時相論理による抽象化を用いた検証とその応用*

Verification Using Abstraction by Temporal Logic and Its Applications

山本 光晴 † Mitsuharu YAMAMOTO mituharu@math.s.chiba-u.ac.jp
萩谷 昌己 †† Masami HAGIYA hagiya@is.s.u-tokyo.ac.jp

† 千葉大学理学部 Faculty of Science, Chiba University
†† 東京大学大学院情報理工学系研究科
Information Science and Technology, University of Tokyo

抽象化は状態空間が無限になりうるシステムに対して有限的な検証手法を適用する際に用いられる主要な技術である。我々はこれまで、リンク構造の抽象化に時相論理とその充足可能性を用いる手法について研究してきた。本発表では時相論理による抽象化とその応用について、これまでの結果および現在進行中の研究の経過を述べる。

1 はじめに

我々はこれまで、時相論理による抽象化について研究を行ってきた。抽象化は、状態空間が無限になるようなシステムの解析や検証を、モデル検査 [4] などの有限的手法で行うために有効な手法である。時相論理を用いた抽象化の特徴として、2 方向性を持つ論理を使用することにより逆向きの関係を自然に表現可能なことと、決定可能な論理を使用することにより充足可能性判定を抽象化の自動化に利用可能なことが挙げられる。

本稿では時相論理による抽象化について、その枠組の説明を行い、なぜ抽象化に時相論理を用いるのかについて述べる。また、時相論理による抽象化の応用例についても述べる。

2 具体システムと抽象化

時相論理式による抽象化を用いた検証において我々が対象としているのは、グラフ書き換え系によって表現できるようなシステムである。すなわち、システムの状態全体がラベル付きのノードとリンクから構成されるグラフで表現され、そのグラフがある書き換え規則によって変化していくようなシステムである。以下、抽象化の対象となるシステムを具体システムと呼ぶことにする。

各々の具体システムにおけるノードの数が有限であったとしても、ノードの追加・削除を含む書き換えにおいては、一般には書き換えの結果におけるグラフのノードの数は有界にならない。また、具体システムのノードの数によらず、ある特徴を持ったどのような具体グラフを初期状態とするシステムに対しても、特定の性質が成り立つことを検証したいという場合もある。このような場合、「書き換えによって現れる全ての具体グラフ」や「ある共通した特徴を持った全ての具体グラフ」を列挙することは一般には不可能であるから、抽象化を用いた検証を行うことが考えられる。

ノードと辺にラベルが付いた有向グラフは、自然に Kripke 構造 $\mathcal{M} = (M, \{R_a\}, L)$ とみなすことができる。ここで、M はグラフのノードの集合、$\{R_a\}$ は M 上の関係の集まりで、$s \in M$ から $t \in M$ へラベル a で向かう辺があるとき、かつそのときに限り sR_at である。$L(s)$ は $s \in M$ に付けられているラベルの集合に対応する。

我々の枠組の特徴は空間的特徴を表現するために時相論理式を用いることである。例えば、具体グラフ上でノード s からノード t にラベル a で辺があり、t のラベルが p を含んでいれば、ノード s では $\mathsf{EX}_a p$ が成り立つ。これは sR_at かつ $p \in L(t)$ であれば $\mathcal{M}, s \models \mathsf{EX}_a p$ であるということに対応する。

ここで、Kripke 構造全体がシステムの 1 状態に対応していることに注意されたい。通常、モデル検査などで Kripke 構造 $\mathcal{M} = (M, \{R_a\}, L)$ を考えるときは、M の各点がシステムの 1 状態に対応し、R_a は状態間の遷移関係を表しているが、ここではそうではない。論理としては時相論理を使用しているが、実際に記述する内容はシステムの状態間の時間的特徴ではなく、既に述べたように、システムの 1 状態における空間的特徴である。

抽象ノード は共通した空間的特徴をもつ具体ノー

*本研究は、文部科学省科学研究費補助金，特定領域研究 16016211 の援助を受けている。

ドの集合を表現するものである。既に述べたように空間的特徴は時相論理式によって記述される。すなわち、抽象ノードは与えられた時相論理式の真偽が一致するような具体ノードの集合を表現するのである。抽象化に先立ち、まず時相論理式の有限集合 F を定める。すると、各抽象ノードは F の部分集合 C によって次のように記述できる。

> 抽象ノード C に対応する各具体ノードでは、$\bigwedge\{\phi \mid \phi \in C\} \wedge \bigwedge\{\neg\phi \mid \phi \in F \setminus C\}$ が成り立つ。

以下、抽象ノード C に対して上の論理式を ϕ_C と書くことにする。すなわち、具体グラフ $\mathcal{M}$ におけるノード s が抽象ノード $C \subseteq F$ に抽象化されるとは、$\mathcal{M}, s \models \phi_C$ が成り立つことである。具体ノード s に対して上の条件を満たす $C \subseteq F$ は一意に存在するので、そのような C を $\alpha_{\mathcal{M}}(s)$ と書くことにする。いま F は有限集合であるから、抽象ノードの種類は $2^{|F|}$ で押さえられる。

具体グラフの抽象化は**抽象グラフ**によって与えられる。抽象グラフは抽象ノードと**抽象リンク**からなるものである。**抽象リンク**をどのように付加するかについては 3.1 節で述べる。

抽象化の目的は与えられた具体システムに対し、保守的な模倣となる抽象システムを求めることである。すなわち、具体システムの初期状態から到達可能な具体グラフ $\mathcal{M} = (M, \{R_a\}, L)$ に対し、抽象システムの初期状態から到達可能な抽象グラフ $\mathcal{G}$ が存在して、任意の $s \in M$ に対して $\alpha_{\mathcal{M}}(s)$ が $\mathcal{G}$ のノードであり、sR_at となる任意の $s, t \in M$ に対して $\mathcal{G}$ において $\alpha_{\mathcal{M}}(s)$ から $\alpha_{\mathcal{M}}(t)$ へ a でラベル付けられた抽象リンクが存在するようなものを求めることである。このような抽象システムは、具体システムが絶対にデッドロック状態に陥らないなどの安全性検査に用いることができる。

保守的な模倣では、具体システムにおいて存在するものは抽象システムでも存在するということは言えるが、一般には逆は成り立たず、具体システムにおいて存在しえないもの抽象システムが含んでしまうことがある。検証を成功させるには、検証したい性質に従って精度を落とさずに模倣することが重要となる。時相論理式の有限集合 F のとり方を変えて様々な抽象化を試すことで、適切な模倣を行う抽象化を得ることを可能にしようというのが我々の狙いである。

具体システムにおける書き換えによる遷移を抽象システム上の遷移として模倣する方法は、問題領域によっていくつかバリエーションが考えられる。文献 [7] のセル・オートマトンの例では、書き換えに必要な情報を抽象グラフが持てるように F を与え、抽象グラフ上で書き換えを行い、さらに時相論理式の充足可能性判定を用いて抽象ノード・リンクの復元、および不要な抽象ノード・リンクの削除を行っている。文献 [11] ではヒープ中のポインタ構造をグラフと、プログラムによるポインタ操作をグラフの書き換えとそれぞれみなしているが、ここでは各ポインタ操作に対する最弱事前条件を求め、時相論理式の充足可能性判定と組み合わせることによって抽象グラフを求めている。いずれにしても時相論理式の充足可能性判定が抽象システム上の遷移の自動計算において重要な役割を果たしている。

3 なぜ時相論理か?

我々が時相論理を抽象化に用いているのは、

1. 時相論理に対するいくつかの拡張がグラフ構造の抽象化に有用であること
2. 決定可能な論理を用いることによって、充足可能性判定を抽象化に利用できること

が主要な理由である。これらについて以下に説明する。

3.1 2 方向の様相

空間的な性質を記述する際、順方向だけでなく逆方向の様相についての記述を要することがある。例えば、1 次元セル・オートマトン [7] においては、各抽象ノード (セル) は抽象システムにおける書き換えのために自身の左隣りと右隣りのセルの色の情報を持っていなくてはならない。このような場合、左方向の様相を右方向の様相の逆様相とするのが自然である。逆様相を持つ時相論理を 2 方向の時相論理と呼ぶ。

様相 a に対し、その逆様相を $\overline{a}$ と書く。さらに $\overline{\overline{a}} = a$ である。この 2 方向の設定のもとで、抽象ノード間の抽象リンクは次のように充足可能性と結びつけられる。

> 抽象ノード C, D と様相 a に対し、$\phi_C \wedge \mathsf{EX}_a\phi_D$ と $\phi_D \wedge \mathsf{EX}_{\overline{a}}\phi_C$ の両方が充足可能であるときに限り、C から D に a でラベル付けされた抽象リンクが存在する。

一般には抽象化の精度を上げるため、抽象リンクは初期状態における抽象リンクから遷移とともに継承していき、上記の条件を満たさないものを削っていくというようなことを行うのだが、ここでは簡単のため、抽象グラフにおける抽象リンクは抽象ノードのみから定まるものとする。すなわち、抽象ノードに対して上記の条件を満たす抽象リンクを全て付け加えたものを抽象グラフとするのである。すると、抽象グラフは抽象ノードの集合 $\mathcal{G} \subseteq 2^F$ で表現することができる。

抽象グラフ $\mathcal{G}$ と具体グラフ $\mathcal{M} = (M, \{R_a\}, L)$ に対し、$\forall s \in M.\ \exists C \in \mathcal{G}.\ \mathcal{M}, s \models \phi_C$ が成り立つとき、$\mathcal{G}$ は $\mathcal{M}$ に対して**健全**であるという。このとき、具体グラフにおいて sR_at が成り立つならば、抽象グラフにおいて $\alpha_{\mathcal{M}}(s)$ から $\alpha_{\mathcal{M}}(t)$ に a でラベル付けされた抽象リンクが存在する。よって、このようにして生成された抽象グラフは具体グラフの保守的な模倣となっている。

既に述べた通り、具体ノードの性質を特徴づけるためにユーザが選んだ 2 方向時相論理式の集合の真偽値によって抽象ノードが定まる。この手法は 2 方向時相論理式による述語抽象化とみなすことができる。Sagiv ら [9] はユーザが定義した 3 値論理における述語の集合による述語抽象化を用いて、抽象解釈によって shape analysis を行う枠組を提唱した。これらの述語は「変数 x から指されている」などの検証対象のプログラムから直接導かれるものの他に、「変数 x から到達可能である」といったものも含まれる。後者のような述語は特に「instrumentation predicate」と呼ばれ、プログラムには直接現れないが、静的解析には重要な役割を果たす。これらの述語がプログラムの実行によってどのように変化するかは「述語更新式 (predicate update formula)」という論理式で表現されるが、instrumentation predicate に対して、このような論理式を精度を落とさないように自動的に生成する問題は、特に述語が推移閉包を含むときに難しい問題となる [8]。我々の手法では instrumentation predicate を時相論理式で表現し、抽象システムにおける遷移を求める際には、既に述べたように最弱事前条件の計算と充足可能性判定を組み合わせることにより、この問題に統一的な解法を与えることを目指している。

3.2 Nominal

通常の命題定数の他に、nominal と呼ばれる特別な原子論理式を使用することもある。nominal を含む時相論理は hybrid 時相論理と呼ばれる。nominal は構文的には命題定数と同じであるが、意味的には異なっており、Kripke 構造 $\mathcal{M} = (M, R, L)$ において、各 nominal x について $x \in L(s)$ となる $s \in M$ は唯一でなくてはならない。すなわち、x は Kripke 構造のちょうど 1 点で成立するということであり、x がその 1 点を指定しているとみなすことができる。

nominal によりグラフ上の空間的性質を記述する能力が強化される。例えば、「ノード s がループ上にある」というという性質は「s は s からリンクを辿ることによって到達可能である」と言い換えることができ、これは s を指定する nominal x を用いて「$x \supset \mathsf{EXEF}x$」と記述される。x が通常の命題定数であればこのような性質は記述できない。

文献 [9] における shape analysis では、抽象的なヒープ構造は summary ノードと、non-summary ノードという、2 種類のノードを持っている。summary ノードは「2 つ以上の具体ノード」を表現し、non-summary ノードは「ちょうど 1 つの具体ノード」を表現するものである。このようなノードの種類の区別が、表現力および抽象化の正確さを増すうえで重要な役割を果たしている。我々の枠組では前者が nominal を用いない通常の論理式に、後者が nominal を用いた論理式にそれぞれ対応する。nominal なしでは「ちょうど 1 つの具体ノード」は表現できないため、nominal の導入は [9] と同様の shape analysis を行ううえで欠かせないものである。

3.3 充足可能性

抽象システムにおける抽象リンクや遷移の計算に充足可能性を用いることは既に述べた通りである。しかしいくら使用している論理が決定可能であったとしても、充足可能性判定手続きがあまりに複雑になってしまうと、計算量が上がるだけでなく、実装が困難になって実際的でないという問題も発生する。例えば、2 方向様相 μ 計算については決定手続きが既に知られている [13] が、実際の実装は我々の知る限り存在しない。

そこで、我々は 2 方向様相 μ 計算よりは弱いが 2 方向性を有する時相論理の例である 2 方向 CTL[10] およびそれを含む無交代 2 方向様相 μ 計算 [12] に対

して、BDD[3] を用いた充足可能性判定手続きとその実装を与えた。この手続きはタブロー法を利用した反復計算によるもので、各ステップは BDD で簡潔に表現できる集合演算の組合せで表現されているため、実装はそれほど困難ではない。

1 方向の CTL については充足可能性に関するシンプルな手続きが知られている [6]。時相論理式の展開 (例えば $\mathsf{EF}\varphi$ を展開すると $\varphi \vee \mathsf{EXEF}\varphi$ となる) を利用すると、すべての論理式は命題定数と $\mathsf{EX}\varphi$ か $\mathsf{AX}\varphi$ の形の論理式のブール結合で表現される。この形の論理式の集合をタブローのひとつのノードとし、このようなノードを全て集めてできるタブローを考える。タブローのノードに含まれる論理式はそのノードで真となるものである。このように考えうる全てのノードを含むタブローから始めて、「矛盾する」ノードを繰り返し除去していく。ここでいう「矛盾」にはいくつかの種類があるが、最も重要なのは eventuality に関するものである。例えば $\mathsf{EF}\varphi$ が成り立つノードがあるのに、そのノードから φ が真となるノードに全く到達できない場合である。1 方向 CTL の場合はこの種の矛盾は比較的簡単に判定可能である。しかし、2 方向の場合には逆方向の様相の影響による、1 方向には存在しない種類の矛盾も考慮しなければならない。2 方向 CTL の場合はタブローの隣り合うノードの間で付加的なチェックを行えば十分である。しかし、無交代 2 方向様相 μ 計算の場合にはタブローのノードに含まれる論理式の間に付加的な関係を導入する必要がある。詳細は [12] を参照されたい。

hybrid 版 (nominal を含むもの) についても同様の方法で充足可能性検査器を開発中である。

3.4 “Shape Analysis is Tableau”

上に述べたように、我々は 2 方向時相論理式の充足可能性検査にタブロー法を用いている。この方法では考えうる全てのタブローのノードの集合から始めて、矛盾するノードを繰り返し除去していくことによってタブローを構成する。このようにして作られたタブローは実質的に与えられた論理式を充足させる全てのモデルを符号化している。

タブローの構成と shape analysis は以下のように密接に関連している。抽象グラフ $\mathcal{G}$ に対し、論理式 $\psi_{\mathcal{G}} = \bigwedge\{\mathsf{A}\neg\phi_C \mid C \notin \mathcal{G}\}$ を考える。ここで、A は “global modality”[2] の一種であり、$\mathsf{A}\phi$ はモデルのあらゆる点において ϕ が真となることを示している。このとき Kripke 構造 $\mathcal{M} = (M, \{R_a\}, L)$ は $\forall s \in M.\ \mathcal{M}, s \models \bigwedge\{\neg\phi_C \mid C \notin \mathcal{G}\}$ が成り立つとき、かつそのときに限り $\psi_{\mathcal{G}}$ のモデルとなる。

いま $\mathcal{M}$ が $\psi_{\mathcal{G}}$ のモデルであるとする。$\forall s \in M.\ \exists C \subseteq F.\ \mathcal{M}, s \models \phi_C$ はいつでも成り立つから、$\forall s \in M.\ \exists C \in \mathcal{G}.\ \mathcal{M}, s \models \phi_C$ が言える。これは $\mathcal{G}$ が $\mathcal{M}$ に関して健全であるということに他ならない。

よって、shape analysis に用いられる抽象グラフが表す全ての具体グラフは、抽象グラフから作られるある論理式に対応するタブローによってカバーされる。この意味で、決定可能な時相論理の使用とその充足可能性検査が、タブロー法と shape analysis を強く結びつけている。

3.5 時空相論理

我々は時相論理における様相を空間的性質の記述に用いてきたが、これは元来はもちろん時間的性質を記述するために導入されたものである。セル・オートマトンの例ではセルは生成も破壊もされないため、セルの連結の様子は時間発展に関して変化しない。このように時間をまたがるノード (セル) の同一性が考慮できる場合には、空間的性質の時間的変化を考えることができる。例えば、S, T, U を空間的性質として、「S を満たすノードについては、時間発展において U が成り立つまではずっと T が成り立っている」などの性質である。

一般にはグラフに対して確めたい性質は空間的関係だけではなく、それに時間的関係が加わったものとなる。そこで我々は両方の種類の様相の相互作用を記述する「時空相」論理を確立したいと考えている。なお、3.2 節で述べた nominal を用いると、「ある nominal が各時刻で指定しているノード」を考えることにより、時間をまたがるノードの同一性を考えることができる。文献 [11] に基いて構築されようとしているソフトウェアモデル検査器では、このように nominal を通した形である種の時空相的性質を検証する試みがなされている。

4 応用例

ここでは我々が時相論理による抽象化を用いた検証の対象として考えているものについて述べる。特殊化した枠組の中で既に検証したものも含んでいる。

並行ごみ集めに見られるようなリンク構造における非同期的書き換えは、グラフ書き換えの典型例である。並行ごみ集めにおいては、mutator と collector と

いう 2 種類のプロセスが並行動作している。mutator は与えられたプログラムに従ってヒープ中のリンク構造を変化させる。collector は mutator プロセスを止めることなく、ヒープ中の使用していないセルを回収する。

我々の枠組の特徴はヒープに関する性質を時相論理式を用いて表現し、さらに自動的に抽象化を行うために充足可能性検査を行うことである。我々はこれを文献 [5] にあるような述語抽象化とモデル検査による shape analysis の枠組に適用することを考えている。文献 [5] ではヒープにおける空間的性質の記述に用いることのできる述語が固定されており、専用の抽象解釈アルゴリズムを使用していたが、これを時相論理式で表現するようにし、一般的な抽象化アルゴリズムを構築しようとしている。ここで充足可能性検査とともに重要となるのがポインタ操作に関する最弱事前条件の計算で、これは田辺ら [11] の研究で行われている。

いくつかのネットワーク上の分散アルゴリズムのグラフ書き換え系とみなすことができる。例えば、相互排他、経路制御プロトコル、DNS 情報の伝播、モバイル・アドホック・ネットワークにおけるリーダ選挙アルゴリズムなどである。本稿では触れられなかったが、各ノードが時間オートマトン [1] のようにクロック値を持つような拡張も研究中であり、タイムアウトを含むようなネットワークプロトコルではこの拡張は重要である。

セル・オートマトンもグラフ書き換えの一種とみなすことがきるが、グラフの形状は変化せずに、グラフのノード (セル) に結び付けられたラベルが遷移していくのが特徴である。セル・オートマトンの遷移は同期的・非同期的なものの両方を考えることができる。同期的遷移は全てのノードのラベルが一斉に変化するもので、非同期的遷移は一度にに一つのノードのラベルだけが変化するものである。文献 [7] に挙げられている食事をする哲学者の例題は非同期的セル・オートマトンの例で、そこではデッドロックの解析に抽象化を用いている。

5 おわりに

時相論理による抽象化について、枠組の説明と、時相論理を用いる理由、さらにその応用例について、現在進行中の研究も含めて述べた。

謝辞 本研究に関して議論いただいた産業技術総合研究所の田辺良則氏、高橋孝一氏に感謝する。

参考文献

[1] Alur, R., and Dill, D.L.: A Theory of Timed Automata, *Theoretical Computer Science*, Vol. 126, 1994, pp. 183–236.

[2] Blackburn, P., de Rijke, M., and Venema, Y.: *Modal Logic*, Cambridge University Press, 2001

[3] Bryant, R.E: Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams, *ACM Computing Surveys*, Vol.24, No.3(1992), pp. 293–318.

[4] Clarke, Jr., E. M., Grumberg O., and Peled, D. A.: *Model Checking* The MIT Press, 1999.

[5] Dams, D., and Namjoshi, K. S.: Shape analysis through predicate abstraction and model checking, *Fourth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI03)*, 2003, pp. 310–324.

[6] Emerson, E. A.: Temporal and Modal Logic, *Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics*, Elsevier Science Publishers B.V., 1990, pp.995–1072.

[7] Hagiya, M., Takahashi, K., Yamamoto, M., and Sato, T.: Analysis of Synchronous and Asynchronous Cellular Automata using Abstraction by Temporal Logic, *FLOPS2004: The Seventh Functional and Logic Programming Symposium*, LNCS, Vol.2998, 2004, pp. 7–21.

[8] Reps, T., Sagiv, M., Loginov, A.: Finite Differencing of Logical Formulas for Static Analysis, *European Symposium on Programming*, 2003, pp. 380–398.

[9] Sagiv, M., Reps, T., and Wilhelm, R.: Parametric shape analysis via 3-valued logic, *ACM Transactions on Programming Languages and Systems*, Vol. 24, No. 3, May 2002, pp. 217–298.

[10] 田辺良則, 高橋孝一, 山本光晴, 佐藤貴洋, 萩谷昌己: BDD を用いた 2 方向 CTL 論理式充足可能性決定手続きの実装, **コンピュータソフトウェア**, Vol. 22, No.3, 2005, pp. 154–166.

[11] Tanabe, Y., Takai, T., Sekizawa, T., Takahashi, K.: Preconditions of Properties Described in CTL for Statements Manipulating Pointers. *Supplemental Volume of the 2005 International Conference on Dependable Systems and Networks*, June 28–July 1, 2005, pp.228–234

[12] Tanabe, Y., Takahashi, K., Yamamoto, M., Tozawa, A., and Hagiya, M.: A Decision Procedure for the Alternation-Free Two-Way Modal μ-Calculus, *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, 2005, to appear.

[13] Vardi, M.Y.: Reasoning about The Past with Two-Way Automata: *Automata, Languages and Programming ICALP 98*, LNCS, Vol. 1443, 1998, pp. 628–641.