時相論理による段階的仕様記述プロセスに対応した検証法

4J－3

三木 一秀　米崎 直樹
東京工業大学大学 情報理工学研究科 計算工学専攻

1 はじめに

形式的な仕様記述に関する研究が行なわれている。例えば、時相論理によるリアクティブシステムの仕様記述に関する研究がある。その主な目的は、仕様の段階での性質検証である。また一方、仕様記述の計算機支援を行なう研究も行なわれている。多くは、構文エディタやブラウザ等に見られるように、「記述そのもの」に対する支援が主であり、検証のような意味的な支援を記述プロセスと深く関係付けた研究は少ない。

現実の仕様記述過程では、まずプロトタイプを記述し何度も修正を行ないつつ目的となるシステムの仕様を完成させる。仕様を修正する度に、記述された仕様に対する検証結果が得られるならば、仕様誤りの早期発見にもつながり、開発時間の短縮につながると考えられる。これを低い計算コストで実現するためには、仕様が修正される過程と密接に関係した検証方法が必要である。修正前後の仕様の差分と前回の検証から、修正後の仕様の検証を行なう方法を実現することが本研究の目的である。

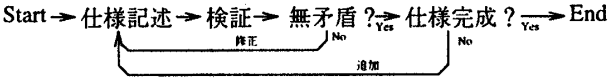
論理式で記述された仕様の様々な性質検証の多くは、論理式の無矛盾性の判定、すなわち充足可能性判定に帰着される。充足可能性判定の方法のひとつにタブロー法があるが、この方法が用いるデータ構造は意味グラフであるため理解しやすく、また、データ構造、アルゴリズムが単純なため計算機による支援に適している。

[1] [2] において、以前の検証に用いたタブローを再利用し効率的なタブロー構成を行なう手続きが示されている。この手続きは、すでに記述されている仕様の中に誤りを発見した際に、その仕様を削除する場合には対応していない。そこで本研究では、より柔軟な仕様の修正方法を考え、仕様を書き加える場合だけでなく誤った仕様を取り除く場合にも、修正前のタブローの再利用が可能なタブローを与える。

Stepwise Verification of Specifications in Temporal Logic
Kazuhide Miki　Naoki Yonezaki
Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
2-12-1, Oookayama, Meguro-ku Tokyo 152, Japan

2 仕様記述プロセス

大きなシステムの仕様を記述する場合には、目標となるシステム全体の仕様を一度に記述することは稀であり、まず基本的な部分を記述し、段階的に修正を加えることで記述を進めてゆく。そこでは、部分的に記述をされた仕様に対して検証を行ない、その結果からさらにどのように仕様を修正、追加するのか判断しながら、仕様を完成させる。



そこで仕様記述プロセスについて以下の仮定をおく。

- 仕様は段階的に記述、修正される。
- 各記述段階において検証を行う。
- 重要な仕様を先に思いつく。
- できるだけ重要ではない仕様を修正する。

3 時相論理

本研究で扱う時相論理をは、様相演算子として until 演算子 [ ] を持つ古典命題論理である。

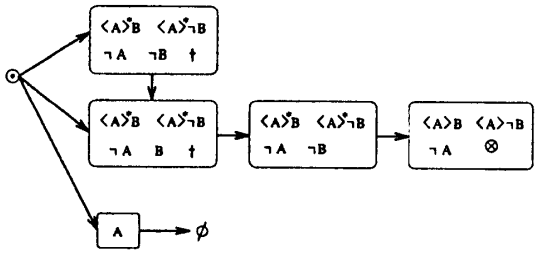
定義 3.1 (式) PV を命題変数の集合とする。$p \in PV$ は式である。A, B が式であるとき、$\neg A, A \wedge B, [A]B$ は式である。また、$A \vee B$, $A \rightarrow B$, $\langle A \rangle B$, $\Box A$, $\Diamond A$ をそれぞれ $\neg(\neg A \wedge \neg B)$, $\neg A \vee B$, $\neg[A]\neg B$, $[\bot]A$, $\langle \bot \rangle A$ の略記とする。ただし $\bot = p \wedge \neg p (\exists p \in PV)$。

定義 3.2 (意味) モデル $M = \langle \langle N, \geq \rangle, s \rangle$、時間 $i \in N$ において式 F が真であることを $M, i \models F$ と表す。until 演算子以外は通常の古典論理と同様な意味を持つ。until 演算子は、$M, i \models [A]B \Leftrightarrow \forall j.(j \geq i) M, j \models B$ または、$\exists k.(M, k \models A$ かつ $\forall .(k > j \geq i) M, j \models B)$ と定義される。いわゆる弱い until である。$\exists M. \exists i. M, i \models A$ のとき A は充足可能。

4 非循環タブロー

ここで与える再利用可能なタブローの基礎として、非循環タブロー [2] を用いる。式 F の非循環タブローは、各頂点に式の集合がラベル付けされた有効グラフであり、F を充足するモデルを表す。以下、特に混乱の可能性のない限り、頂点とその頂点にラベル付けされた式の集合を同一視する。タブロー中の ⊙ はグラフ

の開始点、⊗はそれを含む頂点中の式が成立し続ける循環、†は循環してはいけないことを表す。式Fのタブローにおいて、⊙からϕまたは⊗を含む頂点に到達可能なとき、そのときに限り、Fは充足可能と判定される。例として$[A]\langle A\rangle B \wedge [A]\langle A\rangle \neg B$のタブローを示す。



5 再利用可能なタブローの概要

仕様fの追加 FのタブローTを元に$F \wedge f$のタブローT'を構成する。$\langle b, e\rangle \in T$それぞれに$f$の部分式の成立順序を表す式集合を加え、$T'$の辺とする。このとき頂点に新しく加えられる式には$f$をラベル付けする。

T	$\langle b, e\rangle$
f	$\langle b_1, e_1\rangle, ..., \langle b_n, e_n\rangle$
T'	$\langle b \cup b_1, b \cup e_1\rangle, ..., \langle b \cup b_n, b \cup e_n\rangle$ $\langle b \cup b_1, e \cup e_1\rangle, ..., \langle b \cup b_n, e \cup e_n\rangle$ $\langle e \cup b_1, e \cup e_1\rangle, ..., \langle e \cup b_n, e \cup e_n\rangle$

定義 5.1 (頂点、辺の矛盾) $p(\in PV)$とする。式の集合sは、$p \in s$かつ$\neg p \in s$または、$\otimes \in s$かつ$\dagger \in s$ならば矛盾[1]。頂点vにラベルされている式の集合sが矛盾しているならば、vは矛盾。

頂点が矛盾する場合には、矛盾を表す印をつけてタブロー中に残す。

仕様fの削除 $f \wedge F$のタブローTを元にFのタブローT'を構成する。このときFを充足するモデルのうちfと矛盾するモデルが存在する場合には、それに対応する部分グラフを構成しなければならない場合がある。仕様f_1, f_2, f_3がこの順に追加記述されたのち、仕様f_2を削除するときのタブロー構成を考える。T中からf_2のラベルされた式を取り除く。

f_1	$\langle b_1, e_1\rangle$
f_2	$\langle b_2, e_2\rangle$
f_3	$\langle b_3, e_3\rangle$
T	$\langle b_1 \cup b_2 \cup b_3,$ $e_1 \cup e_2 \cup e_3\rangle$

$\Rightarrow$

f_1	$\langle b_1, e_1\rangle$
f_2	
f_3	$\langle b_3, e_3\rangle$
T'	$\langle b_1 \cup b_3, e_1 \cup e_3\rangle$

しかし、仕様f_2を追加したときに矛盾した辺については仕様f_3を追加したときの処理が行なわれていない。そこで、その部分を構成する必要が生じる。

f_1	$\langle b_1, e_1\rangle$
f_2	$\langle b_2, e_2\rangle$ 矛盾
f_3	
T	矛盾

$\Rightarrow$

f_1	$\langle b_1, e_1\rangle$
f_2	
f_3	???
T'	$\langle b_1 \cup ?, e_1 \cup ?\rangle$

[1] ⊗, †も式として扱う (拡張された式 [2])。

$\langle b, e\rangle \in T$それぞれに対して以下の処理を行なう。$v^f$を$v$中の$f$でラベル付けされた式の集合、$v \backslash f$を$v$に対する$v^f$の補集合$v - v^f$、$ic(v, f)$を$\bigvee_{f' \in v'}$ ($v \backslash f \cup \{f'\}$が矛盾) とする。

- $\neg ic(b, f)$かつ$\neg ic(e, f)$のとき、$\langle b \backslash f, e \backslash f\rangle$を$T'$の辺とする。特に構成するグラフはない。
- $ic(b, f)$または$ic(e, f)$のとき、$b \backslash f \subset b' \backslash f$かつ$e \backslash f \subset e' \backslash f$かつ$\neg ic(b', f)$かつ$\neg ic(e', f)$を満たす辺$\langle b', e'\rangle \in T$が存在するならば、なにもしない。$\langle b', e'\rangle$に対する処理で、ここで構成すべき辺が構成される。
- $b \backslash f \subset b' \backslash f$かつ$e \backslash f \subset e' \backslash f$かつ$\neg ic(b', f)$かつ$\neg ic(e', f)$を満たす辺$\langle b', e'\rangle \in T$が存在しないとき、ある仕様$f'$がラベルされた式のみが$\langle b, e\rangle$と異なる辺を元に必要な辺を構成する。$b \backslash f \backslash f' \subset b' \backslash f \backslash f'$かつ$e \backslash f \backslash f' \subset e' \backslash f \backslash f'$かつ$\neg ic(b', f)$かつ$\neg ic(e', f)$かつ$\neg ic(b', f')$かつ$\neg ic(e', f')$を満たす辺$\langle b', e'\rangle \in T$および$f'$が存在するとき、$b'' = b^{f'} \cup b' \backslash f \backslash f'$が矛盾していない、かつ$e'' = e^{f'} \cup e' \backslash f \backslash f'$が矛盾していないならば、$\langle b'', e''\rangle$を$T'$の辺とする。$b''$または$e''$が矛盾しているなら、矛盾を表す印をつけて$T'$の辺とする。

  2個以上の仕様について、それらがラベルされている式が$\langle b, e\rangle$と異なる辺を元にしても、同様な手続きでT'の辺を構成できる。しかし、そのような仕様の数が増えるにしたがって、処理が複雑になる。
- それ以外の場合、追加の手続きと同様に構成する。これは前項目の処理の最悪の場合である。

6 まとめ

仕様の追加・削除を繰り返すような現実的な仕様記述プロセスにおいて、以前に構成したタブローを再利用することが可能となった。これは、非循環タブローの、充足可能性判定の結果のすべてをグラフとして表現しているという性質による。従来のタブローでは、未来で必ず成立すべき式が成立したかどうかの判定結果がグラフとして表されていなかった。

今後の課題として、再利用可能な部分グラフを探索するアルゴリズムのヒューリスティックの導入による高速化、計算機への実装などが挙げられる。

参考文献

[1] 友石 正彦 , 米崎 直樹. 動作仕様の差分的無矛盾性判定. 日本ソフトウエア科学会 第 10 回大会論文集, pp. 217–220, 1993.

[2] 友石 正彦 , 米崎 直樹. 時相論理による仕様記述の無矛盾性判定のための再利用可能なタブローについて. 電子情報通信学会 信学技報, pp. 37–46, 7 1994.