# 再帰呼び出しを含む分離論理の部分正当性のための循環証明体系

佐藤 拓海　中澤 巧爾

分離論理は，ホーア論理をヒープ・メモリを操作するプログラムを扱えるよう拡張した体系である．循環証明体系は帰納的な言明を含む論理式のための証明体系で，帰納法を証明の循環構造として表現する．Rowe らは，再帰呼び出しを含むプログラムの停止性検証のために，分離論理の完全正当性のための循環証明体系を提案した．本研究では，再帰呼び出しを含むプログラムに対して，分離論理の部分正当性のための循環証明体系を提案する．この体系において，再帰呼び出しを含むポインタを持つプログラムのメモリ安全性を循環証明体系において証明できることを示し，さらにこの体系の健全性を証明する．

## 1　はじめに

### 1.1　研究背景

**分離論理** (Separation Logic) [10] は，ホーア論理 [6] をヒープ・メモリを操作するプログラムに対して拡張した体系である．事前条件 $A$ が成り立つとき，プログラム $P$ を実行して停止すれば，事後条件 $B$ を満たす状態になっているとき，プログラム $P$ は**部分正当性**を満たすという．ホーア論理は，この性質について事前条件，事後条件をそれぞれ論理式 $A$，$B$ で表したホーア・トリプル

$$\{A\}\ P\ \{B\}$$

を証明する体系である．分離論理ではヒープ中の単一メモリ・セルの状態を表す論理式である**単元ヒープ** $x \mapsto a$(アドレス $x$ に値が $a$ 格納されているような単元ヒープ) と，互いに共通領域を持たないヒープ $A, B$ の結合を表す**分離連言** $A * B$ によりヒープ・メモリの状態を表し，局所的な推論を行う事ができる．論理式の妥当性の判定が決定可能となるように，特に論理式を**シンボリック・ヒープ**と呼ばれる形に制限した分離論理が広く研究されている． [1] [7] [8] [12].

**循環証明体系** [5] は帰納的な言明を含む論理式のための証明体系で，証明の導出木の一部の葉に内部ノードへの循環を許すことにより帰納法を表現する．証明が健全であるためには，**大域トレース条件**と呼ばれる健全性条件を満たす必要がある．

Rowe らは，再帰呼び出しを含む分離論理に対する完全正当性のための循環証明体系を提案した [11]. **完全正当性**とは，部分正当性にプログラムの停止性を加えた性質で，Rowe らが提案した体系は「論理式 $A$ を満たす状態で，プログラム $P$ を実行すると停止して，論理式 $B$ を満たす状態になっている」ことを表すホーア・トリプル

$$[A]\ P\ [B]$$

を証明する体系である．以下のような null 終端の線形リストをアドレス x が指すメモリ・セルから順に走査するプログラムを考える．

```
void TraverseList (Node ∗x){
  if x != NULL {
    y := x → nxt; TraverseList(y)}
```

Rowe らが提案した体系においては，例えば以下のようなトリプルを証明することができる．

$$[\mathsf{List}_\alpha(x)]\ \mathtt{TraverseList(x)}\ [\mathsf{List}_\alpha(x)]$$

この体系においては，帰納的述語にラベルを付与

---

A Cyclic Proof System for Partial Correctness of Separation Logic with Recursive Procedure Calls

Takumi Sato, Koji Nakazawa, 名古屋大学大学院情報学研究科, Graduate School of Informatics, Nagoya University.

し，帰納的述語を展開するごとに順序数で表されるラベルの付値が減少するような制約式を追加する.（ここでいう，述語の展開とは上記の例ではリストの先頭アドレスから始まる単元ヒープを展開することを指す.）これにより，ラベルの付値が無限降下しないことを用いて停止性を証明する.（上記の例では $\alpha$ が述語 $\mathsf{List}(x)$ に付されたラベルを表す.）この体系においては，大域トレース条件は循環するパスごとにラベルの付値が真に減少していることとされている.

### 1.2 Motivating example

Rowe らの体系は，完全正当性を証明するための体系であり，分離論理の部分正当性のための循環証明証明体系は提案されていなかった．例えば，以下のような，線形リストの要素を順に走査するプログラム $P$ を考える.

$$\begin{aligned} P = {}& \texttt{while} \star \texttt{do if x = nil then } \epsilon \\ & \texttt{else x := [x]; od;} \epsilon \end{aligned}$$

ここで，$\texttt{while} \star$ は非決定的な繰り返し命令を表し，$\epsilon$ は skip 文を表す.

以下のようなトリプルを証明することを考える.

$$\{\mathsf{List}(x)\}\ P\ \{\mathsf{List}(x)\}$$

このプログラムは，リストの走査が終了しても分岐命令の if 節が繰り返し実行される可能性があるため，停止しない可能性があるが，停止すればヒープ・メモリでは変わらずリストを表す領域が確保されている．したがって，完全正当性は成立しないが，部分正当性は成立する．ゆえに，Rowe らの体系ではこのトリプルを証明する事ができない.

### 1.3 本研究の成果

本研究では再帰呼び出しを含む分離論理の部分正当性のための循環証明体系を提案し，その健全性を示した．これにより，提案体系において再帰呼び出しを含むポインタを持つプログラムのメモリ安全性を循環証明体系において証明できることが示された.

本提案体系は，Rowe らの体系 [11] をもとに構成し，述語に付するラベルを廃止した．それに伴い，健全性条件を，循環証明中の無限パスがプログラムの実行ステップに対応する規則 (Symbolic Execution Proof Rule と呼ぶ) の適用を無限回含むこととした.

本論文の構成は以下の通りである．まず，第 2 章では提案体系にて扱うプログラムの構文と意味論，表明の構文と意味論について述べる．第 3 章では提案体系における健全性条件を与える．第 4 章では前節で述べた例が実際に提案体系で証明できることを示す．第 5 章では提案体系が健全であることの証明を与える．最後に，6 章では前章までのまとめと今後の展望を述べる.

### 1.4 関連研究

Rowe らは，再帰呼び出しを含むプログラムの停止性検証のために，分離論理の完全正当性のための循環証明体系を提案した [11]．この体系の特徴は上記の説明の通り，述語に対してラベルを付すことである．ラベルは述語を展開する規則を適用するごとにその値が減少する．循環証明における健全性条件は循環するパスごとにラベルの付値が真に減少していることとされている.

Oheimb は相互再帰を含むプログラムに対し，ホーア論理を用いてトリプルの部分正当性を証明するための体系を提案した [13]．この体系では，再帰呼び出しの際，手続き本体の検証の仮定に現在展開されている手続きの仕様を含めることができるようにホーア・トリプルを文脈付きホーア・トリプルへと拡張している．この体系は循環証明ではなく，通常の証明体系である.

## 2 プログラムと分離論理の表明

本章では，Rowe らの再帰呼び出しを含む分離論理の完全正当性のための循環証明体系 [11] において扱う言語を紹介する．プログラムには再帰呼び出しが含まれ，言語はシンボリック・ヒープにより制限された分離論理の表明により表す．また，関数定義中に大域変数は現れないものとする．我々の提案する体系においても同様の言語を扱うが，以下のような差異がある．第一に，簡単のためにヒープ・メモリ中のレコードは単一のフィールドからなるものとし，フィールド名は省略する．第二に，プログラム中で呼び出されている関数は必ず定義が与えられているとし，関数呼び出し

における引数の数は定義されているアリティに等しいとする．第三に，述語はその意味が与えられているものとする．最後に，プログラム実行時の特別な状態 $fault$ はメモリ・エラーに加えて手続き未定義などのエラーも表していたが，本体系においてはメモリ・エラーだけを表すものとする．

**記法についての注意**

列 $\vec{s}$ について，$\vec{s}_i$ は $\vec{s}$ の $i$ 番目の要素を表す．$\epsilon$ は空列を，$\vec{s_1}\cdot\vec{s_2}$ は列の結合を，$|\vec{s}|$ は列 $\vec{s}$ に含まれる要素の数を表す．$\vec{s}$ 中の要素の集合を指して $\vec{s}$ を用いる事がある．また，部分関数 $(\vec{s}\mapsto\vec{t})$ による関数 $f$ の更新を $f[\vec{s}\mapsto\vec{t}]$ と書く．$\vec{s}$ 上の全ての点で $f$ と $f'$ が等しい時，$f =_{\vec{s}} f'$ と書く．

## 2.1 プログラムの構文と意味論

### 2.1.1 プログラムの構文

$x, y, \cdots$ を変数として用いる．変数の集合を $\mathsf{Var}$ とおく．また，プログラムは手続きを持つ列からなる．各手続きは $p(\vec{x})\{C\}$ の形で宣言される．ここで，$p$ は手続き名を，$C$ は手続き本体のコマンドの列であり，$body(p)$ で表される．相異なる変数の列 $\vec{x}$ はパラメータであり，$params(p)$ で表される．また，$locals(p)$ で $p$ の局所変数の集合を表す．本論文で扱う手続きは大域変数を含まないので，$locals(p)$ は $\mathsf{fv}(body(p)) \setminus params(p)$ に等しい．

原子的なコマンドは以下の 6 つからなる．

1. 代入 ($x$:=$E$)
2. 領域からの読み取り ($y$:=$[x]$)
3. 領域への書き込み ($[x]$:=$E$)
4. メモリ領域の割当 ($x$:= `new()`)
5. メモリ領域の解放 (`free`$(x)$)
6. 手続き呼び出し ($p(\vec{E})$)

ここで，式 $E$ は変数か定数 $\mathsf{nil}$ のいずれかを表す．また，コマンドには分岐命令と繰り返し命令がある．

1. 分岐命令 (`if` $B$ `then` $C_1$ `else` $C_2$ `fi`)
2. 繰り返し命令 (`while` $B$ `do` $C$ `od`)

ここで，分岐条件 $B$ は式の等式，またはその否定，もしくは非決定的な条件 ($\star$) により与えられる．$\overline{B}$ は分岐条件の否定を表し，$\overline{\star} \equiv \star$ とする．$mod(C)$ は $C$ 中に $x := \cdots$ が出現するような $x$ の集合を表す．

**定義 2.1** (コマンド)**.** コマンドの列 $C$ を以下のように定義する．

$$
\begin{aligned}
B &::= \star \mid E = E \mid E \neq E\\
C &::= \epsilon \mid x{:=}E; C \mid y{:=}[x]; C \mid [x]{:=}E; C \mid\\
&\qquad x{:=}\ \mathtt{new}(); C \mid \mathtt{free}(x); C \mid p(\vec{E}); C \mid\\
&\qquad \mathtt{if}\ B\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2\ \mathtt{fi}; C \mid\\
&\qquad \mathtt{while}\ B\ \mathtt{do}\ C\ \mathtt{od}; C
\end{aligned}
$$

コマンド $C_1, C_2$ に対し，$C_1\,; C_2$ を以下で定義する．

$$
C_1\,; C_2 = \begin{cases} c\,; C_2 & C_1 = c\ \text{のとき}\\ c\,; (C_1'\,; C_2) & C_1 = c\,; C_1'\text{のとき}\end{cases}
$$

ただし，$c$ は原始的なコマンド，分岐命令，繰り返し命令のいずれかであるとする．

### 2.1.2 プログラムの意味論

$\mathsf{Val}$ を値の集合とする．また，ヒープとして使用可能なロケーションの集合を $\mathsf{Loc}$ とする．ここで，$\mathsf{Loc} \subsetneq \mathsf{Val}$ かつ $nil \in \mathsf{Val}\backslash\mathsf{Loc}$ とする．

**定義 2.2** (ヒープ・モデル)**. ヒープ・モデル**は 2 項組 $(s, h)$ で表される．$s$ は**ストア**，$h$ は**ヒープ**を表す．

**ストア** $s$ は関数 $s : \mathsf{Var} \to \mathsf{Val}$ である．ストア $s$ における式 $E$ の解釈 $[\![E]\!]s$ を $[\![x]\!]s = s(x), [\![\mathsf{nil}]\!]s = nil$ により定義し，さらに式の列 $\vec{E}$ に拡張する．

分岐条件の解釈はストアの集合によって以下のように定義する．任意の $s$ について，$s \in [\![\star]\!]$. $s \in [\![E_1 = E_2]\!] \Leftrightarrow [\![E_1]\!]s = [\![E_2]\!]s$，$s \in [\![E_1 \neq E_2]\!] \Leftrightarrow [\![E_1]\!]s \neq [\![E_2]\!]s$.

**ヒープ** $h$ は有限部分関数 $h : \mathsf{Loc} \to \mathsf{Val}$ である．$\mathsf{emp}$ は定義域が空であるヒープを表す．$\mathrm{dom}(h)$ で $h$ の定義域を表す．また，全てのヒープの集合を $\mathsf{Heaps}$ で表す．

2 つのヒープ $h_1, h_2$ について，$\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \varnothing$ であるとき，それらは**互いに素**であるという．互いに素なヒープ $h_1, h_2$ 間の**ヒープ合成演算** $h_1 \circ h_2$ は以下のように定義される．

$$
(h_1 \circ h_2)(l) = \begin{cases} h_1(l) & l \in \mathrm{dom}(h_1)\ \text{のとき}\\ h_2(l) & l \in \mathrm{dom}(h_2)\ \text{のとき}\\ \text{undefined} & l \notin \mathrm{dom}(h_1) \cup \mathrm{dom}(h_2)\\ & \text{のとき}\end{cases}
$$

$h_1, h_2$ が互いに素でなければ $h_1 \circ h_2$ は定義されない．

**定義 2.3** (操作的意味論)．各手続き呼び出しは**スタック・フレーム** $(C, s)$ 内で実行される．ここで，$C$ は手続き本体の残りの部分，$s$ は手続き内の引数と局所変数の値を記録するスタックを表す．スタック・フレームの列を $\Xi$ で表す．また，$\Xi' < \Xi$ によって，ある $\Xi''$ があって，$\Xi = \Xi'' \cdot \Xi'$ であることを表す．

プログラム実行時の**状態**は以下の $\kappa$ によりモデル化される．$\kappa$ は非空なスタック・フレームの列 $\Xi$ とヒープ $h$ の 2 項組 $(\Xi, h)$，もしくはメモリ・エラーを表す状態 $fault$ で表される．

プログラムの操作的意味論は図 1 の規則によって，状態に対する small-step 関係 $\rightsquigarrow$ により定義される．ここで，$n$ ステップの遷移を $\overset{n}{\rightsquigarrow}$，0 ステップ以上の遷移を $\overset{*}{\rightsquigarrow}$ と書く．また，ある $n$ について状態 $\kappa$ が $\kappa \overset{n}{\rightsquigarrow} ((\epsilon, s), h)$ を満たすとき，$(s, h)$ を**最終状態**といい，$\kappa \overset{n}{\rightsquigarrow} (s, h)$ と書く．

### 2.2 表明の構文と意味論

$P$ を述語の名前の集合 $\mathsf{Pred}$ 上の変数とする．また，$\mathsf{ar}(P)$ で述語 $P$ のアリティを表す．ただし，各述語記号 $P$ に対して，解釈 $[\![P]\!] \subseteq \mathsf{Heaps} \times \mathsf{Val}^{\mathsf{ar}(P)}$ が与えられているものとする．

**定義 2.4** (シンボリック・ヒープ)．**ヒープ論理式** $\Sigma$ と**ストア論理式** $\Pi$ は以下のように定義される．

$$\Pi ::= E = E \mid E \neq E \mid \Pi \wedge \Pi$$

$$\Sigma ::= \bot \mid \top \mid \mathsf{emp} \mid x \mapsto E \mid P(\vec{E}) \mid \Sigma * \Sigma$$

ただし，$P(\vec{E})$ について $|\vec{E}| = \mathsf{ar}(P)$ を満たすものとする．シンボリック・ヒープ $\phi$ は以下のような形をとる論理式である．

$$\phi := \exists \vec{x}.\Pi : \Sigma$$

$\vec{x}$ は相異なる変数の列を表す．$\mathsf{fv}(\phi)$ でシンボリック・ヒープ $\phi$ に現れる自由変数の集合を表す．

シンボリック・ヒープは充足関係 $\models$ によりメモリの状態を記述する．

**定義 2.5** (充足関係)．スタックとヒープ，論理式との間の**充足関係**は、図 2 のように定義される．

**定義 2.6** (エンテイルメント)．シンボリック・ヒープ $\phi, \psi$ について，$\phi \models \psi$ は任意の $(s, h)$ について $(s, h) \models \phi$ ならば $(s, h) \models \psi$ を意味する．

本証明体系では，ホーア・トリプル $\{\phi\}\ C\ \{\psi\}$ を対象としてプログラムの部分正当性を証明する．ここで，$C$ は本章で定義したコマンドである．事前条件 $\phi$，事後条件 $\psi$ はシンボリック・ヒープを用いて記述される表明である．

**定義 2.7** (妥当性)．$(s, h) \models \phi$ を満たす任意の $(s, h)$ について，$((C, s), h) \overset{*}{\rightsquigarrow} fault$ が成立せずかつ，$((C, s), h)$ の任意の最終状態 $(s', h')$ が $(s', h') \models \psi$ を満たすとき，かつそのときに限りホーア・トリプル $\{\phi\}\ C\ \{\psi\}$ は妥当であるという．

## 3 循環証明体系

本章では再帰呼び出しを含む部分正当性のための循環証明体系を定義する．

本体系の推論規則を図 3,4 に示す．

図 3 の *Symbolic Execution Rule* は規則の結論から前提の向きで見たとき，プログラムの先頭コマンドを一つ実行して，事前条件を変更する規則である．図 4 の *Logical Rule* はそれ以外の標準的な分離論理の規則である．

本体系では，循環証明体系を採用しているため，証明の有限導出木において，公理でない葉が内部のノードへの後方参照により閉じている可能性がある．したがって，循環証明体系では，さらに健全性のための条件を課す必要がある [5]．以下ではその条件を形式的に述べる．

**定義 3.1** (循環擬証明)．**循環擬証明** $(\mathcal{P}, F)$ を，公理ではない葉 $\{S_1 \cdots S_n\}$ を持つ導出木 $\mathcal{P}$ と $S_i$ $(1 \leqslant i \leqslant n)$ と同じトリプルをラベルに持つ内部ノードを割り当てる写像 $F$ の組とする．また，公理でない葉の集合を $\mathsf{Bud}$ とおく．

参照されたノードを識別することで，循環擬証明は無限の導出木を循環グラフで表現したものとみなすことができる．循環擬証明 $(\mathcal{P}, F)$ のグラフを $\mathcal{G}_{\mathcal{P}}$ と表す．$\mathcal{G}_{\mathcal{P}}$ はノードとして循環擬証明 $(\mathcal{P}, F)$ に現れる各ホーア・トリプル $S$ と $S$ を結論とする推論規則の名前 $r$ の組 $(S, r)$ をもつ．$\mathcal{G}_{\mathcal{P}}$ は辺として以下の 2 種類の有向辺を持つ．

$S \in \mathsf{Bud}$ のとき：$(S, r)$ から $(F(S), r')$ への辺

$S \notin \mathsf{Bud}$ のとき：$(S, r)$ から $(S', r')$ への辺

ただし，$S'$ は $S$ を結論とする $r$ の仮定とする

$$\frac{}{((x{:=}E; C, s)\cdot\Xi, h) \rightsquigarrow ((C, s[x \mapsto [\![E]\!]s])\cdot\Xi, h)}$$

$$\frac{[\![x]\!]s \in dom(h)}{((y{:=}[x]; C, s)\cdot\Xi, h) \rightsquigarrow ((C, s[y \mapsto h([\![x]\!]s)])\cdot\Xi, h)} \qquad \frac{[\![x]\!]s \in dom(h)}{(([x]{:=}E; C, s)\cdot\Xi, h) \rightsquigarrow ((C, s)\cdot\Xi, h[[\![x]\!]s \mapsto [\![E]\!]s])}$$

$$\frac{l \notin dom(h) \quad v \subset \mathsf{Val}}{((x{:=}\,\texttt{new}(); C, s)\cdot\Xi, h) \rightsquigarrow ((C, s[x \mapsto l])\cdot\Xi, h[l \mapsto v])}$$

$$\frac{[\![x]\!]s \in dom(h)}{((\texttt{free}(x); C, s)\cdot\Xi, h) \rightsquigarrow ((C, s)\cdot\Xi, h \upharpoonright dom(h) \setminus \{[\![x]\!]s\})}$$

$$\frac{s \in [\![B]\!]}{((\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C, s)\cdot\Xi, h) \rightsquigarrow ((C_1; C, s)\cdot\Xi, h)}$$

$$\frac{s \in [\![\overline{B}]\!]}{((\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C, s)\cdot\Xi, h) \rightsquigarrow ((C_2; C, s)\cdot\Xi, h)}$$

$$\frac{s \in [\![B]\!]}{((\texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}; C', s)\cdot\Xi, h) \rightsquigarrow ((C; \texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}; C', s)\cdot\Xi, h)}$$

$$\frac{s \in [\![\overline{B}]\!]}{((\texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}; C', s)\cdot\Xi, h) \rightsquigarrow ((C', s)\cdot\Xi, h)}$$

$$\frac{body(p) = C' \quad params(p) = \vec{x} \quad |\vec{E}| = |\vec{x}| \quad s' \upharpoonright \vec{x} = (\vec{x} \mapsto [\![\vec{E}]\!]s)}{((p(\vec{E}); C, s)\cdot\Xi, h) \rightsquigarrow ((C', s')\cdot(C, s)\cdot\Xi, h)}$$

$$\frac{\Xi \neq \epsilon}{((\epsilon, s)\cdot\Xi, h) \rightsquigarrow (\Xi, h)} \qquad \frac{[\![x]\!]s \notin dom(h)}{((y{:=}[x]; C, s)\cdot\Xi, h) \rightsquigarrow fault} \qquad \frac{[\![x]\!]s \notin dom(h)}{(([x]{:=}E; C, s)\cdot\Xi, h) \rightsquigarrow fault}$$

$$\frac{[\![x]\!]s \notin dom(h)}{((\texttt{free}(x); C, s)\cdot\Xi, h) \rightsquigarrow fault}$$

**図 1　操作的意味論**

$$\begin{array}{rcl}
(s,h) \models E_1 = E_2 & \Leftrightarrow & [\![E_1]\!]s = [\![E_2]\!]s \\
(s,h) \models E_1 \neq E_2 & \Leftrightarrow & [\![E_1]\!]s \neq [\![E_2]\!]s \\
(s,h) \models \Pi_1 \wedge \Pi_2 & \Leftrightarrow & (s,h) \models \Pi_1 \text{ かつ } (s,h) \models \Pi_2 \\
(s,h) \models \bot & \Leftrightarrow & \mathsf{false} \\
(s,h) \models \top & \Leftrightarrow & \mathsf{true} \\
(s,h) \models \texttt{emp} & \Leftrightarrow & \mathsf{dom}(h) = \varnothing \\
(s,h) \models x \mapsto E & \Leftrightarrow & \mathsf{dom}(h) = \{[\![x]\!]s\} \text{ かつ } h([\![x]\!]s) = [\![E]\!]s \\
(s,h) \models P(\vec{E}) & \Leftrightarrow & (h, [\![\vec{E}]\!]s) \in [\![P]\!] \\
(s,h) \models \Sigma_1 * \Sigma_2 & \Leftrightarrow & \exists h_1, h_2. h = h_1 \circ h_2 \text{ かつ } (s,h_1) \models \Sigma_1 \text{ かつ } (s,h_2) \models \Sigma_2 \\
(s,h) \models \Pi : \Sigma & \Leftrightarrow & (s,h) \models \Pi \text{ かつ } (s,h) \models \Sigma \\
(s,h) \models \exists x.\phi & \Leftrightarrow & \exists v \in \mathsf{Val}.(s[x \mapsto v], h) \models \phi
\end{array}$$

**図 2　充足関係**

**定義 3.2** (progress point)**.** $\mathcal{P}$ が循環擬証明であるとし，$\mathcal{G}_{\mathcal{P}}$ におけるパスを $\boldsymbol{\nu} = (S_1, r_1), (S_2, r_2), \ldots$ とする．$r_i$ が Symbolic Execution Rule であるとき，$i$ は ***progress point*** であるという．

循環擬証明は以下の健全性条件を満たすときに限り，妥当な証明であるとみなされる．

(STOP)
$$\frac{}{\vdash \{\phi\}\ \epsilon\ \{\psi\}}(\phi \models \psi)$$

(WRITE)
$$\frac{\vdash \{\Pi : x \mapsto E' * \Sigma\}\ C\ \{\psi\}}{\vdash \{\Pi : x \mapsto E * \Sigma\}\ [x] := E'; C\ \{\psi\}}$$

(ASSIGN)
$$\frac{\vdash \{\Pi[x'/x] \wedge x = E[x'/x] : \Sigma[x'/x]\}\ C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ x := E; C\ \{\psi\}}(x' \text{ はフレッシュな変数 })$$

(READ)
$$\frac{\vdash \{\Pi[x'/x] \wedge x = E[x'/x] : (y \mapsto E * \Sigma)[x'/x]\}\ C\ \{\psi\}}{\vdash \{\Pi : y \mapsto E * \Sigma\}\ x := [y]; C\ \{\psi\}}(x' \text{ はフレッシュな変数 })$$

(FREE)
$$\frac{\vdash \{\Pi : \Sigma\}\ C\ \{\psi\}}{\vdash \{\Pi : x \mapsto E * \Sigma\}\ \texttt{free}(x); C\ \{\psi\}}$$

(NEW)
$$\frac{\vdash \{\Pi[x'/x] : x \mapsto y * \Sigma[x'/x]\}\ C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ x := \texttt{new()}\ ; C\ \{\psi\}}(x', y \text{ はフレッシュな変数 })$$

(IF-NONDET)
$$\frac{\vdash \{\phi\}\ C_1; C\ \{\psi\} \qquad \vdash \{\phi\}\ C_2; C\ \{\psi\}}{\vdash \{\phi\}\ \texttt{if} \star \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C\ \{\psi\}}$$

(IF-DET)
$$\frac{\vdash \{\Pi \wedge B : \Sigma\}\ C_1; C\ \{\psi\} \qquad \vdash \{\Pi \wedge \bar{B} : \Sigma\}\ C_2; C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C\ \{\psi\}}$$

(WHILE-NONDET)
$$\frac{\vdash \{\phi\}\ C'; \texttt{while} \star \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\} \qquad \vdash \{\phi\}\ C\ \{\psi\}}{\vdash \{\phi\}\ \texttt{while} \star \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\}}$$

(WHILE-DET)
$$\frac{\vdash \{\Pi \wedge B : \Sigma\}\ C'; \texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\} \qquad \vdash \{\Pi \wedge \bar{B} : \Sigma\}\ C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ \texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\}}$$

(PROC)
$$\frac{\vdash \{\phi\}\ body(p)\ \{\psi\}}{\vdash \{\phi\}\ p(\vec{x})\{\psi\}}(\vec{x} = params(p), locals(p) \cap (\mathsf{fv}(\phi) \cup \mathsf{fv}(\psi)) = \emptyset)$$

**図 3　Symbolic Execution Proof Rules**

(BOT)
$$\frac{}{\vdash \{\bot\}\ C\ \{\psi\}}$$

(CONSEQUENCE)
$$\frac{\vdash \{\chi\}\ C\ \{\xi\}}{\vdash \{\phi\}\ C\ \{\psi\}}\quad(\phi \models \chi, \xi \models \psi)$$

(FRAME)
$$\frac{\vdash \{\phi\}\ C\ \{\psi\}}{\vdash \{\phi * \xi\}\ C\ \{\psi * \xi\}}(\mathsf{fv}(\xi) \cap mod(C) = \emptyset)$$

(SUBST)
$$\frac{\vdash \{\phi\}\ C\ \{\psi\}}{\vdash \{\phi[E/x]\}\ C\ \{\psi[E/x]\}}(x \notin vars(C), x \in \mathsf{fv}(\psi) \Rightarrow \mathsf{fv}(E) \cap \mathsf{fv}(C) = \emptyset)$$

(PARAM)
$$\frac{\vdash \{\phi\}\ p(\vec{E})\ \{\psi\}}{\vdash \{\phi[E/x]\}\ p(\vec{E}[E/x])\ \{\psi[E/x]\}}$$

(SEQ)
$$\frac{\vdash \{\phi\}\ C_1\ \{\chi\} \qquad \vdash \{\chi\}\ C_2\ \{\psi\}}{\vdash \{\phi\}\ C_1; C_2\ \{\psi\}}$$

($\exists$VAR)
$$\frac{\vdash \{\phi[y/x]\}\ C\ \{\psi\}}{\vdash \{\exists x.\phi\}\ C\ \{\psi\}}(y \text{ はフレッシュな変数 })$$

**図 4　Logical Proof Rules**

**定義 3.3** (循環証明)．循環擬証明 $\mathcal{P}$ について，$\mathcal{G}_{\mathcal{P}}$ における全ての無限パス $\boldsymbol{\nu}$ が無限の progress point をもつとき，$\mathcal{P}$ は**循環証明**であるという．

健全性条件は，大域トレース条件 [5] と呼ばれる条件に対応する．この条件は Büchi オートマトンの包含問題に帰着することで決定可能となることが知られている [4]．

## 4　提案体系における証明例

本章では 1 章で紹介した例を本提案体系で証明で

きることを示す．

**例 1.**

以下はリストを先頭の要素から順に走査していくが，各ループの段階で非決定的に skip 文 ($\epsilon$ で表される) を実行するプログラムである．

$$\mathtt{while} \star \mathtt{do\ if\ x\ =\ nil\ then}\ \epsilon$$
$$\mathtt{else\ x := [x];\ od;}\ \epsilon$$

以下のトリプルを考える．

$\{\mathsf{List}(x)\}$
  $\mathtt{while} \star \mathtt{do\ if\ x\ =\ nil\ then}\ \epsilon$
  $\mathtt{else\ x := [x];\ od;}\ \epsilon$
$\{\mathsf{List}(x)\}$

このプログラムは，リストの走査が終了しても分岐命令の if 節が繰り返し実行される可能性があり，停止しない可能性があるため，Rowe らの体系では証明できない．しかし，部分正当性の意味では妥当なトリプルである．実際，我々の体系において図 5 のように証明できる．(ここで，図中の (†) や (∗) が付されたトリプルは Bud に属する葉とそれに対応する内部ノードを示す. また，図中の $C$ は分岐命令の全体を，$C'$ は分岐命令の else 節を指す．) また，図 5 の証明は健全性条件を満たしている．

**例 2.**

相互再帰を含む手続き f, g, h を以下の通り定義する．

```
void f(int x, int y, int z){
  if x = 0 then g(x, y, z);
  else h(x, y, z); }

void g(int x, int y, int z){
  if y = 0 then [z] := 0;
  else y := y − 1; h(x, y, z); }

void h(int x, int y, int z){
  if x = 0 then y := y − 1; g(x, y, z);
  else x := x − 1; f(x, y, z); }
```

以下のトリプルを考える．

$$\{\top\}\ \mathtt{z := \ new();\ f(x,y,z)}\{\mathtt{z} \mapsto 0\}$$

このプログラムは，$x < 0$ の場合停止しない可能性があるが，部分正当性の意味で妥当なトリプルであり，図 6 のように証明できる．(例 1 と同様に図中の (†) や (∗) が付されたトリプルは Bud に属する葉とそれに対応する内部ノードを示す．) また，図 6 の証明は健全性条件を満たしている．

## 5 健全性

本章では提案する体系が健全であることを示す．

準備として，非負整数 $n$ について以下のような性質 $n$-invalid を定義する．

**定義 5.1** ($n$-invalid)**.** $\{\phi\}C\{\psi\}$ が *$n$-invalid* であるとは，あるモデル $(s, h)$ に対して，$((C, s), h) \stackrel{n}{\rightsquigarrow} ((\epsilon, s'), h')$ なる最終状態 $(s', h')$ が存在し，$(s, h) \models \phi$ かつ $(s', h') \not\models \psi$ を満たす，もしくは $((C, s), h) \stackrel{n}{\rightsquigarrow} fault$ を満たす事をいう．

**補題 5.1.** $\{\phi\}C\{\psi\}$ が妥当でないことと $\{\phi\}C\{\psi\}$ がある n について $n$-invalid であることは同値である．

*Proof.* 定義 5.1 より明らかである． □

健全性の証明のために，まず以下の命題を示す．

**命題 5.1** (局所健全性)**.** 各推論規則 $r$ の適用について $S$ を $r$ の結論とし，$S$ が $n$-invalid であると仮定する．このとき，$m \leqslant n$ なるある $m$ に対して，$m$-invalid であるような $r$ の前提 $S'$ が存在する．特に，$r$ が Symbolic Execution Rule であるとき，$m < n$ なる $m$ に対して $m$-invalid であるような $r$ の前提 $S'$ が存在する．

証明は付録 A に記載する．

以上より以下を得る．

**定理 5.1** (健全性)**.** $\vdash \{\phi\}\ C\ \{\psi\}$ が循環証明を持つならば，$\{\phi\}\ C\ \{\psi\}$ は妥当な表明である．

*Proof.* $\vdash \{\phi\}\ C\ \{\psi\}$ が循環証明 $\mathcal{P}$ を持ち，$\{\phi\}\ C\ \{\psi\}$ が妥当でないと仮定する．補題 5.1 より，$\{\phi\}\ C\ \{\psi\}$ はある $n$ に対して $n$-invalid である．循環証明 $\mathcal{P}$ が存在することから，命題 5.1 より，$\{\phi\}\ C\ \{\psi\}$ から始まる妥当でない表明のパス $\boldsymbol{\nu}$ が存在する．$\boldsymbol{\nu}$ が有限

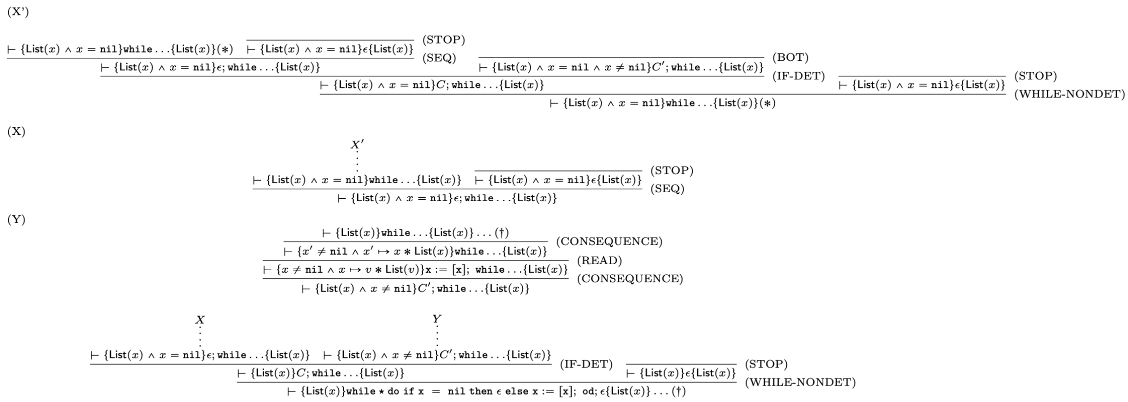

**図 5　部分正当性のみ満たす例**

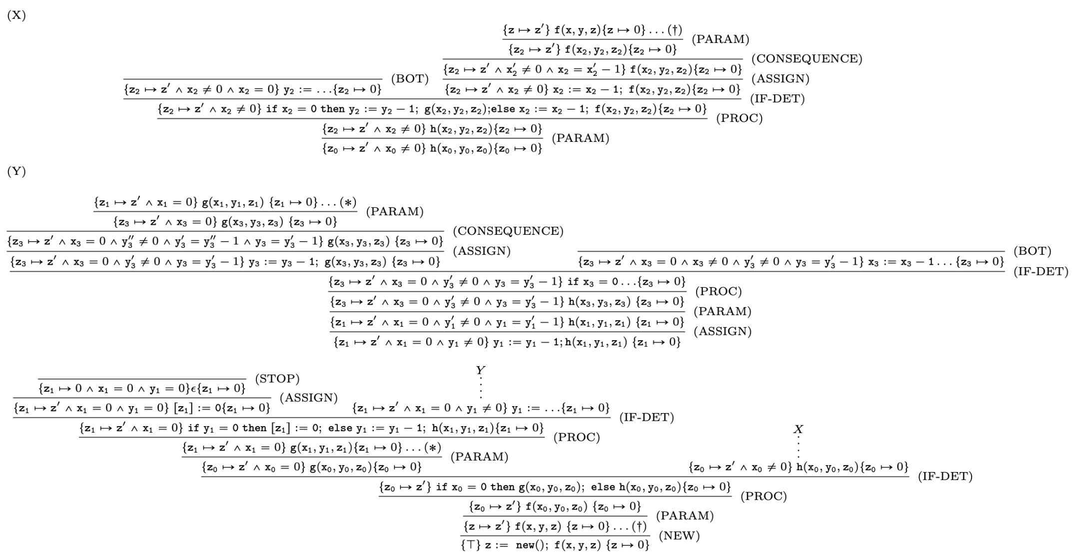

**図 6　相互再帰を含む例**

であるとすると，axiom でパス $\boldsymbol{\nu}$ が止まっているが，これは命題 5.1 に矛盾する．ゆえに $\boldsymbol{\nu}$ は無限パスである．

$\mathcal{P}$ は妥当な循環証明であるから，大域トレース条件より，無限パス $\boldsymbol{\nu}$ において無限回の Symbolic Execution Proof Rule を適用していなければならない．ここで，命題 5.1 より，無限パス $\boldsymbol{\nu}$ における $n$ の無限降下が生じ，矛盾．よって $\{\phi\}\ C\ \{\psi\}$ は妥当な表明である．

□

## 6　まとめと今後の課題

本論文では，分離論理の部分正当性のための循環証明体系を提案し，その健全性を示した．体系の構築にあたって，健全性条件のための progress point をプログラムの先頭のコマンドを実行するような推論規則 (Symbolic Execution Proof Rule) を適用する箇所と定義した．健全性，局所健全性の証明にあたって，実行系列のステップ数が $n$ であるような妥当でないホーア・トリプルを表す $n$-invalid という性質を

用いた．我々の提案体系により，Rowe らの体系では証明できない，停止しない可能性があるが部分正当性の意味で妥当なホーア・トリプルを証明することができるようになる．

並行分離論理[2][9]は分離論理を並行プログラムに対して拡張した体系である．並行分離論理では複数のスレッドを並行に動作させた際の仕様を証明することができる．Brotherston の体系 $SL_{LP}$ [3] では，分割可能な権限値 (fractional permission) と弱分離連言によって，あるスレッドが共有メモリの特定の領域に対して持つアクセス権限をより詳細に表現する．$SL_{LP}$ ではこれらの機構に加えてラベルによって，権限値の分割と合併を表現する．今後の課題としては，我々の提案する分離論理の部分正当性のための循環証明体系に権限値とラベルを導入することによって，並行分離論理を扱えるよう拡張し，再帰呼び出しを含むポインタをもつ並行プログラムのメモリ安全性を循環証明体系において証明できることを示すことが挙げられる．

## 参 考 文 献


[1] Berdine, J., Calcagno, C., and O'Hearn, P.: Symbolic execution with separation logic, *Programming Languages and Systems (APLAS 2005)*, Lecture Notes in Computer Science, Vol. 3780, Springer, 2005, pp. 52–68.

[2] Brookes, S.: A semantics for concurrent separation logic, *Theoretical Computer Science*, Vol. 375, No. 1-3(2007), pp. 227–270.

[3] Brotherston, J., Costa, D., Hobor, A., and Wickerson, J.: Reasoning over Permissions Regions in Concurrent Separation Logic, *International Conference on Computer Aided Verification*, Springer, 2020, pp. 203–224.

[4] Brotherston, J., Gorogiannis, N., and Petersen, R. L.: A generic cyclic theorem prover, *Asian Symposium on Programming Languages and Systems*, Springer, 2012, pp. 350–367.

[5] Brotherston, J. and Simpson, A.: Sequent calculi for induction and infinite descent, *Journal of Logic and Computation*, Vol. 21, No. 6(2011), pp. 1177–1216.

[6] Hoare, C. A. R.: An axiomatic basis for computer programming, *Communications of the ACM*, Vol. 12, No. 10(1969), pp. 576–580.

[7] Iosif, R., Rogalewicz, A., and Simacek, J.: The tree width of separation logic with recursive definitions, *International Conference on Automated Deduction*, Springer, 2013, pp. 21–38.

[8] Katelaan, J., Matheja, C., and Zuleger, F.: Effective entailment checking for separation logic with inductive definitions, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2019)*, Lecture Notes in Computer Science, Vol. 11428, Springer, 2019, pp. 319–336.

[9] O'Hearn, P. W.: Resources, concurrency, and local reasoning, *Theoretical computer science*, Vol. 375, No. 1(2007), pp. 271–307.

[10] Reynolds, J. C.: Separation logic: A logic for shared mutable data structures, *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, IEEE, 2002, pp. 55–74.

[11] Rowe, R. N. and Brotherston, J.: Automatic cyclic termination proofs for recursive procedures in separation logic, *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, 2017, pp. 53–65.

[12] Tatsuta, M., Nakazawa, K., and Kimura, D.: Completeness of cyclic proofs for symbolic heaps with inductive definitions, *Asian Symposium on Programming Languages and Systems*, Springer, 2019, pp. 367–387.

[13] Von Oheimb, D.: Hoare logic for mutual recursion and local variables, *Foundations of Software Technology and Theoretical Computer Science: 19th Conference Chennai, India, December 13-15, 1999 Proceedings 19*, Springer, 1999, pp. 168–180.

## A　命題 5.1 の証明

以下では，命題 5.1 の証明を示す．

まず，命題 5.1 の証明のために以下の補題を用意する．

**補題 A.1.** $((C, s), h) \overset{n}{\rightsquigarrow} (s', h')$ かつ $x \notin \mathsf{fv}(C)$ ならば，$((C, s[x \mapsto v]), h) \overset{n}{\rightsquigarrow} (s'[x \mapsto v], h')$.

*Proof.* $n$ に関する帰納法による． □

**補題 A.2.** $s =_{locals(p) \,\cup\, params(p)} s'$ かつ $((body(p), s), h) \overset{n}{\rightsquigarrow} (s_1, h_1)$ ならば，$s_1$ に対して $s_1 =_{locals(p) \,\cup\, params(p)} s'_1$ であるような $s'_1$ が存在して，$((body(p), s'), h) \overset{n}{\rightsquigarrow} (s'_1, h_1)$ を満たす．

*Proof.* 補題 A.1 と，$\mathsf{fv}(body(p)) \subseteq locals(p) \cup params(p)$ より． □

**補題 A.3.** $\Xi' < \Xi$ であるようなスタック・フレームの列 $\Xi$，$\Xi'$ とヒープ $h$ について $(\Xi, h) \overset{n}{\rightsquigarrow} ((\epsilon, s'), h')$ ならば，$m_1 + m_2 = n$ なる $m_1$，$m_2$ と $h''$ が存在し，$(\Xi, h) \overset{m_1}{\rightsquigarrow} (\Xi', h'') \overset{m_2}{\rightsquigarrow} ((\epsilon, s'), h')$ を満たす．ただし，$\Xi'$ は非空であるとする．

*Proof.* $\Xi = (C, s) \cdot \Xi'$ の場合を示せば十分である．$n$ に関する帰納法による． □

**補題 A.4.** $((C_1; C_2, s), h) \overset{n}{\rightsquigarrow} (s', h')$ ならば，$((C_1, s), h) \overset{m}{\rightsquigarrow} (s'', h'')$ なる $(s'', h'')$ と $m \leqslant n$ が存在して，$((C_1; C_2, s), h) \overset{m}{\rightsquigarrow} ((C_2, s''), h'') \overset{n-m}{\rightsquigarrow} (s', h')$

*Proof.* 実行系列 $((C_1; C_2, s), h) \overset{n}{\rightsquigarrow} (s', h')$ のステップ数 $n$ に関する帰納法により示す．

(基底部分)　$n = 0$ の場合．$s'', h''$ として $s', h'$ をとる．$n = 0$ であるから，定義 2.3 より，$((C_1; C_2, s), h) \overset{0}{\rightsquigarrow} ((C_2, s'), h') \overset{0}{\rightsquigarrow} (s', h')$ が成り立つ．

(帰納部分)　$n > 0$ の場合．$((C_1; C_2, s), h) \overset{n}{\rightsquigarrow} (s', h')$ より，実行系列を $((C_1; C_2, s), h) \overset{1}{\rightsquigarrow} (\Xi, h_1) \overset{n-1}{\rightsquigarrow} (s', h')$ の形であるとする．

Case1. $C_1 = p(x); C'_1$ の場合．この時，定義 2.3 より実行系列の最初のステップは $((p(x); C'_1; C_2, s), h) \rightsquigarrow ((body(p), \hat{s}) \cdot (C'_1; C_2, s), h)$ の形を取る．(このとき $\Xi = (body(p), \hat{s}) \cdot (C'_1; C_2, s))$ ここで，補題 A.3 より，$((\hat{C}_1, \hat{s}) \cdot (C'_1; C_2, s), h) \rightsquigarrow ((C'_1; C_2, s), \hat{h}) \overset{m}{\rightsquigarrow} (s', h')$ なる $\hat{h}$ が存在する．ここで，$m \leqslant n - 1 < n$ より帰納法の仮定から $((C'_1; C_2, s), \hat{h}) \overset{m'}{\rightsquigarrow} ((C_2, \hat{s''}), \hat{h''})$ なる $\hat{s''}, \hat{h''}$ と $m' \leqslant m$ が存在し，$((C'_1; C_2, s), \hat{h}) \overset{m'}{\rightsquigarrow} ((C_2, \hat{s''}), \hat{h''}) \overset{m-m'}{\rightsquigarrow} (s', h')$ を満たす．よって，$s'' = \hat{s''}, h'' = \hat{h''}$ とすれば，題意を満たす．

Case2. その他の場合．この時，$C_1 = c; C'_1$ とおくと，定義 2.3 より実行系列の最初のステップは $((c; C'_1; C_2, s), h) \rightsquigarrow ((C''_1; C'_1; C_2, s'_1), h'_1)$ の形を取る．今，$((c; C'_1; C_2, s), h) \rightsquigarrow ((C''_1; C'_1; C_2, s'_1), h'_1) \overset{n-1}{\rightsquigarrow} (s', h')$ が成り立っている．ここで，帰納法の仮定より，$((C''_1; C'_1; C_2, s'_1), h'_1) \overset{m}{\rightsquigarrow} ((C_2, \hat{s''}), \hat{h''})$ なる $\hat{s''}, \hat{h''}$ と $m \leqslant n - 1 < n$ が存在し，$((C''_1; C'_1; C_2, s'_1), h'_1) \overset{m}{\rightsquigarrow} ((C_2, \hat{s''}), \hat{h''}) \overset{n-1-m}{\rightsquigarrow} (s', h')$ を満たす．よって，$s'' = \hat{s''}, h'' = \hat{h''}$ とすれば，題意を満たす．

□

**補題 A.5.** $((C_1; C_2, s), h) \overset{n}{\rightsquigarrow} fault$ ならば以下のいずれかが成立する．

*Case 1.* $((C_1, s), h) \overset{n}{\rightsquigarrow} fault$

*Case 2.* $((C_1, s), h) \overset{m}{\rightsquigarrow} (s'', h'')$ なる $(s'', h'')$ と $m \leqslant n$ が存在して，$((C_1; C_2, s), h) \overset{m}{\rightsquigarrow} ((C_2, s''), h'') \overset{n-m}{\rightsquigarrow} fault$

*Proof.* 補題 A.4 と同様． □

**補題 A.6.** *1.* $(\Xi, h) \overset{n}{\rightsquigarrow} (s', h')$，$h = h_1 \circ h_2$ ならば以下のいずれかが成立する．

*Case 1.* $h' = h'_1 \circ h_2$ なる $h'_1$ があって，$(\Xi, h_1) \overset{n}{\rightsquigarrow} (s', h'_1)$.

*Case 2.* ある $m \leqslant n$ があって，$(\Xi, h_1) \overset{m}{\rightsquigarrow} fault$.

*2.* $(\Xi, h) \overset{n}{\rightsquigarrow} fault$，$h = h_1 \circ h_2$ ならば，ある

$m \leqslant n$ があって，$(\Xi, h_1) \overset{m}{\rightsquigarrow} fault$.

*Proof.* 1. $n$ に関する帰納法による．

$n = 0$ のとき，すなわち $\Xi = (\epsilon, s')$ かつ $h = h'$ のとき，$h'_1 = h_1$ とすれば Case 1 が成立する．

$n > 0$ のとき，実行系列の最初のステップの形で場合分けをする．$y \coloneqq [x]$，$[x] \coloneqq E$，$\mathtt{free}(x)$ 以外の場合は帰納法の仮定から簡単に示せる．

$\Xi = (y \coloneqq [x]; C', s) \cdot \Xi'$ のとき，最初のステップは，$((y \coloneqq [x]; C', s) \cdot \Xi', h) \rightsquigarrow ((C', s[y \mapsto h(\llbracket x \rrbracket s)]) \cdot \Xi', h) \overset{n-1}{\rightsquigarrow} (s', h')$ の形である．

$\llbracket x \rrbracket s \in dom(h_1)$ のとき，$((y \coloneqq [x]; C', s) \cdot \Xi', h_1) \rightsquigarrow ((C', s[y \mapsto h_1(\llbracket x \rrbracket s)]) \cdot \Xi', h_1)$ であり，帰納法の仮定より，(Case 1.) $((C', s[y \mapsto h_1(\llbracket x \rrbracket s)]) \cdot \Xi', h_1) \overset{n-1}{\rightsquigarrow} (s', h'_1)$ かつ $h' = h'_1 \circ h_2$ なる $h'_1$ が存在するか，(Case 2.) ある $m \leqslant n-1$ があって $((C', s[y \mapsto h_1(\llbracket x \rrbracket s)]) \cdot \Xi', h_1) \overset{m}{\rightsquigarrow} fault$.

$[x] \coloneqq E$，$\mathtt{free}(x)$ の場合も同様である．

2. $n$ に関する帰納法による． □

以下では命題 *5.1* を適用される規則ごとに証明する．

*Proof.* （命題 5.1 の証明）

**Symbolic Execution Proof Rules**

**(STOP)**

$$\frac{}{\vdash \{\phi\}\ \epsilon\ \{\psi\}}\ (\phi \models \psi)$$

$\phi \models \psi$ が成り立つから，$\{\phi\}\ \epsilon\ \{\psi\}$ は妥当である．

**(WRITE)**

$$\frac{\vdash \{\Pi : x \mapsto E' * \Sigma\}\ C\ \{\psi\}}{\vdash \{\Pi : x \mapsto E * \Sigma\}\ [x] \coloneqq E'; C\ \{\psi\}}$$

結論 $\{\Pi : x \mapsto E * \Sigma\}\ C\ \{\psi\}$ が $n$-invalid であるとすると，$(s, h) \models \Pi : x \mapsto E * \Sigma$ なる $s, h$ が存在し，$(([x] \coloneqq E'; C, s), h) \overset{n}{\rightsquigarrow} (s', h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} fault$ である．

ここで，$h'' = h[\llbracket x \rrbracket s \mapsto \llbracket E' \rrbracket s]$ とおく．

まず，$(s, h'') \models \Pi : x \mapsto E' * \Sigma$ であることを示す．

定義 2.5 より，

$(s, h) \models \Pi : x \mapsto E * \Sigma$

$\Leftrightarrow (s, h) \models \Pi$ かつ，ある $h_1, h_2$ があって

$h = h_1 \circ h_2$ かつ $(s, h_1) \models x \mapsto E$

かつ $(s, h_2) \models \Sigma$

$\Leftrightarrow (s, h) \models \Pi$ かつ，ある $h_1, h_2$ があって

$dom(h_1) = \{\llbracket x \rrbracket s\}$ かつ $h_1(\llbracket x \rrbracket s) = \llbracket E \rrbracket s$

かつ $(s, h_2) \models \Sigma$

ここで，$h'' = h[\llbracket x \rrbracket s \mapsto \llbracket E' \rrbracket s]$ かつ $\llbracket x \rrbracket s \in dom(h_1)$ であるから，ある $h'_1 \subseteq h$ が存在して，

$h'' = h'_1 \circ h_2$ *s.t.*

$dom(h'_1) = \{\llbracket x \rrbracket s\}$ かつ $h'_1(\llbracket x \rrbracket s) = \llbracket E' \rrbracket s$

かつ $(s, h_2) \models \Sigma$

が成り立つ．ゆえに

$(s, h'') \models \Pi$ かつ，ある $h_1, h_2$ があって

$h'' = h'_1 \circ h_2$ かつ $(s, h'_1) \models x \mapsto E'$

かつ $(s, h_2) \models \Sigma$

$\Leftrightarrow (s, h'') \models \Pi : x \mapsto E' * \Sigma$

さらに，$\llbracket x \rrbracket s \in dom(h)$ より，定義 2.3 から，実行系列において $(([x] \coloneqq E; C, s), h)(= \kappa_0) \rightsquigarrow \kappa_1$ となる $\kappa_1$ は $((C, s), h'')$ のみである．よって，$\kappa_0 \rightsquigarrow ((C, s), h'') \overset{n-1}{\rightsquigarrow} (s', h') \not\models \psi$ または $\overset{n-1}{\rightsquigarrow} fault$ である．よって，規則の仮定 $\{\Pi : x \mapsto E' * \Sigma\}\ C\ \{\psi\}$ は $(n-1)$-invalid である．

**(ASSIGN)**

$$\frac{\vdash \{\Pi[x'/x] \wedge x = E[x'/x] : \Sigma[x'/x]\}\ C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ x \coloneqq E; C\ \{\psi\}}$$

($x'$ はフレッシュな変数 )

規則の結論 $\{\Pi : \Sigma\}\ x \coloneqq E; C\ \{\psi\}$ が $n$-invalid であるとすると，$(s, h) \models \Pi : \Sigma$ なる $s, h$ が存在し，$((x \coloneqq E; C, s), h) \overset{n}{\rightsquigarrow} (s', h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} fault$ である．

ここで，$s'' = s[x' \mapsto \llbracket x \rrbracket s][x \mapsto \llbracket E \rrbracket s]$ とおく．

まず，$(s'', h) \models \Pi[x'/x] \wedge x = E[x'/x] : \Sigma[x'/x]$

であることを示す．

$(s,h) \models \Pi : \Sigma$

$\Leftrightarrow (s[x' \mapsto [\![x]\!]s], h) \models \Pi[x'/x] : \Sigma[x'/x]$

($\because x'$ はフレッシュ)

$\Leftrightarrow (s[x' \mapsto [\![x]\!]s][x \mapsto [\![E]\!]s], h) \models \Pi[x'/x] : \Sigma[x'/x]$

($\because x \notin \mathsf{fv}(\Pi[x'/x] : \Sigma[x'/x])$)

ここで，$s'' = s[x' \mapsto [\![x]\!]s][x \mapsto [\![E]\!]s]$ なので，$[\![x]\!]s'' = [\![E]\!]s$ と $[\![E[x'/x]]\!]s'' = [\![E]\!]s$ より，$[\![x]\!]s'' = [\![E[x'/x]]\!]s''$．ゆえに

$$(s'',h) \models \Pi[x'/x] \wedge x = E[x'/x] : \Sigma[x'/x]$$

一方，実行系列は定義 2.3 より $((x \coloneqq E; C, s), h) \rightsquigarrow ((C, s[x \mapsto [\![E]\!]s]), h) \overset{n-1}{\rightsquigarrow} \kappa (= (s',h')$ または $fault)$ の形に限られる．$x'$ はフレッシュなので，補題 A.1 より，$((C, s[x \mapsto [\![E]\!]s][x' \mapsto [\![x]\!]s]), h) \overset{n-1}{\rightsquigarrow} \kappa' (= (s'[x' \mapsto [\![x]\!]s], h')$ または $fault)$ が成り立つ．$x \neq x'$ より $s[x \mapsto [\![E]\!]s][x' \mapsto [\![x]\!]s] = s''$ であり，さらに，$(s',h') \not\models \psi$ と $x' \notin \mathsf{fv}(\psi)$ より $(s'[x' \mapsto [\![x]\!]s], h') \not\models \psi$ であるから，規則の仮定 $\{\Pi[x'/x] \wedge x = E[x'/x] : \Sigma[x'/x]\}\ C\ \{\psi\}$ は $(n-1)$-invalid である．

**(READ)**

$$\frac{\vdash \Pi[x'/x] \wedge x = E[x'/x] : (y \mapsto E * \Sigma)[x'/x]\}\ C\ \{\psi\}}{\vdash \{\Pi : y \mapsto E * \Sigma\}\ x \coloneqq [y]; C\ \{\psi\}}$$

($x'$ はフレッシュな変数 )

規則の結論 $\{\Pi : y \mapsto E * \Sigma\}\ x \coloneqq [y]; C\ \{\psi\}$ が $n$-invalid であるとすると，$(s,h) \models \Pi : y \mapsto E * \Sigma$ なる $s,h$ が存在し，$((x \coloneqq [y]; C, s), h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} fault$ である．

ここで，$s'' = s[x' \mapsto [\![x]\!]s][x \mapsto h([\![y]\!]s)]$ とおく．

まず，$(s'',h) \models \Pi[x'/x] \wedge x = E_i[x'/x] : (y \mapsto E * \Sigma)[x'/x]$ であることを示す．

$(s,h) \models \Pi : y \mapsto E * \Sigma$

$\Leftrightarrow (s[x' \mapsto [\![x]\!]s], h) \models \Pi[x'/x] : (y \mapsto E * \Sigma)[x'/x]$

($\because x'$ はフレッシュ)

$\Leftrightarrow (s[x' \mapsto [\![x]\!]s][x \mapsto [\![E]\!]s], h)$

$\models \Pi[x'/x] : (y \mapsto E * \Sigma)[x'/x]$

ここで，$s'' = s[x' \mapsto [\![x]\!]s][x \mapsto [\![E]\!]s]$ なので，$[\![x]\!]s'' = [\![E]\!]s$ と $[\![E[x'/x]]\!]s'' = [\![E]\!]s$ より，$[\![x]\!]s'' = [\![E[x'/x]]\!]s''$．ゆえに，

$$(s'',h) \models \Pi[x'/x] \wedge x = E[x'/x] : (y \mapsto E * \Sigma)[x'/x]$$

一方，$[\![y]\!]s \in dom(h)$ なので，実行系列は定義 2.3 より，$((x \coloneqq [y]; C, s), h) \rightsquigarrow ((C, s[x \mapsto [\![E]\!]s]), h) \overset{n-1}{\rightsquigarrow} \kappa (= (s',h')$ または $fault)$ の形に限られる．$x'$ はフレッシュなので，補題 A.1 より，$((C, s[x \mapsto [\![E]\!]s][x' \mapsto [\![x]\!]s]), h) \overset{n-1}{\rightsquigarrow} \kappa' (= (s'[x' \mapsto [\![x]\!]s], h')$ または $fault)$ が成り立つ．$x \neq x'$ より $s[x \mapsto [\![E]\!]s][x' \mapsto [\![x]\!]s] = s''$ であり，さらに，$(s',h') \not\models \psi$ と $x' \notin \mathsf{fv}(\psi)$ より $(s'[x' \mapsto [\![x]\!]s], h') \not\models \psi$ であるから，規則の仮定 $\{\Pi[x'/x] \wedge x = E[x'/x] : (y \mapsto E * \Sigma)[x'/x]\}\ C\ \{\psi\}$ は $(n-1)$-invalid である．

**(FREE)**

$$\frac{\vdash \{\Pi : \Sigma\}\ C\ \{\psi\}}{\vdash \{\Pi : x \mapsto E * \Sigma\}\ \texttt{free}(x); C\ \{\psi\}}$$

規則の結論 $\{\Pi : x \mapsto E * \Sigma\}\ \texttt{free}(x); C\ \{\psi\}$ が $n$-invalid であるとすると，$(s,h) \models \Pi : x \mapsto E * \Sigma$ なる $s,h$ が存在し，$((\texttt{free}(x); C, s), h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} fault$ である．

ここで，$h'' = h \upharpoonright dom(h) \backslash \{[\![x]\!]s\}$ とおく．

まず，$(s,h'') \models \Pi : \Sigma$ であることを示す．定義 2.5 より，

$(s,h) \models \Pi : x \mapsto E * \Sigma$

$\Leftrightarrow (s,h) \models \Pi$ かつ，ある $h_1, h_2$ があって

$h = h_1 \circ h_2$ かつ $dom(h_1) = \{[\![x]\!]s\}$ かつ

$h_1([\![x]\!]s) = [\![E]\!]s$ かつ $(s,h_2) \models \Sigma$

ここで，$h'' = h \upharpoonright dom(h) \backslash \{[\![x]\!]s\}$ であるから，

$$h'' = h_2 \text{ かつ } (s,h'') \models \Sigma$$

が成り立つ．ゆえに

$$(s,h'') \models \Pi : \Sigma$$

が成り立つ．

ここで，$[\![x]\!]s \in dom(h)$ なので，$((\texttt{free}(x); C, s), h))(= \kappa_0) \rightsquigarrow \kappa_1$ となる $\kappa_1$ は $((C,s),h'')$ のみである．よって，$\kappa_0 \rightsquigarrow ((C,s),h'') \overset{n-1}{\rightsquigarrow} (s',h') \not\models \psi$ または $\overset{n-1}{\rightsquigarrow} fault$ である．よって，規則の仮定 $\{\Pi : \Sigma\}\ C\ \{\psi\}$ は $(n-1)$-invalid である．

**(NEW)**

$$\frac{\vdash \{\Pi[x'/x] : x \mapsto y * \Sigma[x'/x]\}\ C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ x \coloneqq \texttt{new()}\ ; C\ \{\psi\}}$$

($x'$ と $y$ はフレッシュな変数)

規則の結論 $\{\Pi : \Sigma\}\ x \coloneqq \texttt{new()}\ ; C\ \{\psi\}$ が $n$-invalid であるとすると，$(s,h) \models \Pi : \Sigma$ なる $s,h$ が存在し，$((x \coloneqq \texttt{new()}\ ; C, s), h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi$ または，

$\overset{n}{\rightsquigarrow}$ *fault* である．定義 2.3 より，この実行系列の最初のステップは，$l \notin dom(h)$ なる $l$ と $v \in \mathsf{Val}$ があって，$((x \coloneqq \texttt{new()}\ ; C, s), h) \rightsquigarrow ((C, s[x \mapsto l]), h[l \mapsto v])$ の形である．

この $l, v$ に対して，$s'' = s[x' \mapsto [\![x]\!]s][x \mapsto l], h'' = h[l \mapsto v]$ とおく．

まず，$(s'', h'') \models \Pi[x'/x] : x \mapsto v * \Sigma[x'/x]$ であることを示す．

$$
\begin{aligned}
&(s, h) \models \Pi : \Sigma \\
\Leftrightarrow &(s[x' \mapsto [\![x]\!]s], h) \models \Pi[x'/x] : \Sigma[x'/x] \\
&(\because x' \text{ はフレッシュ}) \\
\Rightarrow &(s[x' \mapsto [\![x]\!]s][x \mapsto l], h[l \mapsto v]) \\
&\quad \models \Pi[x'/x] : x \mapsto v * \Sigma[x'/x] \\
&(\because l \notin dom(h))
\end{aligned}
$$

ここで，$s'' = s[x' \mapsto [\![x]\!]s][x \mapsto l]$，$h'' = h[l \mapsto v]$ であるから，

$$(s'', h'') \models \Pi[x'/x] : x \mapsto v * \Sigma[x'/x]$$

が成り立つ．

一方，実行系列 $((C, s[x \mapsto l]), h[l \mapsto v]) \overset{n-1}{\rightsquigarrow} (s', h')$ または *fault* に対して補題 A.1 を適用すると，$x'$ はフレッシュより，$((C, s[x \mapsto l][x' \mapsto [\![x]\!]s], h[l \mapsto v]) \overset{n-1}{\rightsquigarrow} s'[x' \mapsto [\![x]\!]s], h')$ または *fault* が成り立つ．$x \neq x'$ より $s[x \mapsto l][x' \mapsto [\![x]\!]s] = s''$ であり，$(s', h') \not\models \psi$ と $x' \notin \mathsf{fv}(\psi)$ であることより $(s'[x' \mapsto [\![x]\!]s], h') \not\models \psi$ であるから，規則の仮定 $\{\Pi[x'/x] : x \mapsto y * \Sigma[x'/x]\}\ C\ \{\psi\}$ は $(n-1)$-invalid である．

**(IF-DET)**

$$\frac{\vdash \{\Pi \wedge B : \Sigma\}\ C_1; C\ \{\psi\} \quad \vdash \{\Pi \wedge \bar{B} : \Sigma\}\ C_2; C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C\ \{\psi\}}$$

規則の結論

$\{\Pi : \Sigma\}\ \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C\ \{\psi\}$

が $n$-invalid であるとすると，$(s, h) \models \Pi : \Sigma$ なる $s, h$ が存在し，$((\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C, s), h) \overset{n}{\rightsquigarrow} (s', h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow}$ *fault* である．

$s \in [\![B]\!]$ のとき．定義 2.3 より，実行系列は，$(\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C, s), h)$ $\rightsquigarrow ((C_1; C, s), h) \overset{n-1}{\rightsquigarrow} (s', h')$ または *fault* の形に限られる．$(s, h) \models \Pi \wedge B : \Sigma$ であるから，仮定 $\{\Pi \wedge B : \Sigma\}\ C_1; C\ \{\psi\}$ は $(n-1)$-invalid である．

$s \in [\![\bar{B}]\!]$ のとき．定義 2.3 より，実行系列は，$(\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C, s), h)$ $\rightsquigarrow ((C_2; C, s), h) \overset{n-1}{\rightsquigarrow} (s', h')$ または *fault* の形に限られる．$(s, h) \models \Pi \wedge \bar{B} : \Sigma$ であるから，仮定 $\{\Pi \wedge \bar{B} : \Sigma\}\ C_2; C\ \{\psi\}$ は $(n-1)$-invalid である．

**(IF-NONDET)**

$$\frac{\vdash \{\phi\}\ C_1; C\ \{\psi\} \quad \vdash \{\phi\}\ C_2; C\ \{\psi\}}{\vdash \{\phi\}\ \texttt{if} \star \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C\ \{\psi\}}$$

規則の結論 $\{\phi\}\ \texttt{if} \star \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C\ \{\psi\}$ が $n$-invalid であるとすると，$(s, h) \models \phi$ なる $s, h$ が存在し，$((\texttt{if}\ \star\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C, s), h) \overset{n}{\rightsquigarrow} (s', h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow}$ *fault* である．定義 2.3 より，実行系列の最初のステップ $(\texttt{if}\ \star\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}; C, s), h) \rightsquigarrow \kappa$ の $\kappa$ は $((C_1; C, s), h)$ または $((C_2; C, s), h)$ のいずれかである．

$((C_1; C, s), h)$ のとき，$(s, h) \models \phi$ であり，$((C_1; C, s), h) \overset{n-1}{\rightsquigarrow} (s', h') \not\models \psi$ または *fault* なので，規則の仮定 $\{\phi\}\ C_1; C\ \{\psi\}$ は $(n-1)$-invalid である．

$((C_2; C, s), h)$ のときも同様に，規則の仮定 $\{\phi\}\ C_2; C\ \{\psi\}$ は $(n-1)$-invalid である．

**(WHILE-DET)**

$$\frac{\vdash \{\Pi \wedge B : \Sigma\}\ C'; \texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\} \quad \vdash \{\Pi \wedge \bar{B} : \Sigma\}\ C\ \{\psi\}}{\vdash \{\Pi : \Sigma\}\ \texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\}}$$

規則の結論 $\{\Pi : \Sigma\}\ \texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\}$ が $n$-invalid であるとすると，$(s, h) \models \Pi : \Sigma$ なる $s, h$ が存在し，$((\texttt{while}\ \pi\ \texttt{do}\ C'\ \texttt{od}; C, s), h) \overset{n}{\rightsquigarrow} (s', h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow}$ *fault* である．

$s \in [\![B]\!]$ のとき．定義 2.3 より，実行系列は，$((\texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C, s), h)$ $\rightsquigarrow ((C'; \texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C, s), h) \overset{n-1}{\rightsquigarrow} (s', h')$ または *fault* である．$(s, h) \models \Pi \wedge B : \Sigma$ であるから，規則の仮定 $\{\Pi \wedge B : \Sigma\}\ C'; \texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C\ \{\psi\}$ は $(n-1)$-invalid である．

$s \in [\![\bar{B}]\!]$ のとき．定義 2.3 より，実行系列は，$((\texttt{while}\ B\ \texttt{do}\ C'\ \texttt{od}; C, s), h) \rightsquigarrow ((C, s), h) \overset{n-1}{\rightsquigarrow} (s', h')$ または *fault* である．$(s, h) \models \Pi \wedge \bar{B} : \Sigma$ であるから，規則の仮定 $\{\Pi \wedge \bar{B} : \Sigma\}\ C\ \{\psi\}$ は $(n-1)$-invalid である．

**(WHILE-NONDET)**

$$\frac{\vdash \{\phi\}\ C';\mathtt{while} \star \mathtt{do}\ C'\ \mathtt{od}; C\ \{\psi\} \quad \vdash \{\phi\}\ C\ \{\psi\}}{\vdash \{\phi\}\ \mathtt{while} \star \mathtt{do}\ C'\ \mathtt{od}; C\ \{\psi\}}$$

規則の結論 $\{\phi\}$ $\mathtt{while} \star \mathtt{do}$ $C'$ $\mathtt{od}; C$ $\{\psi\}$ が $n$-invalid であるとすると，$(s,h) \models \phi$ なる $s,h$ が存在し，$((\mathtt{while}\ \star\ \mathtt{do}\ C'\ \mathtt{od}; C, s), h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} \mathit{fault}$ である．定義 2.3 より，最初のステップ $((\mathtt{while}\ \star\ \mathtt{do}\ C'\ \mathtt{od}; C, s), h) \rightsquigarrow \kappa$ の $\kappa$ は，$((C';\mathtt{while}\ \star\ \mathtt{do}\ C'\ \mathtt{od}; C, s), h)$ または $((C, s), h)$ のいずれかである．

$\kappa = ((C';\mathtt{while}\ \star\ \mathtt{do}\ C'\ \mathtt{od}; C, s), h)$ のとき，$(s,h) \models \phi$ であり，$\kappa \overset{n-1}{\rightsquigarrow} (s',h')$ または $\mathit{fault}$ なので，規則の仮定 $\{\phi\}\ C';\mathtt{while} \star \mathtt{do}\ C'\ \mathtt{od}; C\ \{\psi\}$ は $(n-1)$-invalid である．

$\kappa = ((C, s), h)$ のとき，$(s,h) \models \phi$ であり，$\kappa \overset{n-1}{\rightsquigarrow} (s',h')$ または $\mathit{fault}$ なので，規則の仮定 $\{\phi\}\ C\ \{\psi\}$ は $(n-1)$-invalid である．

**(PROC)**

$$\frac{\vdash \{\phi\}\ \mathit{body}(p)\ \{\psi\}}{\vdash \{\phi\}\ p(\vec{x})\{\psi\}}$$
$(\vec{x} = \mathit{params}(p), \mathit{locals}(p) \cap (\mathsf{fv}(\phi) \cup \mathsf{fv}(\psi)) = \emptyset)$

規則の結論 $\{\phi\}\ p(\vec{x})\{\psi\}$ が $n$-invalid であるとすると，$(s,h) \models \phi$ なる $s,h$ が存在し，$((p(\vec{x}), s), h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} \mathit{fault}$ である．定義 2.3 より，ある $s''$ が存在して，実行系列の最初のステップは $((p(\vec{x}), s), h) \rightsquigarrow ((\mathit{body}(p), s'') \cdot (\epsilon, s), h)$ の形である．(ただし，$[\![\vec{x}]\!]s = [\![\vec{x}]\!]s''$) さらに，ある $s''_1$ が存在して $((\mathit{body}(p), s'') \cdot (\epsilon, s), h) \overset{n-2}{\rightsquigarrow} ((\epsilon, s''_1) \cdot (\epsilon, s), h_1) \rightsquigarrow ((\epsilon, s), h_1)$，もしくは $((\mathit{body}(p), s'') \cdot (\epsilon, s), h) \overset{n-1}{\rightsquigarrow} \mathit{fault}$ の形である．

ここで，$\mathit{locals}(p) = \vec{y}$ とおいて $s''' = s[\vec{y} \mapsto [\![\vec{y}]\!]s'']$ なる $s'''$ を取る．このとき，$\mathit{locals}(p) \cap \mathsf{fv}(\phi) = \emptyset$ と $[\![\vec{x}]\!]s = [\![\vec{x}]\!]s''$ より $s''', h \models \phi$ が成り立つ．

補題 A.2 より，$s''_1 =_{\mathit{locals}(p) \cup \mathit{params}(p)} s'''_1$ であるような $s'''_1$ があって，$((\mathit{body}(p), s''') \cdot (\epsilon, s), h) \overset{n-2}{\rightsquigarrow} ((\epsilon, s'''_1) \cdot (\epsilon, s), h_1) \rightsquigarrow ((\epsilon, s), h_1)$，もしくは $((\mathit{body}(p), s''') \cdot (\epsilon, s), h) \overset{n-2}{\rightsquigarrow} \mathit{fault}$ を得る．

ここで，仮定より最終状態 $\kappa$ について $\kappa = (s, h_1) \not\models \psi$ もしくは $\kappa = \mathit{fault}$ が成り立つ．

ここで，$\mathit{locals}(p) = \mathit{vars}(\mathit{body}(p)) \setminus \mathit{params}(p)$，$\mathit{locals}(p) \cap \mathsf{fv}(\phi) = \emptyset$ より $((\mathit{body}(p), s'''), h) \overset{n-2}{\rightsquigarrow} ((\epsilon, s'''_1), h_1)$ もしくは $((\mathit{body}(p), s'''), h) \overset{n-1}{\rightsquigarrow} \mathit{fault}$ を得る．これと，$\mathit{locals}(p) \cap \mathsf{fv}(\psi) = \emptyset$，$s''' = s[\vec{y} \mapsto [\![\vec{y}]\!]s'']$ より，$s'''_1$ は $\mathit{locals}(p)$ を除いて $s$ と一致しているから $((\epsilon, s'''_1), h_1) \not\models \psi$ を得る．よって，規則の仮定 $\{\phi\}\ \mathit{body}(p)\ \{\psi\}$ は，$(n-2)$- または $(n-1)$-invalid である．

### Logical Proof Rules

**(CONSEQUENCE)**

$$\frac{\vdash \{\chi\}\ C\ \{\xi\}}{\vdash \{\phi\}\ C\ \{\psi\}}$$
$(\phi \models \chi, \xi \models \psi)$

規則の結論 $\{\phi\}$ $C$ $\{\psi\}$ が $n$-invalid であるとすると，$(s,h) \models \phi$ なる $s,h$ が存在し，$((C, s), h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} \mathit{fault}$ である．

$(s,h) \models \phi$ と $\phi \models \chi$ より，$(s,h) \models \chi$ であり，また $(s',h') \not\models \psi$ と $\xi \models \psi$ より，$(s',h') \not\models \xi$ であるから，規則の仮定 $\{\chi\}\ C\ \{\xi\}$ は $n$-invalid である．

**(FRAME)**

$$\frac{\vdash \{\phi\}\ C\ \{\psi\}}{\vdash \{\phi * \xi\}\ C\ \{\psi * \xi\}}$$
$(\mathsf{fv}(\xi) \cap \mathit{mod}(C) = \emptyset)$

規則の結論 $\{\phi * \xi\}$ $C$ $\{\psi * \xi\}$ が $n$-invalid であるとすると，$(s,h) \models \phi * \xi$ なる $s,h$ が存在し，$((C, s), h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi * \xi$ または，$\overset{n}{\rightsquigarrow} \mathit{fault}$ である．また，$(s,h) \models \phi * \xi$ より，ある $h_1, h_2$ があって，$h = h_1 \circ h_2$，$(s, h_1) \models \phi$，$(s, h_2) \models \xi$ が成り立つ．

$((C, s), h) \overset{n}{\rightsquigarrow} (s',h')$ のとき．補題 A.6.1 より，(a) $h' = h'_1 \circ h_2$ なる $h'_1$ があって，$((C, s), h_1) \overset{n}{\rightsquigarrow} (s', h'_1)$，または (b) ある $m \leqslant n$ があって，$((C, s), h_1) \overset{m}{\rightsquigarrow} \mathit{fault}$，のいずれかが成り立つ．

(a) のとき，$(s', h'_1) \models \psi$ とすると，$(s', h') \models \psi * \xi$ となり矛盾．よって，$(s', h'_1) \not\models \psi$ なので，規則の仮定 $\{\phi\}\ C\ \{\psi\}$ は $n$-invalid である．

(b) のとき，規則の仮定 $\{\phi\}\ C\ \{\psi\}$ は $m$-invalid である．

$((C, s), h) \overset{n}{\rightsquigarrow} \mathit{fault}$ のとき．補題 A.6.2 より，ある $m \leqslant n$ があって $((C, s), h_1) \overset{n}{\rightsquigarrow} \mathit{fault}$ なので，規則

の仮定 $\{\phi\}\ C\ \{\psi\}$ は $m$-invalid である．

**(SUBST)**

$$\frac{\vdash \{\phi\}\ C\ \{\psi\}}{\vdash \{\phi[E/x]\}\ C\ \{\psi[E/x]\}}$$
$(x \notin vars(C), x \in \mathsf{fv}(\psi) \Rightarrow E \notin vars(C))$

規則の結論 $\{\phi[E/x]\}\ C\ \{\psi[E/x]\}$ が $n$-invalid であるとすると，$(s,h) \models \phi[E/x]$ なる $s,h$ が存在し，$((C,s),h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi[E/x]$ または，$\overset{n}{\rightsquigarrow} fault$ である．

$s'' = s[x \mapsto [\![E]\!]s]$ とおくと，$(s'',h) \models \phi$ が成り立つ．

$x \notin vars(C)$ より補題 A.1 から $((C,s''),h) \overset{n}{\rightsquigarrow} (s'[x \mapsto [\![E]\!]s],h')$ を得る．

$\mathsf{fv}(E) \cap \mathsf{fv}(C) = \varnothing$ のとき，$\mathsf{fv}(E)$ の値は $((C,s),h) \overset{n}{\rightsquigarrow} (s',h')$ で不変．ゆえに $[\![E]\!]s = [\![E]\!]s'$．よって，$(s'[x \mapsto [\![E]\!]s],h') \not\models \psi$．

$x \notin \mathsf{fv}(\psi)$ のとき，$(s'[x \mapsto [\![E]\!]s'],h') \not\models \psi$ より $(s'[x \mapsto [\![E]\!]s],h') \not\models \psi$．

したがって，いずれの場合も規則の仮定 $\{\phi\}\ C\ \{\psi\}$ は $n$-invalid である．

**(PARAM)**

$$\frac{\vdash \{\phi\}\ p(\vec{E})\ \{\psi\}}{\vdash \{\phi[E/x]\}\ p(\vec{E}[E/x])\ \{\psi[E/x]\}}$$

規則の結論 $\{\phi[E/x]\}\ p(\vec{E}[E/x])\ \{\psi[E/x]\}$ が $n$-invalid であるとすると，$(s,h) \models \phi[E/x]$ なる $s,h$ が存在し，$((p(\vec{E}[E/x]),s),h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi[E/x]$ または，$\overset{n}{\rightsquigarrow} fault$ である．

定義 2.3 より $((p(\vec{E}[E/x])s),h)$ からの実行系列は $((p(\vec{E}[E/x]),s),h) \rightsquigarrow ((body(p),[\vec{x} \mapsto [\![\vec{E}[E/x]]\!]s]) \cdot (\epsilon,s),h) \overset{m}{\rightsquigarrow} ((\epsilon,\hat{s}) \cdot (\epsilon,s),h') \overset{m'}{\rightsquigarrow} (s,h')$ と表せる．(ただし，$1+m+m'=n$)

ここで，$[\vec{x} \mapsto [\![\vec{E}[E/x]]\!]s] = [\![\vec{E}]\!]s[x \mapsto [\![E]\!]s] = [\![\vec{E}]\!]s''$ であることと，定義 2.3 より，

$((p(\vec{E}),s''),h'') \rightsquigarrow ((body(p),[\vec{x} \mapsto [\![\vec{E}]\!]s'']) \cdot (\epsilon,s''),h'') \overset{m}{\rightsquigarrow} ((\epsilon,\hat{s}) \cdot (\epsilon,s''),h') \overset{m'}{\rightsquigarrow} (s'',h')$ を得る．ここで，$(s,h') \not\models \psi[E/x]$ より，$(s'',h') \not\models \psi$ が成り立つ．よって，規則の仮定 $\{\phi\}\ p(\vec{E})\ \{\psi\}$ は $n$-invalid である．

**($\exists$VAR)**

$$\frac{\vdash \{\phi[y/x]\}\ C\ \{\psi\}}{\vdash \{\exists x.\phi\}\ C\ \{\psi\}}$$
($y$ はフレッシュな変数)

規則の結論 $\{\exists x.\phi\}\ C\ \{\psi\}$ が $n$-invalid であるとすると，$(s,h) \models \exists x.\phi$ なる $s,h$ が存在し，$((C,s),h) \overset{n}{\rightsquigarrow} (s',h') \not\models \psi$ または，$\overset{n}{\rightsquigarrow} fault$ である．定義 2.5 より，ある $v \in \mathsf{Val}$ があって，$(s[x \mapsto v],h) \models \phi$ である．

$s'' = s[y \mapsto v]$ とおくと，$(s'',h) \models \phi[y/x]$ が成り立つ．

$y$ はフレッシュなので，補題 A.1 より，$((C,s''),h) \overset{n}{\rightsquigarrow} (s'[y \mapsto v],h')$ または，$\overset{n}{\rightsquigarrow} fault$ である．$y \notin \mathsf{fv}(\psi)$ より，$(s'[y \mapsto v],h') \not\models \psi$．したがって，規則の仮定 $\{\phi[y/x]\}\ C\ \{\psi\}$ は $n$-invalid である．

**(SEQ)**

$$\frac{\vdash \{\phi\}\ C_1\ \{\chi\} \qquad \vdash \{\chi\}\ C_2\ \{\psi\}}{\vdash \{\phi\}\ C_1;C_2\ \{\psi\}}$$

規則の結論 $\{\phi\}\ C_1;C_2\ \{\psi\}$ が n-invalid であるとすると，$(s,h) \models \phi$ なる $s,h$ が存在し，ある最終状態 $\kappa$ について $((C_1;C_2,s),h) \overset{n}{\rightsquigarrow} \kappa(=(s',h'))$ かつ $(s',h') \not\models \psi$．あるいは $((C_1;C_2,s),h) \overset{n}{\rightsquigarrow} fault$ が成り立つ．

(1)．$((C_1;C_2,s),h) \overset{n}{\rightsquigarrow} \kappa(=(s',h'))$ かつ $(s',h') \not\models \psi$ の場合．

補題 A.4 より，$((C_1,s),h) \overset{m}{\rightsquigarrow} (s'',h'')$，$(m \leqslant n)$ を得る．

Case 1．$(s'',h'') \models \chi$ の場合．補題 A.4 より，$((C_1;C_2,s),h) \overset{m}{\rightsquigarrow} ((C_2,s''),h'') \overset{n-m}{\rightsquigarrow} (s',h')$ を得る． ここで，$(s',h') \not\models \psi$ より，$\{\chi\}\ C_2\ \{\psi\}$ は $(n-m)$-invalid である．

Case 2．$(s'',h'') \not\models \chi$ の場合．$\{\phi\}\ C_1\ \{\chi\}$ は $m$-invalid である．

(2)．$((C_1;C_2,s),h) \overset{n}{\rightsquigarrow} falult$ の場合．

補題 A.5 より，以下の 2 つの場合が考えられる．

Case 1． $((C_1,s),h) \overset{m}{\rightsquigarrow} fault$ $(m \leqslant n)$ の場合．$\{\phi\}\ C_1\ \{\chi\}$ は $m$-invalid である．

Case 2． $((C_1,s),h) \overset{m}{\rightsquigarrow} (s'',h'')$ なる $(s'',h'')$ が存在して，$((C_1;C_2,s),h) \overset{m}{\rightsquigarrow} ((C_2,s''),h'') \overset{n-m}{\rightsquigarrow} fault$ の場合． $\{\chi\}\ C_2\ \{\psi\}$ は $(n-m)$-invalid である．

□