ヒープ操作を含むプログラムに対するトレース論理の拡張

郷 颯真　中澤 巧爾

本研究では、プログラムの実行過程であるトレースの性質を表すトレース論理を、ヒープ操作を含むプログラムへ拡張する。ヒープを扱うプログラムでは、メモリの確保、参照、更新、解放といった操作において、未確保領域へのアクセスによるメモリ・エラーの発生を考慮する必要がある。そこで本研究では、トレース論理にヒープ操作コマンドに対応する二項関係を追加し、メモリ・エラーによる異常終了を表す状態 abort を含むトレースを考慮することで、正常実行とメモリ・エラーによる異常終了を論理式のレベルで区別できるよう、トレース論理を拡張する。これにより、論理式によってメモリ・エラーが発生しないことを表現できるような枠組を与える。さらに、拡張トレース論理のための循環証明体系を与え、その健全性と相対完全性を証明する。

1　はじめに

トレース論理[2] は、プログラムの実行過程を状態の列として表現したトレースが満たす性質を論理式によって記述するための形式体系である。トレース論理では、ホーア論理のような事前・事後条件だけではなく、実行過程の途中の時点において成立すべき性質を記述することができる。また、不動点演算子により再帰関数やループを含むプログラムの性質を表現することができる。トレース論理においては、各プログラムに対して最強のトレース論理式を構成することができ、その論理式と仕様との間のエンテイルメント（含意関係）を示すことにより、プログラムの実行過程を含めた性質を検証できる。

本研究では、トレース論理に分離論理[4] の考え方を導入し、ヒープ操作を含むプログラムを扱えるようトレース論理を拡張する。従来のトレース論理は、主として変数の値の変化や制御を対象としており、動的に確保されるヒープ・メモリ領域に対する操作を含むプログラムを対象としていなかった。そのため、メモリ領域の確保、値の読出し、値の書換え、領域の解放といったヒープ操作を含むプログラムを検証するためには、トレース論理式や証明体系を拡張する必要がある。

本研究の成果は以下のとおりである.

1. (**メモリ操作に対するトレース論理式の拡張**) メモリ領域の確保、読出し、書換え、解放による実行状態の（正常な）遷移を、基本的な関係として論理式で記述できるようにした。
2. (**トレース論理によるメモリ安全性検証**) メモリエラーで終了することを表す論理式を導入し、正常な実行とメモリエラーによって中断する可能性のある実行を区別して表現できるようにした。また、それらを証明体系において検証できるようにした。
3. (**循環証明体系の導入**) 拡張されたトレース論理に対して、再帰呼び出しやループを含むプログラムに対する帰納的な性質を証明するための循環証明体系を導入した。この循環証明体系の健全性と相対完全性を証明した。

---

*An extension of trace logic for heap manipulating programs. (This is an unrefereed paper. Copyrights belong to the authors.)

Soma Go, Koji Nakazawa, 名古屋大学大学院情報学研究科, Graduate school of informatics, Nagoya University.

2 拡張トレース論理の定義

2.1 拡張トレース意味論

本節では Gurov らのトレース論理 [2] をヒープ操作を含むプログラムに対して拡張した拡張トレース論理を定義する。トレース論理は実行過程を表す状態列（トレース）の性質を証明する論理体系である。本研究では、これをヒープ操作を含むプログラムに対して拡張する。

定義 2.1 (プログラムの構文)**.**

プログラムは組 $\langle S, T\rangle$ として定義される。ここで、文 S は次の構文で与えられる。

$$S ::= \mathsf{skip} \mid x := a \mid S_1; S_2 \mid \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \\ \mid m() \mid x = \mathsf{alloc}(a) \mid x = [a] \mid [a'] = a \mid \mathsf{free}(a)$$

ここで、x は変数、a および a' は算術式、b は真偽式、m は手続き名を表す。T は手続き定義

$$T ::= m_1\{S_1\}, \ldots, m_n\{S_n\}$$

であり，各手続き名 m_i に対してその定義である文 S_i を割り当てる。

skip、代入文 $x := a$、逐次実行 $S_1; S_2$、条件分岐、および手続き呼出しは、それぞれ通常の意味を持つ。本研究では、これらに加えてヒープを操作する文を導入する。$x := \mathsf{alloc}(a)$ は、新たなヒープ領域を確保して値 a を格納し、そのアドレスを x に代入する。$x := [a]$ は、式 a が表すアドレスに格納された値を読み出し、その値を x に代入する。$[a'] := a$ は、式 a' が表すアドレスに値 a を書き込む。$\mathsf{free}(a)$ は、a が表すアドレスに対応するヒープ領域を解放する。

以下では、とくに断わらない限り、手続き定義 T を固定して考え、文 S のことをプログラムと呼ぶ。

プログラムの操作的意味論とトレース意味論は以下で与えられる。

定義 2.2 (状態とヒープ)**.**

1. 状態は変数環境 s とヒープ h の組 (s, h) で表す. 変数環境 s は変数から値への写像 $s : \mathrm{Var} \to \mathrm{Val}$ である。ヒープ h は番地から値への有限部分写像 $h : \mathrm{Loc} \rightharpoonup \mathrm{Val}$ であり、$\mathrm{dom}(h)$ は確保済み番地の有限集合を表す。変数環境 s のもとで、式 a が表す値を $[\![a]\!](s)$ と書く。s の x での値を v に更新した変数環境を $s[x \mapsto v]$ と書く。$h[l \mapsto v]$ も同様である。とくに、$l \notin \mathrm{dom}(h)$ のときは、定義域に l を追加する。また、$h[l \mapsto \bot]$ は h の定義域から l を除いたヒープとする。

2. メモリ・エラーを表す *abort* を特別な状態とする。メタ変数 σ によって、状態 (s, h) または *abort* を表す。

3. 状態の有限列をトレースと呼ぶ。トレースは $\sigma_1 \cdot \sigma_2 \cdots \sigma_n$ などで表す。メタ変数 τ によって、トレースを表す。トレース全体の集合を Tr とする。

定義 2.3 (操作的意味論)**.** 関係 $\langle S, (s, h)\rangle \Rightarrow \sigma$、および $\langle S, (s, h)\rangle \Rightarrow \langle S', (s', h')\rangle$ を、図 *1* の推論規則によって定義する。

定義 2.4 (トレース意味論)**.**

$[\![S]\!]$ はプログラム S の実行により得られるすべてのトレースの集合を表す。すなわち、

$$[\![S]\!] = \{\sigma_1 \cdot \sigma_2 \cdots \sigma_n \mid \\ \langle S, \sigma_1\rangle \Rightarrow \langle S_2, \sigma_2\rangle \Rightarrow \cdots \Rightarrow \sigma_n\}$$

とする。

2.2 拡張トレース論理式

トレース論理式 φ は、状態に関する述語や逐次実行や再帰に対応する論理演算子を用いて構成され、トレースが満たす性質を表す。

定義 2.5 (トレース論理式 [2])**.** トレース論理式は以下で定義される。

$$\varphi ::= R \mid Ab \mid X \mid p \wedge \varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \frown \varphi_2 \mid \mu X.\varphi$$

トレース論理式 ϕ は、プログラムのトレースが満たす性質を表したものである。

R はコマンド実行前後の状態の変化を表す二項関係 $Id, Sb_{x,a}, Al_{x,a}, Rf_{x,a}, Mt_{a',a}, Fr_a$ であり、それぞれ以下で定義される二項関係を表す。Ab 以外の二項関係はメモリエラーの起きない正常実行のみを含む。

定義 2.6 (二項関係)**.** 二項関係 $\|R\|$ を長さ *2* のトレースの集合として図 *2* のとおり定義する.

Ab は、メモリ・エラーによる終了を表す特殊な二項関係を表す。

$$\frac{}{\langle \mathsf{skip}, (s,h)\rangle \Rightarrow (s,h)}\ \textsc{Skip} \qquad \frac{}{\langle x := a, (s,h)\rangle \Rightarrow (s[x \mapsto [\![a]\!](s)], h)}\ \textsc{Assign}$$

$$\frac{\langle S_1, (s,h)\rangle \Rightarrow (s',h')}{\langle S_1; S_2, (s,h)\rangle \Rightarrow \langle S_2, (s',h')\rangle}\ \textsc{Seq-1} \qquad \frac{\langle S_1, (s,h)\rangle \Rightarrow \langle S_1', (s',h')\rangle}{\langle S_1; S_2, (s,h)\rangle \Rightarrow \langle S_1'; S_2, (s',h')\rangle}\ \textsc{Seq-2}$$

$$\frac{B[\![b]\!](s) = \mathrm{tt}}{\langle \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2, (s,h)\rangle \Rightarrow \langle S_1, (s,h)\rangle}\ \textsc{If-1} \qquad \frac{B[\![b]\!](s) = \mathrm{ff}}{\langle \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2, (s,h)\rangle \Rightarrow \langle S_2, (s,h)\rangle}\ \textsc{If-2}$$

$$\frac{m \text{ is declared as } m\{S\} \text{ in } T}{\langle m(), (s,h)\rangle \Rightarrow \langle S, (s,h)\rangle}\ \textsc{Call}$$

$$\frac{l \notin \mathrm{dom}(h)}{\langle x := \mathsf{alloc}(a), (s,h)\rangle \Rightarrow (s[x \mapsto l],\ h[l \mapsto [\![a]\!](s)])}\ \textsc{Alloc}$$

$$\frac{[\![a]\!](s) \in \mathrm{dom}(h)}{\langle x := [a], (s,h)\rangle \Rightarrow (s[x \mapsto h([\![a]\!](s))],\ h)}\ \textsc{Read} \qquad \frac{[\![a]\!](s) \notin \mathrm{dom}(h)}{\langle x := [a], (s,h)\rangle \Rightarrow \mathsf{abort}}\ \textsc{Read-Abort}$$

$$\frac{[\![a]\!](s) \in \mathrm{dom}(h)}{\langle [a] := a', (s,h)\rangle \Rightarrow (s,\ h[[\![a]\!](s) \mapsto [\![a']\!](s)])}\ \textsc{Mutate} \qquad \frac{[\![a]\!](s) \notin \mathrm{dom}(h)}{\langle [a] := a', (s,h)\rangle \Rightarrow \mathsf{abort}}\ \textsc{Mutate-Abort}$$

$$\frac{[\![a]\!](s) \in \mathrm{dom}(h)}{\langle \mathsf{free}(a), (s,h)\rangle \Rightarrow (s,\ h[[\![a]\!](s) \mapsto \bot])}\ \textsc{Free} \qquad \frac{[\![a]\!](s) \notin \mathrm{dom}(h)}{\langle \mathsf{free}(a), (s,h)\rangle \Rightarrow \mathsf{abort}}\ \textsc{Free-Abort}$$

図 1　操作的意味論

$$\begin{aligned}
\|Id\| &= \{(s,h)\cdot(s,h) \mid (s,h) \text{ は状態 }\}\\
\|Sb_{x,a}\| &= \{(s,h)\cdot(s[x \mapsto [\![a]\!](s)],h) \mid (s,h) \text{ は状態 }\}\\
\|Al_{x,a}\| &= \{(s,h)\cdot(s[x \mapsto l], h[\,l \mapsto [\![a]\!](s)\,]) \mid l \notin \mathrm{dom}(h)\}\\
\|Rf_{x,a}\| &= \{(s,h)\cdot(s[x \mapsto h([\![a]\!](s))],h) \mid [\![a]\!](s) \in \mathrm{dom}(h)\}\\
\|Mt_{a',a}\| &= \{(s,h)\cdot(s,h[[\![a]\!](s) \mapsto [\![a']\!](s)]) \mid [\![a]\!](s) \in \mathrm{dom}(h)\}\\
\|Fr_a\| &= \{(s,h)\cdot(s,h[[\![a]\!](s) \mapsto \bot]) \mid [\![a]\!](s) \in \mathrm{dom}(h)\}
\end{aligned}$$

図 2　二項関係

定義 2.7 (二項関係 Ab)**.** 二項関係 Ab を以下で定義する。

$$\|Ab\| = \{(s,h)\cdot abort \mid (s,h) \text{ は状態 }\}$$

本論文において、連言は状態に対する論理式 p に対して $p \wedge \varphi$ の形に制限する。状態論理式 p は状態 (s,h) の性質を表す論理式とし、(s,h) が p を満たすことを $(s,h) \models p$ と書く。ここで、状態論理式として論理式 $a\uparrow$ を含むとする。$a\uparrow$ は、a の示すアドレスがヒープ内に定義されていないことを表す。

定義 2.8 ($a\uparrow$ の定義)**.**

$$(s,h) \models a\uparrow \quad \overset{\mathrm{def}}{\Longleftrightarrow} \quad [\![a]\!](s) \notin \mathrm{dom}(h)$$

$\frown$ 演算子はトレースを連結する。

定義 2.9 (チョップ演算子 [2])**.** $A, B \subseteq \mathrm{Tr}$ に対して、

$$\begin{aligned}
A \frown B = \{\, &\tau_1\cdot\sigma\cdot\tau_2 \mid \tau_1\cdot\sigma \in A,\ \sigma\cdot\tau_2 \in B\,\}\\
&\cup \{\tau\cdot abort \mid \tau\cdot abort \in A\}
\end{aligned}$$

X は不動点演算のための再帰変数とし、$\mu X.\varphi$ は最小不動点を表す。

以上をふまえて、論理式の意味 $\|\varphi\|$ を以下で定義する。

定義 2.10 (トレース論理式の意味)**.** V を再帰変数 X に Tr を割当てる写像とする。トレース論理式 φ の意味 $\|\varphi\| \subseteq \mathrm{Tr}$ を以下で定義する。

$$\begin{aligned}
\|R\|_V &= \|R\| \\
\|Ab\|_V &= \|Ab\| \\
\|X\|_V &= V(X) \\
\|p \wedge \varphi\|_V &= \{\sigma \cdot \tau \mid \sigma \models p \text{ かつ } \sigma \cdot \tau \in \|\varphi\|\} \\
\|\varphi_1 \vee \varphi_2\|_V &= \|\varphi_1\|_V \cup \|\varphi_2\|_V \\
\|\varphi_1 \frown \varphi_2\|_V &= \|\varphi_1\|_V \frown \|\varphi_2\|_V \\
\|\mu X.\varphi\| &= \bigcup_{n \in \mathbb{N}} F^n_{\varphi,V}(\emptyset)
\end{aligned}$$

ここで、

$$F_{\varphi,V}(U) \stackrel{\mathrm{def}}{=} \|\varphi\|_{V[X \mapsto U]}$$

とし（$V[X \mapsto U]$ は、再帰変数 X の値だけを $U \subseteq \mathrm{Tr}$ に更新した付値）、$F^0(U) = S$、$F^{i+1}(U) = F(F^i(U))$ とする。

$F_{\varphi,V}$ は単調であるため、Knaster-Tarski の不動点定理より最小不動点が存在する。最小不動点の展開により、

$$\|\mu X.\varphi\|_V = \|\varphi[\mu X.\varphi/X]\|_V$$

が成り立つ。

定義 2.11. エンテイルメント $\varphi \models \psi$ は、$\|\varphi\| \subseteq \|\psi\|$ であることを意味する。

3 循環証明体系

3.1 循環証明体系の定義

本節では、文 S と論理式 φ に対して $S : \varphi$ の形の**判断**を証明するための循環証明体系を定義する。直観的に判断 $S : \varphi$ は、S の任意の実行トレースがトレース論理式 ϕ を満たすこと、すなわち、

$$[\![S]\!] \subseteq \|\varphi\|$$

を意味する。

循環証明体系 [1] とは、通常の有限な証明木に加えて、証明中に現れた判断を以前に現れた同じ形の判断へ戻すことを許す証明体系である。これにより、帰納的または再帰的に定義された性質を、有限の証明図によって証明することができる。ただし、任意の循環を許すと妥当でない判断が証明可能になるため、各無限枝において何らかの帰納的進行が無限回現れることを要求する。この条件を**大域トレース条件**という。

定義 3.1 (循環証明体系)**.** *1.* 以下の *2* つの要素からなる組 $\langle \pi, f \rangle$ を擬証明と呼ぶ。

- 図 *3.1* の推論規則によって構成される証明木 π。ただし、公理ではない葉を含んでいてもよい。
- π の公理ではない葉から、同一の判断をもつ π 中の葉ではないノードへの写像 f。

2. 擬証明 (π, f) の無限パスとは、π 中のノードの列 $N_1, N_2, \ldots$ で各 $i \in \mathbb{N}$ について以下のいずれかを満たすもののことである。

- N_{i+1} は、N_i を結論としてもつ π 中の推論規則の仮定。
- $f(N_i) = N_{i+1}$。

3. 擬証明のうち以下の条件を満たすものを循環証明という。

大域トレース条件：任意の無限パスが、無限回の Call 規則の適用を含む。

4. 判断 $S : \varphi$ を結論とする循環証明が存在するとき $\vdash S : \varphi$ と書く。

3.2 循環証明の例

例 3.2. メモリ操作を含むプログラム save() に対する仕様を証明する循環証明の例を挙げる。ここで、手続き save の定義は以下である。

$$\begin{aligned}
S_{\mathrm{save}} = \mathsf{if}\ p\ \mathsf{then\ skip} \\
\mathsf{else}\ x = \mathsf{alloc}(a);\ t = [x];\ \mathsf{free}(x);\ \mathrm{save}().
\end{aligned}$$

図 *4* に、循環証明の例を示す。ここで、ψ は save() に対して検証したいトレース論理式

$$\begin{aligned}
\psi \stackrel{\mathrm{def}}{=} emp \to Id \frown \mu X_{\mathrm{save}}.\Big(& (p \wedge Id \frown Id) \\
& \vee \big(\neg p \wedge Al_{x,a} \frown Rf_{t,x} \frown Fr_x \frown Id \frown \\
& (emp \wedge X_{\mathrm{save}})\big)\Big).
\end{aligned}$$

である。emp は、分離論理 *[4]* の論理式であり、状態論理式として利用している。意味は、ヒープが空であること、すなわち

$$(s,h) \models emp \stackrel{\mathrm{def}}{\Leftrightarrow} \mathrm{dom}(h) = \emptyset$$

で定義される。ψ は、実行前にヒープが空ならば手続き save の各呼出しの直前でヒープが空であることを表現している。このように、トレース論理では実行中の各時点で満たすべき性質を表すことができる。証明中に補助的に使っている論理式 φ_{save} は以下で定義さ

$$\frac{}{\vdash \mathsf{skip} : Id}\ \textsc{Skip} \qquad \frac{}{\vdash x := a : Sb_x^a}\ \textsc{Assign} \qquad \frac{\vdash S_1 : \varphi_1 \quad \vdash S_2 : \varphi_2}{\vdash S_1; S_2 : \varphi_1 \frown \varphi_2}\ \textsc{Seq} \qquad \frac{\Gamma \vdash S : \phi' \quad \phi' \models \phi}{\Gamma \vdash S : \phi}\ \textsc{Cons}$$

$$\frac{\vdash S_1 : \varphi_1 \quad \vdash S_2 : \varphi_2}{\vdash \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 : (b \wedge Id \frown \varphi_1) \vee (\neg b \wedge Id \frown \varphi_2)}\ \textsc{If} \qquad \frac{\vdash S_m : \varphi}{\vdash m() : Id \frown \varphi}\ \textsc{Call} \quad (m\{S_m\} \in T)$$

$$\frac{}{\vdash x = \mathsf{alloc}(a) : Al_x^a}\ \textsc{Alloc} \qquad \frac{}{\vdash x = [a] : Rd_x^a \vee (a\uparrow \wedge Ab)}\ \textsc{Read}$$

$$\frac{}{\vdash [a'] := a : Wr_{a'}^a \vee (a'\uparrow \wedge Ab)}\ \textsc{Write} \qquad \frac{}{\vdash \mathsf{free}(a) : Fr_a \vee (a\uparrow \wedge Ab)}\ \textsc{Free}$$

図 3　拡張トレース論理の推論規則

れる。

$$\varphi_{\mathrm{save}} \stackrel{\mathrm{def}}{=} (p \wedge Id \frown Id) \vee \big(\neg p \wedge Al_{x,a} \frown Rf_{t,x} \frown Fr_x \frown Id \frown X_{\mathrm{save}}\big),$$

下側の save() に関する判断は、上側に現れた同一の判断へ循環辺で接続される。この循環は Call 規則を含むため、大域トレース条件を満たす。

4　最強トレース論理式

本節では、任意のプログラム S のトレース集合 $[\![S]\!]$ を表現するトレース論理式 $\mathrm{stf}(S)$ が存在することを示す。

4.1　最強トレース論理式の定義

定義 4.1 (最強トレース論理式)**.** プログラム $\langle S, T\rangle$ に対して、文 S の最強トレース論理式を

$$\mathrm{stf}(S) \stackrel{\mathrm{def}}{=} \mathrm{stf}(\emptyset, S)$$

と定義する。ここで、補助関数 $\mathrm{stf}(\bar{X}, S)$ は図 5 のように帰納的に定義される。

最強トレース論理式は、プログラムのトレース集合をそのまま表す論理式である。図 5 中の $\bar{X}$ はすでに作成された再帰変数を記録するための集合である。各手続き宣言 $m\{S_m\} \in T$ について、手続き m が最初に呼び出されたときには不動点式 $\mu X_m.\mathrm{stf}(\bar{X} \cup \{m\}, S_m)$ を生成し、その後の同じ手続き m の呼び出しは再帰変数 X_m により表す。これにより、手続き呼び出しを無限に展開せずに、再帰的な実行トレースを有限の論理式として表現できる。

定理 4.2. 任意の文 S について、循環証明体系において $\vdash S : \mathrm{stf}(S)$ は証明可能である。

Proof. S の構造に関する帰納法で示す。ここでは逐次合成の場合を示す。

$S = S_1; S_2$ とする。帰納法の仮定より、

$$\vdash S_1 : \mathrm{stf}(S_1) \qquad \vdash S_2 : \mathrm{stf}(S_2)$$

が成り立つ。逐次合成に対する証明規則より、

$$\frac{\vdash S_1 : \mathrm{stf}(S_1) \quad \vdash S_2 : \mathrm{stf}(S_2)}{\vdash S_1; S_2 : \mathrm{stf}(S_1) \frown \mathrm{stf}(S_2)}\ \textsc{Seq}$$

を得る。最強トレース論理式の定義より、

$$\mathrm{stf}(S_1; S_2) = \mathrm{stf}(S_1) \frown \mathrm{stf}(S_2)$$

である。したがって、

$$\vdash S_1; S_2 : \mathrm{stf}(S_1; S_2)$$

が成り立つ。 □

定理 4.3. 任意の文 S について、

$$[\![S]\!] = \|\mathrm{stf}(S)\|$$

が成り立つ。

Proof. S の構造に関する帰納法で示す。ここでは $S = S_1; S_2$ の場合を示す。帰納法の仮定より、

$$[\![S_1]\!] = \|\mathrm{stf}(S_1)\|, \qquad [\![S_2]\!] = \|\mathrm{stf}(S_2)\|$$

が成り立つ。

逐次合成のトレース意味論より、

$$[\![S_1; S_2]\!] = [\![S_1]\!] \frown [\![S_2]\!]$$

である。帰納法の仮定を用いると、

$$[\![S_1; S_2]\!] = \|\mathrm{stf}(S_1)\| \frown \|\mathrm{stf}(S_2)\|$$

となる。

また、$\frown$ の意味論より、

$$\|\mathrm{stf}(S_1)\| \frown \|\mathrm{stf}(S_2)\| = \|\mathrm{stf}(S_1) \frown \mathrm{stf}(S_2)\|$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ \vdash \mathrm{save}() : Id \frown \mu X_{\mathrm{save}}.\varphi_{\mathrm{save}}}{\vdash x = \mathsf{alloc}(a); \cdots; \mathrm{save}() : Al_{x,a} \frown \cdots \frown Id \frown \mu X_{\mathrm{save}}.\varphi_{\mathrm{save}}}\ \textsc{Seq}}{\vdots\ \vdash x = \mathsf{alloc}(a); \cdots : \varphi_{\mathrm{save}}[\mu X_{\mathrm{save}}.\varphi_{\mathrm{save}}/X_{\mathrm{save}}]}\ \textsc{Cons}}{\vdash S_{\mathrm{save}} : \varphi_{\mathrm{save}}[\mu X_{\mathrm{save}}.\varphi_{\mathrm{save}}/X_{\mathrm{save}}]}\ \textsc{If}}{\vdash S_{\mathrm{save}} : \mu X_{\mathrm{save}}.\varphi_{\mathrm{save}}}\ \textsc{Unfold}}{\vdash \mathrm{save}() : Id \frown \mu X_{\mathrm{save}}.\varphi_{\mathrm{save}}}\ \textsc{Call}}{\vdash \mathrm{save}() : \psi}\ \textsc{Cons}\quad (Id \frown \mu X_{\mathrm{save}}.\varphi_{\mathrm{save}} \models \psi)$$

図 4　メモリ操作を含む循環証明の例

$$\begin{aligned}
\mathrm{stf}(\bar{X}, \mathsf{skip}) &\stackrel{\mathrm{def}}{=} Id,\\
\mathrm{stf}(\bar{X}, x := a) &\stackrel{\mathrm{def}}{=} Sb^a_x,\\
\mathrm{stf}(\bar{X}, S_1; S_2) &\stackrel{\mathrm{def}}{=} \mathrm{stf}(\bar{X}, S_1) \frown \mathrm{stf}(\bar{X}, S_2),\\
\mathrm{stf}(\bar{X}, \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2) &\stackrel{\mathrm{def}}{=} (b \wedge Id \frown \mathrm{stf}(\bar{X}, S_1)) \vee (\neg b \wedge Id \frown \mathrm{stf}(\bar{X}, S_2)),\\
\mathrm{stf}(\bar{X}, m()) &\stackrel{\mathrm{def}}{=} \begin{cases} Id \frown \mu X_m.\mathrm{stf}(\bar{X} \cup \{m\}, S_m) & m \notin \bar{X},\ m\{S_m\} \in T,\\ Id \frown X_m & \text{otherwise,} \end{cases}\\
\mathrm{stf}(\bar{X}, x = \mathsf{alloc}(a)) &\stackrel{\mathrm{def}}{=} Al^a_x\\
\mathrm{stf}(\bar{X}, x = [a]) &\stackrel{\mathrm{def}}{=} Rd^a_x \vee (a\uparrow \wedge Ab),\\
\mathrm{stf}(\bar{X}, [a'] := a) &\stackrel{\mathrm{def}}{=} Mt^a_{a'} \vee (a'\uparrow \wedge Ab),\\
\mathrm{stf}(\bar{X}, \mathsf{free}(a)) &\stackrel{\mathrm{def}}{=} Fr_a \vee (a\uparrow \wedge Ab),\\
\mathrm{stf}(\bar{X}, X_m) &\stackrel{\mathrm{def}}{=} X_m.
\end{aligned}$$

図 5　stf の定義

である。さらに、最強トレース論理式の定義より、

$$\mathrm{stf}(S_1; S_2) = \mathrm{stf}(S_1) \frown \mathrm{stf}(S_2)$$

である。したがって、

$$[\![S_1; S_2]\!] = \|\mathrm{stf}(S_1; S_2)\|$$

が従う。　□

プログラム S が要求される仕様 φ を満たすことは、最強トレース論理式を使えば、エンテイルメント $\mathrm{stf}(S) \models \varphi$ に帰着される。

4.2　メモリ安全性

以下では、Ab を含まない論理式を満たすトレースはメモリ・エラーを含まないことを示す。

定義 4.4 (Ab-free)**.**

$$\mathsf{AbFree}(\varphi) \quad \stackrel{\mathrm{def}}{\Longleftrightarrow} \quad Ab \notin \mathrm{Rel}(\varphi)$$

ここで、$\mathrm{Rel}(\varphi)$ は φ 中に現れる関係記号の集合である。

補題 4.5. トレース論理式 φ が *Ab-free* であるならば、$\|\varphi\|$ 中のトレースは abort を含まない。

Proof. φ の構造に関する帰納法で示す。異常終了を含まないトレース全体の集合を

$$\mathrm{Safe} = \{\, \sigma \in \mathrm{Trace} \mid \mathsf{abort} \notin \sigma \,\}$$

とする。示すべきことは、

$$\forall X.V(X) \subseteq \mathrm{Safe} \Longrightarrow \|\varphi\|_V \subseteq \mathrm{Safe}$$

である。

$\varphi = p$ の場合、状態論理式の意味は異常終了を導入しないため、$\|p\|_V \subseteq \mathrm{Safe}$ である。

$\varphi = R$ の場合、$\|R\|$ の定義より、$\|R\|_V \subseteq \mathrm{Safe}$ である。

$\varphi = X$ の場合、仮定より、

$$\|X\|_V = V(X) \subseteq \mathrm{Safe}$$

である。

$\varphi = p \wedge \varphi_1$ の場合、帰納法の仮定より、

$$\|\varphi_1\|_V \subseteq \mathrm{Safe}$$

である。よって、$\|\varphi\|_V \subseteq \|\varphi_1\|_V \subseteq \mathrm{Safe}$ である。

$\varphi = \varphi_1 \vee \varphi_2$ の場合、帰納法の仮定より、

$$\|\varphi_1\|_V \subseteq \mathrm{Safe}, \qquad \|\varphi_2\|_V \subseteq \mathrm{Safe}$$

である。集合の和集合はこの性質を保存するため、結論が成り立つ。

$\varphi = \varphi_1 \frown \varphi_2$ の場合、帰納法の仮定より、φ_1 および φ_2 を満たすトレースは abort を含まない。それらを連結して得られるトレースにも abort は含まれない。したがって、

$$\|\varphi_1 \frown \varphi_2\|_V \subseteq \mathrm{Safe}$$

である。

$\varphi = \mu X.\psi$ の場合、*Ab*-free の定義より ψ は *Ab*-free である。定義より、$\mu X.\psi$ は関数

$$F(U) = \|\psi\|_{V[X \mapsto U]}$$

の最小不動点である。帰納法の仮定より、U が abort を含まないトレース集合であるならば、$F(U)$ も abort を含まないトレース集合である。したがって、空集合からの近似列

$$\emptyset, \quad F(\emptyset), \quad F^2(\emptyset), \ldots$$

はいずれも abort を含まないトレース集合である。ゆえに、その最小不動点である $\|\mu X.\psi\| = \bigcup_n F^n(\emptyset)$ 中のトレースも abort を含まない。

以上より、すべての場合について $\|\varphi\|_V \subseteq \mathrm{Safe}$ が成り立つ。 □

オリジナルのトレース論理 [2] では、単体の状態論理式 p を論理式として認めており、その意味論を、

$$\|p\| = \{\sigma \cdot \tau \mid \sigma \models p\}$$

で定義している。すなわち、状態論理式 p は先頭の状態以外については何も制限を課さない（abort を含んでもよい）ので、補題が成立しない。このため本論文では、状態論理式の出現は $p \wedge \varphi$ の形に制限している。

定理 4.6. *プログラム S に対して、ある Ab-free な閉じたトレース論理式 φ が存在し、*

$$\mathrm{stf}(S) \models \varphi$$

が成り立つならば、S の実行においてメモリエラーは発生しない。

Proof. $\mathrm{stf}(S) \models \varphi$ より、

$$\|\mathrm{stf}(S)\| \subseteq \|\varphi\|$$

が成り立つ。

また、φ は *Ab*-free であるため、先の補題より、任意の $\sigma \in \|\varphi\|$ について、

$$\mathsf{abort} \notin \sigma$$

が成り立つ。

さらに、最強トレース論理式の特徴付けより、

$$[\![S]\!] = \|\mathrm{stf}(S)\|$$

である。したがって、

$$[\![S]\!] = \|\mathrm{stf}(S)\| \subseteq \|\varphi\| \subseteq \mathrm{Safe}$$

を得る。ただし、

$$\mathrm{Safe} = \{\, \sigma \mid \mathsf{abort} \notin \sigma \,\}$$

である。

よって、S のすべての実行トレースは abort を含まない。したがって、S の実行においてメモリエラーは発生しない。 □

例 4.7. プログラム $S = m()$ を考える。ここで、手続き m の定義は

$$S_m = \mathsf{if}\ p\ \mathsf{then}\ x = \mathsf{alloc}(a); \mathsf{free}(x); m()\ \mathsf{else}\ \mathsf{skip}$$

で与えられるとする。

このプログラムの最強トレース論理式は、

$$\begin{aligned}\mathrm{stf}(S) = Id \frown \mu X_m.\Big(&\big(p \wedge Id \frown Al_x^a \\ &\frown (Fr_x \vee (x\uparrow \wedge Ab)) \frown Id \frown X_m\big) \\ &\vee \big(\neg p \wedge Id \frown Id\big)\Big)\end{aligned}$$

であり、

$$\begin{aligned}\varphi = Id \frown \mu X_m.\Big(&\big(p \wedge Id \frown Al_x^a \frown Fr_x \\ &\frown Id \frown X_m\big) \vee \big(\neg p \wedge Id \frown Id\big)\Big).\end{aligned}$$

に対して、エンテイルメント

$$\mathrm{stf}(S) \models \varphi$$

が成り立つ。ただし、

論理式 φ は *Ab-free* である。したがって、

$$\mathrm{stf}(S) \models \varphi \qquad \text{かつ} \qquad \varphi \text{ は } \textit{Ab-free}$$

が成り立つ。よって、前定理より、S の実行においてメモリエラーは発生しない。

5 健全性と相対完全性

本章では、循環証明体系の健全性と相対完全性を示す。

（健全性） $\vdash S : \varphi \implies \models S : \varphi$

（相対完全性） $\models S : \varphi \implies \vdash S : \varphi$

とくに、相対完全性については、(Cons) 規則におけるエンテイルメント関係 $\varphi \models \psi$ を意味論的含意関係として扱っていることに注意する。

5.1 健全性

健全性を示すため、まず各推論規則の局所健全性を示す。一般に、推論規則

$$\frac{\vdash S_1 : \varphi_1 \quad \cdots \quad \vdash S_n : \varphi_n}{\vdash S : \varphi}\ (r)$$

が局所健全であるとは、

$$\bigwedge_{i=1}^{n} \models S_i : \varphi_i \quad \Longrightarrow \quad \models S : \varphi$$

が成り立つことをいう。

本論文では、この対偶を用いる。すなわち、結論に反例トレース

$$\tau \in [\![S]\!] \setminus \|\varphi\|$$

が存在するならば、ある $i \in \{1, \ldots, n\}$ とトレース τ_i が存在して、

$$\tau_i \in [\![S_i]\!] \setminus \|\varphi_i\|$$

が成り立つことを示す。この操作を、結論の反例を前提へ持ち上げるという。

補題 5.1 (局所健全性)**.** 循環証明体系の各推論規則は局所健全である。とくに、Call 規則について、結論の反例トレースを τ、前提へ持ち上げられた反例トレースを τ' とすると、

$$|\tau'| < |\tau|$$

が成り立つ。

Proof. Seq 、Call 規則について、結論に反例トレースが存在するならば、少なくとも一方の前提にも反例トレースが存在することを示す。

Call

規則

$$\frac{\vdash S_1 : \varphi_1 \quad \vdash S_2 : \varphi_2}{\vdash S_1; S_2 : \varphi_1 \frown \varphi_2}\ \textsc{Seq}$$

を考える。結論の反例トレースを τ とすると、

$$\tau \in [\![S_1; S_2]\!], \qquad \tau \notin \|\varphi_1 \frown \varphi_2\|$$

である。S_1 が異常終了する場合と正常終了する場合に分ける。

S_1 が異常終了する場合、あるトレース τ_0 が存在して、

$$\tau = \tau_0 \cdot \mathsf{abort}, \qquad \tau \in [\![S_1]\!]$$

が成り立つ。この場合、S_2 は実行されない。

ここで、$\tau \in \|\varphi_1\|$ と仮定する。異常終了後には後続の文が実行されないため、$\frown$ の意味論より、

$$\tau \in \|\varphi_1 \frown \varphi_2\|$$

となり、反例の仮定に矛盾する。したがって、

$$\tau \in [\![S_1]\!], \qquad \tau \notin \|\varphi_1\|$$

が成り立つ。よって、τ は左の前提 $\vdash S_1 : \varphi_1$ の反例トレースである。

一方、S_1 が正常終了する場合、$[\![S_1; S_2]\!]$ の定義より、あるトレース τ_1, τ_2 および状態 σ が存在して、

$$\tau = \tau_1 \cdot \sigma \cdot \tau_2,$$
$$\tau_1 \cdot \sigma \in [\![S_1]\!], \qquad \sigma \cdot \tau_2 \in [\![S_2]\!]$$

と分解できる。

ここで、

$$\tau_1 \cdot \sigma \in \|\varphi_1\|, \qquad \sigma \cdot \tau_2 \in \|\varphi_2\|$$

がともに成り立つと仮定する。$\frown$ の意味論より、

$$\tau \in \|\varphi_1\| \frown \|\varphi_2\| = \|\varphi_1 \frown \varphi_2\|$$

となり、反例の仮定に矛盾する。したがって、

$$\tau_1 \cdot \sigma \notin \|\varphi_1\| \quad \text{または} \quad \sigma \cdot \tau_2 \notin \|\varphi_2\|$$

が成り立つ。

前者の場合、$\tau_1 \cdot \sigma$ は左の前提 $\vdash S_1 : \varphi_1$ の反例である。後者の場合、$\sigma \cdot \tau_2$ は右の前提 $\vdash S_2 : \varphi_2$ の反例である。

以上より、Seq 規則の結論に反例トレースが存在するならば、少なくとも一方の前提にも反例トレースが存在する。したがって、Seq 規則は局所健全である。

Call

$m\{S_m\} \in T$ とし、規則

$$\frac{\vdash S_m : \varphi_m}{\vdash m() : Id \frown \varphi_m}\ \textsc{Call}$$

を考える。結論の反例トレースを τ とすると、

$$\tau \in [\![m()]\!], \qquad \tau \notin \|Id \frown \varphi_m\|$$

である。

手続き呼び出しの操作的意味論より、あるトレース $\widehat{\tau}$ と状態 σ_0 が存在して、

$$\tau = \sigma_0 \cdot \widehat{\tau}, \qquad \widehat{\tau} \in [\![S_m]\!]$$

と書ける。ここで、$\widehat{\tau}$ の先頭状態も σ_0 である。

$\widehat{\tau} \in \|\varphi_m\|$ と仮定すると、$\sigma_0 \cdot \sigma_0 \in \|Id\|$ であるため、

$$\tau = (\sigma_0 \cdot \sigma_0) \frown \widehat{\tau} \in \|Id \frown \varphi_m\|$$

となり、反例の仮定に矛盾する。したがって、

$$\hat{\tau} \in [\![S_m]\!], \qquad \hat{\tau} \notin \|\varphi_m\|$$

であり、$\hat{\tau}$ は前提の反例トレースである。

また、τ は $\hat{\tau}$ に呼び出し時の状態を一つ加えたトレースであるため、

$$|\tau| = |\hat{\tau}| + 1$$

である。したがって、

$$|\hat{\tau}| < |\tau|$$

が成り立つ。 □

定理 5.2 (健全性)**.** 任意の文 S とトレース論理式 φ について、

$$\vdash S : \varphi \implies \models S : \varphi$$

が成り立つ。

Proof. 背理法により示す。$\vdash S : \varphi$ の循環証明が存在するにもかかわらず、$\not\models S : \varphi$、すなわち

$$[\![S]\!] \not\subseteq \|\varphi\|$$

であると仮定する。このとき、

$$\sigma \in [\![S]\!] \qquad \text{かつ} \qquad \sigma \notin \|\varphi\|$$

を満たす反例トレース σ が存在する。

先に示した各推論規則の局所健全性より、結論に反例トレースが存在するならば、少なくとも一つの前提に反例トレースが存在する。したがって、結論の反例を推論規則に沿って前提へ持ち上げることができる。

ここで、反例の追跡が循環辺を通って無限に継続すると仮定する。循環証明に課された大域的トレース条件より、その経路上には手続き呼び出しの地点が無限回現れる。したがって、反例トレースの長さについて結論の反例トレースを σ、前提へ持ち上げた反例トレースを σ' とすると、

$$|\sigma'| < |\sigma|$$

という自然数の無限下降列が生じる。しかし、有限トレースの長さは自然数であるため、このような無限下降列は存在しない。

よって、反例の追跡が循環し続けることはなく、最終的に導出の葉に置かれた公理へ到達する。しかし、各公理の結論は意味論的に妥当であり、公理の結論には反例トレースが存在しない。これは、公理まで持ち上げられた反例トレースが存在することと矛盾する。

したがって、反例トレースは存在せず、

$$[\![S]\!] \subseteq \|\varphi\|$$

が成り立つ。 □

5.2 相対完全性

定理 5.3 (相対完全性)**.** 任意の文 S とトレース論理式 φ について、

$$\models S : \varphi \implies \vdash S : \varphi$$

が成り立つ。

Proof. $\models S : \varphi$、すなわち、

$$[\![S]\!] \subseteq \|\varphi\|$$

であると仮定する。

定理 4.1 より、

$$\vdash S : \mathrm{stf}(S) \tag{*}$$

が成り立つ。また、定理 4.2 より、

$$[\![S]\!] = \|\mathrm{stf}(S)\|$$

が成り立つ。したがって、仮定より、

$$\|\mathrm{stf}(S)\| \subseteq \|\varphi\|$$

である。すなわち、

$$\mathrm{stf}(S) \models \varphi \tag{**}$$

が成り立つ。

よって、(∗)、(∗∗) および Cons 規則より、

$$\frac{\vdash S : \mathrm{stf}(S) \quad \mathrm{stf}(S) \models \varphi}{\vdash S : \varphi}$$

を得る。 □

6 まとめと今後の課題

本研究では、再帰プログラムの実行トレースの性質を証明するためのトレース論理を、動的メモリ操作を含むプログラムに対して拡張した。具体的には、メモリ確保、読み出し、書き込みおよび解放に対応する関係記号と、異常終了を表す関係記号 Ab を導入し、トレース論理上の意味を定義した。これに伴って、各プログラム S の実行トレースを特徴付ける最強トレース論理式 $\mathrm{stf}(S)$ を拡張した。これにより、

$$[\![S]\!] = \|\mathrm{stf}(S)\|$$

が成り立ち、プログラムの実行トレースをトレース論理式として表現できる。さらに、$\mathrm{stf}(S)$ が Ab-free な論理式を含意する，すなわち，$\mathrm{stf}(S) \models \varphi$ なる

Ab-free な論理式 φ が存在することを示すことで、S の実行トレースが異常終了を含まないことを示すことができる。

加えて、プログラムとトレース論理式の関係を判断 $S : \varphi$ として表す循環証明体系を定義した。本証明体系について、導出可能な判断が意味論的に妥当であることを表す健全性と、意味論的に妥当な判断が導出可能であることを表す相対完全性を示した。

本研究では、トレース論理式間のエンテイルメントを正しく判定できることを仮定している。そのため、今後の課題として、エンテイルメント判定の自動化と、そのための証明体系の構築が挙げられる。Heidler ら [3] は、オリジナルのトレース論理におけるエンテイルメント判定のためのシーケント計算を提案している。また、Sato ら [5] は、最大不動点を導入して拡張したトレース論理に対して、エンテイルメント判定のための循環証明体系を提案している。今後は、これらの証明体系を、本研究で導入した動的メモリ操作および異常終了を含む拡張トレース論理に適用できるよう拡張し、エンテイルメント判定の自動化を目指す。

参 考 文 献

[1] Brotherston, J., Bornat, R., and Calcagno, C.: Cyclic proofs of program termination in separation logic, *ACM SIGPLAN Notices*, Vol. 43, No. 1(2008), pp. 101–112.

[2] Gurov, D. and Hähnle, R.: An Expressive Trace Logic for Recursive Programs, *10th International Conference on Formal Structures for Computation and Deduction (FSCD 2025)*, LIPIcs, Vol. 337, 2025, pp. 21:1–21:22.

[3] Heidler, N. and Hähnle, R.: A Sequent Calculus For Trace Formula Implication, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2025)*, Vol. 15980, 2026, pp. 473–490.

[4] 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.

[5] Sato, T. and Nakazawa, K.: A cyclic proof system for trace formula implication with least and greatest fixpoints, *12th International Workshop on Rewriting Techniques for Program Transformations and Evaluation (WPTE 2026)*, 2026.