Sigma \vdash_s x \coloneqq e.m(\overline{e})\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi_n}\ \text{[T-MCall]}$$

$$\frac{\Psi \vdash_{le} e\colon \mathrm{bool}_{\omega}^{\eta_e} \qquad \Sigma' = \Sigma, \eta, \eta \sqcup \eta_e \qquad \Theta; \Psi; \Sigma' \vdash_s B_i\colon (\eta_{s_i}, \eta_{h_i}) \Rightarrow \Theta_i; \Psi_i \qquad \eta_s \sqsubseteq \eta_{s_1} \sqcap \eta_{s_2} \qquad \eta_e \sqsubseteq \eta_s \sqcap \eta_h \qquad \eta_h \sqsubseteq \eta_{h_1} \sqcap \eta_{h_2} \qquad i = 1, 2}{\Theta; \Psi; \Sigma, \eta \vdash_s \mathrm{if}\ e\ B_1\ \mathrm{else}\ B_2\colon (\eta_s, \eta_h) \Rightarrow \Theta_1 \oplus \Theta_2; \Psi_1 \oplus \Psi_2}\ \text{[T-If]}$$

図 6　文の型付け規則

M_n の略記である．プログラム直下の文，関数とメソッドの本体ブロックが型付けできる必要がある．

4.3　操作的意味論

FlowRust プログラムの意味は大ステップの操作的意味論により定義する．プログラムの状態をストア ψ，ヒープ μ，型環境 σ の組とし，文や式の評価に伴い状態が更新される．文の評価値は skip であり，式の評価値は整数，真偽値，ロケーション，オブジェクトの状態のいずれかである．ストアは変数名から値への関数，ヒープはアドレスから構造体オブジェクトや参照が生成された値への関数である．ロケーションはヒープ上のアドレスであり，オブジェクトの状態はフィールド名からフィールドの値への関数である．評価対象のプログラムは型付け可能で

$$\frac{\vdash \overline{F} \vdash \overline{I} \quad \Theta; \Psi, \overline{x\colon \tau}; \Sigma \vdash_s s\colon (\eta_s, \eta_{h_s}) \Rightarrow \Theta'; \Psi'}{\Theta; \Psi; \Sigma \vdash \overline{S}\ \overline{F}\ \overline{I}\ \text{let}\ \overline{x\colon \tau};\ s;}\ \text{[T-Prog]} \qquad \frac{ST(sn) = \text{struct}\ sn\ \eta\ \{\overline{fld\colon T^{\eta}}\} \quad \text{struct}\ sn\ \eta\ \{\overline{fld\colon T^{\eta}}\} \vdash \overline{M}}{\vdash \text{impl}\ sn\ \{\overline{M}\}}\ \text{[T-Impl]}$$

$$\frac{\emptyset; \overline{x\colon \tau}, res\colon \tau_r; \emptyset \vdash_s B\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi}{\vdash \text{fn}\ f(\overline{x\colon \tau})\colon \tau_r\ B}\ \text{[T-Func]} \qquad \frac{\emptyset; \overline{x\colon \tau}, self\colon sn^{\eta}_{\text{mut}}, res\colon \tau_r; \emptyset \vdash_s B\colon (\eta_s, \eta_h) \Rightarrow \Theta; \Psi}{\text{struct}\ sn\ \eta\ \{\overline{fld\colon T^{\eta}}\} \vdash \text{fn}\ m\ (\overline{x\colon \tau})\colon \tau_r\ \eta_h\ B}\ \text{[T-Method]}$$

図 7　プログラムの型付け規則

$$\frac{}{\langle \psi; \mu; \sigma \vdash x \rangle \Downarrow_l \langle \psi; \mu; \sigma \vdash \psi(x) \rangle}\ \text{[E-Var]} \qquad \frac{\sigma(x) \in \{sn^{\eta}_{\omega}, \&^{\eta'}\text{mut}\ T^{\eta}_{\text{mut}}\}}{\langle \psi; \mu; \sigma \vdash x \rangle \Downarrow_r \langle \psi \backslash x; \mu; \sigma \backslash x \vdash \psi(x) \rangle}\ \text{[E-Move]}$$

$$\frac{\sigma(x) \in \{\text{int}^{\eta}_{\omega}, \text{bool}^{\eta}_{\omega}, \&^{\eta'}\text{shrd}\ T^{\eta}_{\text{shrd}}\}}{\langle \psi; \mu; \sigma \vdash x \rangle \Downarrow_r \langle \psi; \mu; \sigma \vdash \psi(x) \rangle}\ \text{[E-Copy]} \qquad \frac{\langle \psi; \mu; \sigma \vdash \pi \rangle \Downarrow_r \langle \psi'; \mu'; \sigma' \vdash v \rangle \quad l = fresh(valtyp(v), \mu')}{\langle \psi; \mu; \sigma \vdash x := \&^{\eta}\omega\ \pi \rangle \Downarrow \langle \psi'[x \mapsto l]; \mu'[l \mapsto v]; \sigma' \vdash \text{skip} \rangle}\ \text{[E-Ref]}$$

図 8　FlowRust の操作的意味論（抜粋）

```
let h:shrd bool^{H}, l1:mut int^{L},
    l2: mut int^{L}, l:&^{H} mut int^{H}_{mut};
h = false;  l1 = 1;  l2 = 2;
if h {l := &^{H} mut l1;} else{l := &^{L} mut l2;}
*l := 4;
```

図 9　図 2 に対応する FlowRust プログラム

あることが前提である．

FlowRust に特徴的な意味を抜粋して図 8 に示す．$fresh$ は新しいロケーションを生成する関数，$valtyp(v)$ は値 v の型を表す．規則 E-Var は左辺の変数の意味であり単に変数の値を取り出す．一方，右辺の変数は所有権が移動する型の場合は規則 E-Move によりストアと型環境から変数が削除される．所有権が移動しない型の場合は規則 E-Copy により削除は行われない．規則 E-Ref は参照生成文の意味であり，参照先の値をヒープ上の新しいアドレスに格納する．

4.4 適 用 例

参照を介する不正な情報流が存在する図 2 に対応する図 9 の FlowRust プログラムに対し提案する型付け規則を適用する．

if 文の導出木を図 10，if 文の then 節における参照生成文の導出木を図 11 に示す．ただし，

$$\begin{cases} \Psi = h\colon \text{bool}^{H}_{\text{shrd}}, l1\colon \text{int}^{L}_{\text{mut}}, l2\colon \text{int}^{L}_{\text{mut}}, l\colon \&^{H}\text{mut int}^{H}_{\text{mut}} \\ B_1 = \{l := \&^{H}\text{mut}\ l1;\} \\ B_2 = \{l := \&^{L}\text{mut}\ l2;\} \end{cases}$$

である．図 10 では，規則 T-If により if 文の条件式 h の機密度 H をコンテキスト機密度のスタック Σ に記録して then 節と else 節に対応するブロック B_1, B_2 を型付けする．図 11 では，参照生成文 $l := \&^{H}\text{mut}\ l1;$ に対して規則 T-Ref を適用する．コンテキスト機密度 H が代入元の参照の機密度 H と参照先の変数 $l1$ の機密度 L の交わり $H \sqcap L$ 以下である必要があるが，$H \sqcap L = L$ ゆえ $H \not\sqsubseteq H \sqcap L$ であるためこの制約を満たしておらず，型付け不能である．

5. 非干渉性に対する健全性

非干渉性 [12] とは，プログラム中の機密情報が公開情報に影響を与えないことを表す．プログラムが非干渉性を満たす場合，公開情報を観測しても機密情報自体や機密情報の変更はわからない．本稿で提案する型システムは非干渉性に対して健全である，すなわち型付け可能なプログラムは非干渉性を満たす．非干渉性を形式化するため，プログラムのある 2 つの状態について，機密度 L のデータはすべて等価であることを表す L 等価関係を定義する．

L 等価関係の定義を定義 1 に示す．ここで，$[\![state(sn)]\!]$ は構造体 sn の構造体オブジェクトの状態の集合，$[\![\Psi]\!]$ はストアの集合，$[\![Heap]\!]$ はヒープの集合，$loctyp(l)$ はロケーション l の値の型である．

［定義 1］ ストアに対する L 等価関係 $\lesssim_L$ と，構造体オブジェクトの状態およびヒープそれぞれに対する L 等価関係 $\sim_L$ を定義する．$LLoc$ は機密度が L のロケーションの集合である．

- $s, s' \in [\![state(sn)]\!]$ について，$s \sim_L s'$ **iff** $\forall (fld\colon T^{\eta}) \in fields(sn).\eta = L \Longrightarrow s(fld) = s'(fld)$
- 型環境 Ψ, Ψ' が $dom(\Psi') \subseteq dom(\Psi)$ である時，$\psi \in [\![\Psi]\!], \psi' \in [\![\Psi']\!]$ について，$\psi' \lesssim_L \psi$ **iff** $\forall x \in dom(\Psi').(\Psi'(x) = T^{L}_{\omega} \vee \Psi'(x) = \&^{L}\omega\ T^{L}_{\omega}) \Longrightarrow \psi(x) = \psi'(x)$
- $\mu, \mu' \in [\![Heap]\!]$ について，$\mu \sim_L \mu'$ **iff** $dom(\mu) \cap LLoc = dom(\mu') \cap LLoc$ かつすべての $l \in dom(\mu) \cap LLoc$ に対し $\mu(l) \in [\![state(loctyp(l))]\!]$ ならば $\mu(l) \sim_L \mu'(l)$，さもなければ $\mu(l) = \mu'(l)$ である．

型付け可能な FlowRust プログラムを構成するすべての文と式について，L 等価な 2 つのヒープとストアの下で実行した後の 2 つのヒープとストアは L 等価である．式については，機密度が L であれば値も等しい．これらの等価性は文あるいは式の型導出に関する帰納法により証明できる．

［定理 1］ L 等価な 2 つのヒープとストアにおいて型付け可能なプログラム $\overline{S}\ \overline{F}\ \overline{I}$ let $\overline{x\colon \tau};\ s;$ を実行すると，実行後のヒープとストアも L 等価である．すなわち，$\Theta; \Psi; \Sigma \vdash \overline{S}\ \overline{F}\ \overline{I}$ let $\overline{x\colon \tau};\ s;$ かつ $\psi_1 \lesssim_L \psi_2$ かつ $\mu_1 \sim_L \mu_2$ の時，$i = 1, 2$ に対し $\langle \psi_i; \mu_i; \sigma_i \vdash \overline{S}\ \overline{F}\ \overline{I}$ let $\overline{x\colon \tau};\ s; \rangle \Downarrow \langle \psi'_i; \mu'_i; \sigma'_i \vdash \text{skip} \rangle$ ならば，$\psi'_1 \lesssim_L \psi'_2$ かつ $\mu'_1 \sim_L \mu'_2$ である．

6. 関 連 研 究

Rust のサブセットの形式化がいくつか提案されている．Reed らは，Rust のユニークポインタや借用，参照などの機能を有するサブセット言語 Patina を形式化した [13]．Benitez らは，Rust のサブセット言語 Metal において，所有権や借用を *capabilities*

$$\dfrac{\dfrac{}{\Psi \vdash_{le} h\colon \mathrm{bool}^{H}_{\mathrm{shrd}}}\ [\text{T-Var}] \qquad \Sigma' = \Sigma, \eta, \eta \sqcup H \quad \Theta; \Psi; \Sigma', H \vdash_s B_1\colon (?,?) \Rightarrow ?;? \quad \Theta; \Psi; \Sigma', H \vdash_s B_2\colon (?,?) \Rightarrow ?;? \quad \begin{array}{c} ? \sqsubseteq ?\sqcap ? \quad ? \sqsubseteq ?\sqcap ? \\ H \sqsubseteq ?\sqcap ? \quad i = 1, 2 \end{array}}{\Theta; \Psi; \Sigma, \eta \vdash_s \text{if } h\ B_1 \text{ else } B_2\colon (?,?) \Rightarrow ?;?}\ [\text{T-If}]$$

図 10　if 文の型導出木

$$\dfrac{\dfrac{}{\Psi \vdash_{le} l\colon \&^{H}\mathrm{mut}\ \mathrm{int}^{H}_{\mathrm{mut}}}\ [\text{T-Var}] \qquad \dfrac{}{\Psi \vdash_{re} l1\colon \mathrm{int}^{L}_{\mathrm{mut}} \Rightarrow \Psi}\ [\text{T-Copy}] \qquad \begin{array}{c} {}^{\mathrm{mut}}\, l1 \notin \Theta \quad \Theta' = \Theta \cup \{{}^{\mathrm{mut}}\, l1\} \quad H \sqsubseteq H \\ L \sqsubseteq H \quad L \sqsubseteq H \quad \boxed{H \not\sqsubseteq H \sqcap L} \end{array}}{\Theta; \Psi; \Sigma', H \vdash_s l := \&^{H}\mathrm{mut}\ l1\colon (?,?) \Rightarrow ?;?}\ [\text{T-Ref}]$$

図 11　参照生成文 $l := \&^{H}\mathrm{mut}\ l1$ の型導出木

の概念を用いて形式化した [14]．Aaron らは，Rust の型システムを形式化するために，型推論などの複雑な機能を除外したサブセット言語 Oxide を提案している [9]．本研究のサブセット言語 FlowRust は Oxide に基づいているが，所有権や参照と情報流解析との関係に焦点を当てており，他のサブセット言語よりも小さなサブセット言語である．

Rust プログラムを形式検証する手法が提案されている．Lindner らは，配列の境界外アクセスなどで発行される実行時例外を記号実行によって静的に検出する手法を提案している [15]．Astrauskas らは，Rust プログラム中に開発者がプログラムの仕様を記述できる言語を定義し，記述された仕様をプログラムが満たすかどうか検証する手法を提案している [16]．本研究の提案手法では，Rust プログラムが機密情報を漏洩していないことを型システムにより静的に検査する．

Baranowski らは Rust プログラムの情報流を自動検証するツールを実装したが，非干渉性に対する議論はなされていない [17]．本研究では，Rust プログラムを対象とする情報流解析のための型システムを提案し，提案する型システムが非干渉性に対して健全であることを示した．

7. おわりに

本稿では，Rust プログラムの情報流解析のための型システムを提案し，Rust プログラム中の参照を介する不正な情報流の静的検出を実現した．Rust のサブセット言語 Oxide から Rust の主要な言語機能である所有権や参照を抽出し，型に対して機密度を加えて拡張したサブセット言語 FlowRust を定義した．型付け可能な FlowRust プログラム中に不正な情報流が含まれないように型システムを構築した．提案する型システムが非干渉性に対して健全であることを証明し，型付け可能なプログラムは不正な情報流を含まないことを示した．

今後の課題として，参照の参照や変数以外の所有者への対応，Rust の並行処理や非同期処理への対応，型付け可能な FlowRust プログラムから Rust プログラムへの変換が挙げられる．

謝辞　本研究の一部は JSPS 科研費 JP15K00112, JP17K12666，JP18K11241 および 2019 年度南山大学パッヘ研究奨励金 I-A-2 の助成による．

文　献

[1] N.D. Matsakis and F.S. Klock, II, “The Rust Language,” Proceedings of the 2014 ACM SIGAda Annual Conference on High Integrity Language Technology, pp.103–104, ACM, 2014.

[2] B. Anderson, L. Bergstrom, M. Goregaokar, J. Matthews, K. McAllister, J. Moffitt, and S. Sapin, “Engineering the Servo Web Browser Engine Using Rust,” Proceedings of the 38th International Conference on Software Engineering Companion, pp.81–89, ACM, 2016.

[3] D. Volpano, C. Irvine, and G. Smith, “A Sound Type System for Secure Flow Analysis,” J. Comput. Secur., vol.4, no.2-3, pp.167–187, 1996.

[4] A. Sabelfeld and A.C. Myers, “Language-Based Information-Flow Security,” IEEE J.Sel. A. Commun., vol.21, no.1, pp.5–19, 2006.

[5] A. Banerjee and D.A. Naumann, “Secure Information Flow and Pointer Confinement in a Java-like Language,” Proceedings of the 15th IEEE Workshop on Computer Security Foundations, p.253, IEEE Computer Society, 2002.

[6] M. Assaf, “From Qualitative to Quantitative Program Analysis: Permissive Enforcement of Secure Information Flow,” PhD thesis, Université Rennes 1, 2015.

[7] M. Assaf, J. Signoles, F. Tronel, and É. Totel, “Program Transformation for Non-interference Verification on Programs with Pointers,” IFIP International Information Security Conference, pp.231–244, 2013.

[8] G. Barany and J. Signoles, “Hybrid Information Flow Analysis for Real-World C Code,” Tests and Proofs, vol.10375, pp.23–40, LNCS, Springer International Publishing, 2017.

[9] A. Weiss, D. Patterson, N.D. Matsakis, and A. Ahmad, “Oxide: The Essence of Rust,” arXiv:1903.00982, 2019.

[10] D.E. Denning, “A Lattice Model of Secure Information Flow,” Commun. ACM, vol.19, no.5, pp.236–243, 1976.

[11] 桑原寛明，國枝義敏，“実行時例外に伴う情報流の型検査に基づく解析手法,” ソフトウェア工学の基礎 XXIII (FOSE 2016), pp.229–234，2016.

[12] J.A. Goguen and J. Meseguer, “Security Policies and Security Models,” 1982 IEEE Symposium on Security and Privacy, pp.11–11, 1982.

[13] E. Reed, “Patina: A Formalization of the Rust Programming Language,” Technical report, UW-CSE-15-03-02, University of Washington, 2015.

[14] S. Benitez, “Short Paper: Rusty Types for Solid Safety,” Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, pp.69–75, ACM, 2016.

[15] M. Lindner, J. Aparicius, and P. Lindgren, “No Panic! Verification of Rust Programs by Symbolic Execution,” 2018 IEEE 16th International Conference on Industrial Informatics, pp.108–114, 2018.

[16] V. Astrauskas, P. Müller, F. Poli, and A.J. Summers, “Leveraging Rust Types for Modular Specification and Verification,” Proc. ACM Program. Lang., vol.3, no.OOPSLA, pp.147:1–147:30, 2019.

[17] M. Baranowski, S. He, and Z. Rakamarić, “Verifying Rust Programs with SMACK,” International Symposium on Automated Technology for Verification and Analysis, pp.528–535, 2018.