日本ソフトウェア科学会第 27 回大会 (2010 年度) 講演論文集

分離論理によるポインタープログラム検証の完全性

龍田 真　Wei-Ngan Chin　Mahmudul Faisal Al Ameen

本論文は，Reynolds によるポインタープログラム検証のための分離論理体系に対して，その完全性定理および表現性定理を証明する．

1 序論

while プログラムに対するプログラム検証はよく研究されてきた [1]．しかし，ポインタープログラムの検証は, assertion 付きのポインタープログラム に対するよい論理体系の設計が難しかったため，まだ十分には調べられていない．Reynolds は分離論理を用いてよい論理体系を与えることに成功した [7]．これにより，プログラムの性質を簡潔に表すことができ，また検証の証明を実用的に行うことができるようになった．分離論理は理論的にも実際的にもうまくいっている．この結果に基づいて，いくつかのポインタープログラム検証器が実装されている [6] [2]．例えば，[6] のシステムは，クイックソートのプログラムの正しさを自動的に 1 秒間で証明する．

証明システムに関するもっとも重要な理論的問題の一つは，その完全性である．健全性は，検証システムがプログラムが正しいということを証明すれば，そのプログラムが意味上も正しいことを保障する．健全性は，検証システムの意義を保障する．多くの実装システムに関してその健全性が証明されている．しかし，このことは，すべての正しいプログラムに対して，それが正しいことがこのシステムで証明できることは意味しない。正しいけれども検証できないプログラムがあるかも知れない．

完全性は，健全性の逆である．完全性は，もしプログラムが意味上正しいならば，検証システムがそのプログラムの正しさを必ず証明できることを，保障する．完全性は，検証システムがどれほど強力であるかを表す．一般にはシステム全体に関する完全性は成り立たないことが多い．しかし，このような場合でも，システムのコアの部分についてある条件下で完全性が証明できれば，このシステムの限界や改良方向を知ることができる点で，完全性は重要である．

本論文の貢献は，(1) 分離論理とポインタープログラムに対する完全性定理を証明すること，(2) ペアノ算術，分離論理，ポインタープログラムに対する表現性定理を証明すること，である．

完全性は，while プログラムに対する完全性の証明 [1] をポインタープログラムと分離論理に拡張することによってなされる．本論文は，Reynolds の後方推論の論理体系を扱う．もっとも難しい点は，表現性定理の証明である．

本論文の表現性定理は，標準解釈においてペアノ算術と分離論理がポインタープログラムに対して表現的であること，つまり，任意のポインタープログラムに対する最弱事前条件がペアノ算術と分離論理により表現できることを主張する．この結果は，ヒープと

Completeness for Verification of Pointer Program by Separation Logic.

Makoto Tatsuta, 国立情報学研究所, National Institute of Informatics.

Wei-Ngan Chin, , National University of Singapore.

Mahmudul Faisal Al Ameen, 総合研究大学院大学, Graduate University for Advanced Studies.

store の情報を自然数でコード化し，プログラムの実行と assertion の真偽をペアノ算術で模倣することにより得られる．表現可能性定理は，一見すると自明のように見えるが，これは本当は微妙な問題であり，少しでも体系が異なると病的な反例がいくつも知られている [3].

表現性定理の証明の鍵は，現在のヒープを正確に表す assertion が構成できることである．与えられたヒープ h をコード化して表す自然数 m を定めることができる．Heap(m) を，現在の state (s, h) において自然数 m が heap h をコード化により表すときに真となるように，定めることができる．

本論文では相対完全性のみを証明する．相対完全性とは，すべての assertion の真偽は所与であるという条件の下での asserted program の推論に関する完全性のことである．while プログラム検証において，相対的でない完全性は成り立たず，相対完全性が証明された [4] のと同じ理由で，ポインタープログラム検証でも，相対的でない完全性は成り立たず，相対完全性が期待できる最善の完全性である．

完全性証明の一部分はすでに [5] で論じられている．その論文では各後方推論規則に対してその中の事前条件が最弱事前条件であることを示している．しかし，これだけであるため，その論文が，複文と while 文を含まないプログラムのみに対する完全性を証明しており，プログラム全体に対する完全性は証明していない．

本論文は，本論文と同じ著者による国際会議論文 [8] に基づいている．

2 while プログラム

まず，既知である While プログラムの完全性を説明する．

式 e を次のように定める．x は変数である．

$e ::= x|0|1|e + e|e \times e.$

この式に基づくペアノ算術を以下では考える．

boolean expression b をペアノ算術の quantifier のない論理式と定める．例えば，$x < 11$ は boolean expression である．

while プログラム P の構文は次のように定義される．

$P ::= x := e|\text{if } (b) \text{ then } (P) \text{ else } (P)|$
$\text{while } (b) \text{ do } (P)|P; P$

例えば次は while プログラムである．

$\text{while } (x < 11) \text{ do } (y := y + x; x := x + 1)$

while プログラム P の意味 $[\![P]\!]$ は次のように定義される．store s を変数から自然数への関数と定める．これは，変数に格納されている値を表す．プログラムの意味 $[\![P]\!]$ を $[\![P]\!](s_1) = s_2$ により定める．

例えば，$s_1(x) = 1, s_1(y) = 0, s_2(x) = 11, s_2(y) = 55$ であるとき，
$[\![\text{while } (x < 11) \text{ do } (y := y + x; x := x + 1)]\!](s_1) = s_2$ である．

assertion A はペアノ算術の論理式と定める．例えば，$x = 11 \wedge y = 55$ は assertion である．

assertion の真偽は標準モデルで考える．

式 $\{A\}P\{B\}$ を asserted program とよぶ．その意味は，真偽の値をとる．$\{A\}P\{B\}$ が真とは，A を満たす状態のとき，プログラム P を実行し，そのプログラムが停止すれば，停止した状態のとき B が成り立つことである．例えば，$\{x = 1 \wedge y = 0\}\text{while } (x < 11) \text{ do } (y := y + x; x := x + 1)\{x = 11 \wedge y = 55\}$ は真である．

ホーア論理は asserted program を推論する．ホーア論理は次の推論規則で与えられる．

$$\frac{}{\{A[x := e]\}x := e\{A\}} \ (assignment)$$

$$\frac{\{A \wedge b\}P_1\{B\} \quad \{A \wedge \neg b\}P_2\{B\}}{\{A\}\text{if } (b) \text{ then } (P_1) \text{ else } (P_2)\{B\}} \ (if)$$

$$\frac{\{A \wedge b\}P\{A\}}{\{A\}\text{while } (b) \text{ do } (P)\{A \wedge \neg b\}} \ (while)$$

$$\frac{\{A\}P_1\{C\} \quad \{C\}P_2\{B\}}{\{A\}P_1; P_2\{B\}} \ (comp)$$

$$\frac{\{A_1\}P\{B_1\}}{\{A\}P\{B\}} \ (conseq) \quad (A \to A_1,\ B_1 \to B \text{ 真})$$

$\{A\}P\{B\}$ が証明可能であるとはこれを結論とする証明図が存在することと定義される．例えば，

$\{x = 1 \wedge y = 0\}$while $(x < 11)$ do $(y := y + x; x := x + 1)\{x = 11 \wedge y = 55\}$ は証明可能である．図 1 の証明図があるからである．ここで，ループ不変量を表す assertion I を $x \leq 11 \wedge P(y, x - 1)$ とおいた．ここで $P(y, x)$ とは $y = 1 + \ldots + x$ を意味する assertion であり，これはペアノ算術として構成できる．$x = 1 \wedge y = 0 \rightarrow I$ は真である．また，$I \wedge x \geq 11 \rightarrow x = 11 \wedge y = 55$ も真である．

健全性とは，$\{A\}P\{B\}$ が証明可能であれば $\{A\}P\{B\}$ が真であることを指す．健全性は，検証システムがプログラムが正しいことを証明すれば，そのプログラムは意味上も正しいことを保障する．健全性は検証システムの意義を保障する．

完全性とは，$\{A\}P\{B\}$ が真であれば，$\{A\}P\{B\}$ が証明可能であることを指す．完全性は，もしプログラムが意味上正しいならば，検証システムがそのプログラムの正しさを必ず証明できることを保障する．完全性は健全性は逆である．完全性は検証システムの能力が十分強力であることを表す．

ホーア論理は，健全であり完全でもあることが証明されている [4] [1].

3 ポインタープログラム

ポインタープログラムは，while プログラムにヒープとよばれるメモリ領域を追加したプログラム言語であり，次のように定義される．

$P ::= x := e|$if (b) then (P) else $(P)|$
while (b) do $(P)|P; P|$
$x := \text{cons}(e, e)|x := [e]|[e] := e|\text{dispose}(e)$

例で説明しよう．$x := \text{cons}(0, 3)$ は allocation とよばれ，新しい連続した二つのメモリセルをヒープに割り当て，それらにそれぞれ 0 と 3 を入れ，一番目のメモリセルの番地を変数 x に格納する．$x := [101]$ は lookup とよばれ，101 番地のメモリセルの内容を変数 x に格納する．$[100] := 2$ は mutation とよばれ，100 番地のメモリセルに 2 を格納する．dispose(100) は dispose とよばれ，100 番地のメモリセルを解放しそれをフリーリストに戻す．

ポインタープログラムの意味を定義しよう．N により自然数全体を現す．自然数は値と番地の両方を表す．$\text{Locs} = \{n \in N | n > 0\}$ と定める．0 はヌルポインターを表す．

store s は変数から N への関数として定義される．heap h は Locs から N の有限部分関数として定義される．state は (s, h) であると定義する．

プログラムの実行は次の三つの場合がある．(1) 正常に停止する．(2) abort をともなって停止する．これはメモリエラーを表す．(3) 停止しない．

abort は割り当てられていないメモリセルの番地 e を $x := [e]$, $[e] := e_2$, または dispose(e) により使用したときに発生する．

プログラム P の意味 $[\![P]\!]$ は，state 全体から state の集合全体への関数として定義される．初期 state (s, h) であり，プログラム P の実行が正常停止したとき，$P((s, h))$ は実行後にあり得るすべての state 全体である．プログラムが abort をともなって停止するとき，$P((s, h)) = \{\text{abort}\}$ である．非決定性は，$x := \text{cons}(e_1, e_2)$ のとき，新しいメモリセルの選び方により発生する．

Dom(h) により関数 h の定義域を表す．$[\![e]\!]_s$ は式 e を state s の下で評価した結果の値を表す．$[\![b]\!]_s$ は boolean expression b を state s の下で評価した結果の値を表す．$[\![P]\!]$ は次のように定義される．

$$\dfrac{\dfrac{\vdots}{\{I \wedge x < 11\} y := y + x; x := x + 1 \{I\}}}{\dfrac{\{I\}\text{while } (x < 11) \text{ do } (y := y + x; x := x + 1)\{I \wedge x \geq 11\}}{\{x = 1 \wedge y = 0\}\text{while } (x < 11) \text{ do } (y := y + x; x := x + 1)\{x = 11 \wedge y = 55\}}\ (conseq)}\ (while)$$

図 1　証明図の例

$[\![P]\!](\text{abort}) = \{\text{abort}\}$,
$[\![x := e]\!]((s,h)) = \{(s[x := [\![e]\!]_s], h)\}$,
$[\![\text{if } (b) \text{ then } (P_1) \text{ else } (P_2)]\!]((s,h)) =$
　$[\![P_1]\!]((s,h))$ if $[\![b]\!]_s = \text{true}$,
　$[\![P_2]\!]((s,h))$ otherwise,
$[\![\text{while } (b) \text{ do } (P)]\!]((s,h)) = \{(s,h)\}$
　if $[\![b]\!]_s = \text{false}$,
　$[\![\text{while } (b) \text{ do } (P)]\!]([\![P]\!]((s,h)))$ otherwise,
$[\![P_1; P_2]\!]((s,h)) = [\![P_2]\!]([\![P_1]\!]((s,h)))$,
$[\![x := \text{cons}(e_1, e_2)]\!]((s,h)) =$
　$\{(s[x := n], h[n := [\![e_1]\!]_s, n+1 := [\![e_2]\!]_s]) |$
　$n > 0, n, n+1 \notin \text{Dom}(h)\}$,
$[\![x := [e]]\!]((s,h)) =$
　$\{(s[x := h([\![e]\!]_s)], h)\}$ if $[\![e]\!]_s \in \text{Dom}(h)$,
　$\{\text{abort}\}$ otherwise,
$[\![[e_1] := e_2]\!]((s,h)) =$
　$\{(s, h[[\![e_1]\!]_s := [\![e_2]\!]_s])\}$ if $[\![e_1]\!]_s \in \text{Dom}(h)$,
　$\{\text{abort}\}$ otherwise,
$[\![\text{dispose}(e)]\!]((s,h)) =$
　$\{(s, h|_{\text{Dom}(h) - \{[\![e]\!]_s\}})\}$ if $[\![e]\!]_s \in \text{Dom}(h)$,
　$\{\text{abort}\}$ otherwise.

4　分離論理

本論文では Reynolds [7] による分離論理を扱う.

ポインタープログラムに対する assertion A は次のように定義される.

$A ::= \text{emp} | e = e | e < e | e \mapsto e | \neg A | A \wedge A | A \vee A | A \rightarrow A | \forall x A | \exists x A | A * A | A \mathrel{-\!\!*} A$

意味を例で説明する．emp は，現在のヒープが空であることを表す．$3 \mapsto 5$ は，現在のヒープが一つのメモリセルだけから成り，それは 3 番地で，格納されている内容が 5 であることを表す.

$A * B$ は，現在のヒープが二つのヒープ h_1, h_2 に分割でき，A が h_1 で成り立ち，B が h_2 で成り立つことを表す.

$A \mathrel{-\!\!*} B$ は，現在のヒープと共通部分のない任意のヒープ h に対して，A がそこで成り立つなら，h と現在のヒープを合わせたヒープにおいて B が成り立つことを表す.

分離論理とポインタープログラムに関する asserted program に対する推論規則は次である．FV は自由変数の集合を表す.

$$\dfrac{(x' \notin \text{FV}(e_1, e_2, A))}{\{\forall x'((x' \mapsto e_1, e_2) \mathrel{-\!\!*} A[x := x'])\}\ x := \text{cons}(e_1, e_2)\{A\}}\ (cons)$$

$$\dfrac{(x' \notin \text{FV}(e, A))}{\{\exists x'(e \mapsto x' * (e \mapsto x' \mathrel{-\!\!*} A[x := x']))\}\ x := [e]\{A\}}\ (lookup)$$

$$\dfrac{(x \notin \text{FV}(e_1))}{\{(\exists x(e_1 \mapsto x)) * (e_1 \mapsto e_2 \mathrel{-\!\!*} A)\}\ [e_1] := e_2\{A\}}\ (mutation)$$

$$\dfrac{(x \notin \text{FV}(e))}{\{(\exists x(e \mapsto x)) * A\}\text{dispose}(e)\{A\}}\ (dispose)$$

これらの推論規則と while プログラムに対する推論規則を合わせて $\{A\}P\{B\}$ を結論とする証明図が存在するとき，$\{A\}P\{B\}$ が証明可能であるという.

5　完全性定理

定理 1 (完全性) $\{A\}P\{B\}$ が真であるならば，$\{A\}P\{B\}$ は証明可能である.

完全性定理の証明のアイデアとして，ホーア論理の完全性証明を拡張すること，相対完全性を証明目標とすること，を用いる．相対完全性は，ホーア論理の完全性定理でも使われた概念であり，assertion の真偽に関しては標準モデルでその真偽を定めることとし，その証明可能性は考えないで，asserted program の証明可能性だけ対象とする完全性である．assertion

の真偽は次の推論規則 (conseq) において使われている．

$$\frac{\{A_1\}P\{B_1\}}{\{A\}P\{B\}}\ (conseq)\quad (A \to A_1,\ B_1 \to B \text{ 真})$$

また，本論文では最弱事前条件を用いる．与えられたプログラム P と assertion B に対して，その最弱事前条件とは，その条件を満たす初期状態から P を実行し正常停止すれば B が成り立つようなもっとも弱い条件である．

完全性定理の証明の難しさは，表現性定理にある．表現性定理とは，与えられたプログラム P と assertion B に対してその最弱事前条件を表す assertion が存在する，というものである．

6 完全性定理の証明

(n, m) を，自然数 n, m の対を表す自然数とする．$\langle n_1, \ldots, n_k\rangle$ を，自然数列 $n_1, \ldots, n_k$ を表す自然数とする．Lookup(x, y, z) により，列 x が対 (y, z) を含んでいることを表す．Lookup(x, l, k) は，$x = \langle n_1, \ldots, (l, k), \ldots, n_m\rangle$ であることを意味する．

Heap を次のように定義する．これが本論文の完全性定理証明の鍵である．

$$\mathrm{Heap}(m) = \forall xy(\mathrm{Lookup}(m, x, y) \leftrightarrow (x \mapsto y{*}0 = 0)).$$

Heap$(\langle (l_1, n_1), \ldots, (l_k, n_k)\rangle)$ は，現在のヒープ h について，$\mathrm{Dom}(h) = \{l_1, \ldots, l_k\}$ であり，かつ $h(l_i) = n_i$ であることを意味する．

Heap を用いて表現性定理が証明できる．ペアノ算術の基本技法により，次の二つの assertion がある．

現在のヒープ s に対して，$\mathrm{Store}_{x_1,\ldots,x_n}(\langle m_1, \ldots, m_n\rangle)$ を $s(x_i) = m_i$ を表す assertion とする．$\lceil (s, h)\rceil$ により $(\lceil s\rceil, \lceil h\rceil)$ を指す．これは，(s, h) のコードを意味する．ベクトル記法 $\overrightarrow{x} = x_1, \ldots, x_n$ を用いる．

$\mathrm{Exec}_{P,\overrightarrow{x}}(\lceil r_1\rceil, \lceil r_2\rceil)$ を $[\![P]\!](r_1) \ni r_2$ を表す assertion とする．ここで $\overrightarrow{x}$ は P の自由変数をすべて含むとする．

ここで，最弱事前条件を表す assertion $\mathrm{W}_{P,A}$ を次のように定義する．

$$\begin{aligned}\mathrm{W}_{P,A}(\overrightarrow{x}) = \forall xyzw(&\mathrm{Store}_{\overrightarrow{x}}(x) \wedge \mathrm{Heap}(y) \wedge\\ &\mathrm{Pair2}(z, x, y) \wedge \mathrm{Exec}_{P,\overrightarrow{x}}(z, w) \to w > 0 \wedge\\ &\exists y_1 z_1(\mathrm{Pair2}(w, y_1, z_1) \wedge \mathrm{Eval}_{A,\overrightarrow{x}}(y_1, z_1))).\end{aligned}$$

ただし，$\overrightarrow{x}$ は P と A のすべての自由変数を含むとする．また，Pair2(z, x, y) は $z = (x, y)$ を意味する assertion である．

定理 2 (表現性) $W_{P,A}(\overrightarrow{x})$ は P と A の最弱事前条件を表す．

この定理は P に関する帰納法により証明できる．

完全性定理は，表現性定理を用いて，次のように証明される．

定理 1 の証明. P に関する帰納法で証明する．P の形で場合分けする．

P が $P_1; P_1$ の場合．$\{A\}P_1; P_2\{B\}$ が真であると仮定する．C を $\mathrm{W}_{P_2,B}(\overrightarrow{x})$ とおく．ここで $\overrightarrow{x}$ は B, P_2 のすべての自由変数を含むとする．表現性定理より，$\{A\}P_1\{C\}$ と $\{C\}P_2\{B\}$ は真である．P_1 と P_2 に対する帰納法の仮定より，$\{A\}P_1\{C\}$ は $\{C\}P_2\{B\}$ は証明可能である．推論規則 $(comp)$ より，$\{A\}P_1; P_2\{B\}$ は証明可能である．

他の場合も同様に証明できる．表現性定理は，$\{A\}$while (b) do $(P)\{B\}$ の場合にも用いる．□

参 考 文 献

[1] K.R. Apt, Ten Years of Hoare's Logic: A Survey — Part I, *ACM Transactions on Programming Languages and Systems* 3 (4) (1981) 431–483.

[2] J. Berdine, C. Calcagno, and P.W. O'Hearn, Symbolic Execution with Separation Logic, In: Proceedings of the Third Asian Symposium on Programming Languages and Systems (APLAS2005), *Lecture Notes in Computer Science* 3780 (2005) 52-68.

[3] J.A. Bergstra and J.V. Tucker, Expressiveness and the Completeness of Hoare's Logic, *Journal Computer and System Sciences* 25 (3) (1982) 267–284.

[4] S.A. Cook, Soundness and completeness of an axiom system for program verification, *SIAM Journal on Computing* 7 (1) (1978) 70–90.

[5] S. Ishtiaq and P.W. O'Hearn, BI as an Assertion Language for Mutable Data Structures, In: *Proceedings of 28th ACM Symposium on Principles of Programming Languages (POPL2001)* (2001) 14–26.

[6] H. H. Nguyen, C. David, S.C. Qin, and W.N.

Chin, Automated Verification of Shape and Size Properties Via Separation Logic, In: Proceedings of 8th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2007), *Lecture Notes in Computer Science* 4349 (2007) 251–266.

[7] J.C. Reynolds, Separation Logic: A Logic for Shared Mutable Data Structures, In: *Proceedings of Seventeenth Annual IEEE Symposium on Logic in Computer Science (LICS2002)* (2002) 55–74.

[8] M. Tatsuta, W.N. Chin, and M.F. Al Ameen, Completeness of Pointer Program Verification by Separation Logic, In: Proceeding of 7th IEEE International Conference on Software Engineering and Formal Methods (SEFM 2009) 179–188.