状態に依存したプログラムの合成

1J－4

高杉秀樹, 西谷泰昭
群馬大学工学部

1 はじめに

Manna and Waldinger[1,2]は、situational logic の論理式で与えられた仕様からの deductive tableau を用いたプランの導出を試みている。本稿では、その手法を用い、簡単な nonapplicative program である swap を例として、その合成を考える。アクションの効果を公理として与え、出力仕様を定理として証明する過程で、プログラムはアクションの有限列として後向きに導出される。

2 deductive tableau による定理証明

2.1 situational logic による述語表現

はじめに、situational logic の項を次のように定義する。(1) 状態項: 状態を表す (2) プラン項: プランを表す (3) オブジェクト項: 状態が変化しても変わらない項, 定数など (4) オブジェクト変項: 状態から状態へ変わることが許されている。

定義1 一階述語論理の式をPとしたとき、situational logic の式を$P@w$とする。ここで、wは状態項である。$P@w$は、状態wでPが真のとき真である。

例えば、$(x=1)@w$は、状態wで $x=1$ が成り立つとき真である。

定義2 sが状態項で、pがプラン項なら、$s;p$は、状態sでpを実行した結果生じた状態である。

例えば、$s_0; x := y+2$ は、状態s_0でxに$y+2$の値を代入した結果の状態である。

公理3 アクションが存在しないということに対応する項を空プランΛとしたとき、任意の状態wに対して "$w;\Lambda = w$" が仮定できる。また、任意のプラン p に対して "$p;\Lambda = \Lambda;p = p$" である。

2.2 仕様

出力仕様を充足するプログラムを作るためには, それが成立する状態が存在することを証明しなければならない。入力をa, 初期状態をs_0, プログラムをzとしたとき, 出力条件は$Q(a,s_0,s_0;z)$となる。プログラムzを作るためには$\forall a \exists z \forall s_0 Q(a,s_0,s_0;z)$を証明する。証明の際には, スコーレム化した$Q(a,s_0(z),s_0(z);z)$を考える。ここで, aは定数,s_0はスコーレム関数,zは変数である。また$s_0(z)$は単にs_0と表すものとする。

2.3 deductive tableau

deductive tableau は、situational logic の文を含む行の集合であり、文は、*assertion* あるいは *goal* と呼ばれる。goal である文は *plan entry* を持つ。文は quantifier-free と仮定できる。

最初の tableau は次のようになる。

assertions	goals	plan
	$Q(a,s_0,s_0;z)$	z

公理や定理のような妥当な文は、assertion として tableau に入れられる。plan entry は与えられた定理の証明でプログラムを導くためのものであり、導出を通じて以下の性質が成り立つものとする。

あるゴールの例が真ならば、そのとき対応する plan entry の例tは出力条件 $Q(a,s_0,s_0;t)$ を充足する。

2.4 推論規則

ゴール$G[P]$とアサーション$A[P']$を含む tableau を考える。ただし、それらは同じ変数を含んではいけない。P,P'は論理式である。unifier θによってPとP'が単一化可能であるとき、Murray[3]の極戦略によりPに正の極、P'に負の極が割り当てられるなら、それぞれ *true*,*false* で書き換え、次のように導出を行える。

assertions	goals
	$G[P]$
$A[P']$	
	$G\theta[true] \wedge \neg A\theta[false]$

3 swap プログラムの導出

入出力仕様を次のように与える。

入力仕様: $(x=X \wedge y=Y)@s_0$

出力仕様: $(x=Y \wedge y=X)@s_0;z$

ここで、x,yはプログラム変数、X,Yは定数、s_0は初期状態、zは合成されるべきプログラムである。

出力仕様を定理とすると、最初の tableau は次のようになる。

assersions	goals	plan
	$(x=Y \wedge y=X)@s_0;z$	z

---

[0]A synthesis of programs depended on states
Hideki Takasugi, Yasuaki Nishitani
Faculty of Engineering,Gunma University

公理として次の tableau を加える。

$P^u_t@w \supset P@w; u := t$			(1)

この公理は、状態 wで, プログラム変数に対応する変数 uを tで置き換えたPが成り立つなら、wでuにtの値を代入した結果の状態でも論理式Pが成り立つことを述べている。

Pを$(x = Y \wedge y = X)$として、unifier $\{u \leftarrow x, w \leftarrow s_0; z_1, z \leftarrow z_1; x := t\}$で単一化し、上の二つの tableau で導出を行うと、新しいゴールを含む次のような tableau が得られる。

	$(t_1 = Y \wedge y = X)@s_0; z_1$	$z_1; x := t_1$

次に上の公理を用い同様に導出を行うと、

	$(t_1 = Y \wedge$ $t_2 = X)@s_0; z_2$	$z_2; y := t_2; x := t_1$	(2)

unifier$\{t_1 \leftarrow y, t_2 \leftarrow x, z_2 \leftarrow \Lambda\}$を適用すると、

	$(y = Y \wedge x = X)@s_0$	$y := x; x := y$

この tableau は、ゴールが真になることが導き出せるように見えるが、プランは正しくない。これは、変項であるt_1が "$y := t_2$" の副作用を受けるためである。

そこで、副作用の影響を記述するために変項 vに$\binom{u}{t}$という属性を付加し、$(v)^u_t$で表す。すなわち、公理(1)を使用したときは、全ての変項を$(v)^u_t$とする。従って、(2)は次のようになる。

	$((t_1)^y_{t_2} = Y \wedge$ $t_2 = X)@s_0; z_2$	$z_2; y := t_2; x := t_1$	(2′)

ここでオブジェクト変項$v' = (\cdots((v)^{u_1}_{t_1})^{u_2}_{t_1}\cdots)^{u_n}_{t_n}$の単一化について以下の戦略を考える。

(a) v'をvとして unifier を求めv'の置換が、すべてのu_iを含まないときのみ導出を行う。従って、(2′)はこの導出ができない。

(b) (a)の導出ができないとき、新しいプログラム変数$\overline{v}$を導入し、$v'(v)$を$\overline{v}$で置換する。よって、(2′)より次が導かれる。

	$(\overline{t_1} = Y \wedge$ $t_2 = X)@s_0; z_2$	$z_2; y := t_2; x := \overline{t_1}$

この tableau と公理(1)より導出を行うと、新しい tableau は、

	$(t_3 = Y \wedge$ $(t_2)^{\overline{t_1}}_{t_3} = X)@s_0; z_3$	$z_3; \overline{t_1} := t_3;$ $y := t_2; x := \overline{t_1}$

この tableau と入力仕様を unifier$\{t_3 \leftarrow y, t_2 \leftarrow x, z_3 \leftarrow \Lambda\}$によって単一化し導出を行うと、

	true	$\overline{t_1} := y; y := x; x := \overline{t_1}$

となり、ゴールが真になることが導き出せる。プログラム "$\overline{t_1} := y; y := x; x := \overline{t_1}$" の実行は出力仕様を充足する。

4 rotation プログラムの導出

rotation プログラムの入出力仕様は次のように表される。

入力仕様:n$(x = X \wedge y = Y \wedge z = Z)@s_0$
出力仕様:$(x = Y \wedge y = Z \wedge z = X)@s_0; p$

swap と同様に公理(1)を三回用いて導出を行うと、次のような tableau が得られる。

	$(((t_1)^z_{t_2})^y_{t_3} = Y \wedge$ $t_3 = Z \wedge$ $(t_2)^y_{t_3} = X)@s_0; p_3$	$p_3; y := t_3;$ $z := t_2;$ $x := t_1$	(3)

入力仕様との単一化を考えると、t_2, t_3については、$\{t_2 \leftarrow x, t_3 \leftarrow z\}$で$(a)$の単一化が可能であるが、$t_1$については$\{t_1 \leftarrow y\}$は$(a)$の条件に反する。従って、$(b)$により、プログラム変数を導入しなければならない。導出は次のようになる。

	$(\overline{t_1} = Y \wedge$ $t_3 = Z \wedge$ $(t_2)^y_{t_3} = X)@s_0; p_3$	$p_3; y := t_3;$ $z := t_2;$ $x := \overline{t_1}$
	$(t_4 = Y \wedge$ $(t_3)^{\overline{t_1}}_{t_4} = Z \wedge$ $((t_2)^y_{t_3})^{\overline{t_1}}_{t_4} = X)@s_0; p_4$	$p_4; \overline{t_1} := t_4; y := t_3;$ $z := t_2;$ $x := \overline{t_1}$

この tableau と入力仕様との導出によりプログラム "$\overline{t_1} := y; y := z; z := x; x := \overline{t_1}$" が導かれる。

このプログラムは、最初に公理 (1) をx, z, yの順に適用したが、適用の順序により導入されるプログラム変数の個数が変わる。x, y, zの順に適用した場合には、2個のプログラム変数が導入され、"$\overline{t_2} := z; \overline{t_1} := y; z := y; y := \overline{t_2}; x := \overline{t_1}$"が合成される。変数導入を最小にすることは今後の課題である。

5 おわりに

Manna and Waldinger の deductive tableau を用いて swap プログラムと rotation プログラムの導出を行った。公理(1)の適用によって新しい変項が導入されるが、この変項について間違った単一化がされないように属性を持たせ、その属性により, 導出とプログラム変数の導入を制御した。今後は、一般的にもこの手法が適用できるかについて検討したい。

参考文献

[1] Z. Manna and R. Waldinger, A deductive approach to program synthesis, *ACM TOPLAS,* Vol.2,No.1,Jan.1980,pp.90-121.
[2] Z. Manna and R.Waldinger, Unsolved Problems in the Blocks World, *IPA* 講演会資料,1986
[3] N. V. Murray, Completely nonclausal theorem proving, *Artif. Intel.,* Vol.18,No.1,1982,pp.67-85.