Vol.2017-MPS-115 No.15
2017/9/26

階層モデル検査: 論理, 翻訳および具体例

矢野 龍[1,a)]　上出 哲広[2,b)]

概要：階層モデル検査は, 階層構造を持つモデルを適切かつ厳密に検証可能な, モデル検査技術の拡張である. 本研究では, 階層モデル検査のための論理および翻訳を提案し, それらを用いた階層モデル検査の具体例としてのトイ・イグザンプルをいくつか提示する. まず, 階層モデル検査のための論理として, 従来の標準的な線形時間論理（LTL: Linear-time Temporal Logic）と計算木論理（CTL: Computation Tree Logic）を拡張した新たな論理をいくつか導入する. 次に, それら拡張論理の論理式を LTL または CTL の論理式に変換するための翻訳関数をそれぞれの論理に対して与える. そして, それら翻訳関数を用いて埋め込み定理を証明する. 最後に, それら新たな論理および翻訳関数を用いた, 階層モデル検査のトイ・イグザンプルをいくつか提示する. 本稿の内容は, 国際会議論文 [9] の内容を基にしてまとめたものである.

Hierarchical Model Checking: Logics, Translations and Examples

Ryu Yano[1,a)]　Norihiro Kamide[2,b)]

1. 階層モデル検査

モデル検査（model checking）は, システムの数理的なモデルが期待する性質を持っているかどうかを判定する技術であり, 主に並行システムを検証するために広く使用されている [1], [2], [3], [6]. モデル検査で使用されている標準的な時間論理には, LTL（Linear-Time Temporal Logic）[12] と CTL（Computation Tree Logic）[2] がある. 階層モデル検査（hierarchical model checking）は, 階層構造を適切かつ厳密に扱うためのモデル検査パラダイムであり, 従来の標準的なモデル検査で使用されている LTL や CTL の拡張論理体系を使用する [7], [8], [11].

LTL と CTL は, モデル検査のための最も重要な時間論理である. LTL は, 時間の流れを直線で表現する線形時間パラダイムに基づいており, 標準的なモデル検査器である SPIN [6] と NuSMV [1] の基盤となる論理である [12]. CTL は, 時間の流れを木構造で表現する分岐時間パラダイムに基づいており, NuSMV の基盤となる論理である [2]. 一方, モデル検査のための標準的な時間論理であるこれら LTL と CTL は, 階層構造や階層情報を持つシステムを適切に検証するのに適していない. その理由の一つとしては, それらが階層構造・情報を自然に記述するための演算子を持たないことを挙げることができる.

階層構造・情報を適切に記述するための様相演算子の一つとして, 列様相演算子（sequence modal operator）がある. そして, これまで, この列様相演算子を持つ拡張された時間論理がいくつか提案され, 研究されてきた [7], [8], [10], [11]. 例えば, 列様相演算子を持つ時間論理の一種として, CTLS* がある [8], [11]. これは, CTL*（Full Computation Tree Logic）[4], [5] に列様相演算子を加えた拡張である. CTLS* は, 概念階層やオントロジーを記述・検証するために使用された. また, LTL に列様相演算子を加えた拡張である SLTL（Sequence-Indexed Linear-Time Temporal Logic）が, [10] で提案された. SLTL は, セキュアな認証システムを記述・検証するために導入された. CTL に列様相演算子を加えた拡張として, SPCTL（Sequence-Indexed Paraconsistent Computation-Tree Logic）が [7] で導入された. SPCTL は CTL に列様相演算子と矛盾許容否定結合子を加えた拡張であり, 階層情報を考慮した臨床推論を扱うために使用された.

1 帝京大学理工学部ヒューマン情報システム学科
2 帝京大学理工学部情報電子工学科
a) 146123fx@stu.teikyo-u.ac.jp
b) drnkamide08@kpd.biglobe.ne.jp

2. 提案した論理と翻訳

本研究では, 階層モデル検査のための時間論理として sLTL（Sequential Linear-Time Temporal Logic）および sCTL（Sequential Computation Tree Logic）を新たに導入した. これら論理はそれぞれ LTL と CTL に列様相演算子を加えた拡張であり, 階層構造や階層情報を適切に記述可能な論理である. 本研究では, さらに sLTL と sCTL の論理式をそれぞれ LTL と CTL の論理式に翻訳する関数を導入した. これら翻訳関数を用いることにより, sLTL と sCTL の各々に対する埋め込み定理を証明した. これら埋め込み定理の帰結として, sLTL と sCTL のモデル検査問題の決定手続きがそれぞれ LTL と CTL のものと一致するということを導くことができる. これは, 導入した翻訳関数が多項式時間のリダクションになっているからである. これら提案した埋め込み定理は, sLTL と sCTL によって記述された階層構造・情報を持つシステムを, LTL および CTL に対する標準的なモデル検査アルゴリズムを用いて検証できるということを意味している.

これまでの研究で提案されてきた CTLS* [8], [11], SLTL [10] および SPCTL [7] は, 複雑な多重充足関係 $\models^{\hat{d}}$ を使用していた. ここで, $\hat{d}$ は列を表す. 一方, 本研究では, シンプルな単一充足関係 $\models^{\star}$ を提案した. このような単一充足関係は, 標準的な LTL や CTL の単一充足関係との親和性が良く, 埋め込み定理の証明を簡単化できる. そして, このような単一充足関係を用いることにより, 列様相演算子をシンプルに形式化することが可能になり, sLTL と sCTL の両方を同一の枠組みで統一的に扱うことが可能になる. これらメリットに加えて, 本研究で提案した sLTL は, 従来の SLTL より表現力が強いというアドバンテージを持つ. 従来の SLTL では Untile 演算子およびその双対である Release 演算子が存在しなかった. しかし, sLTL ではこれら演算子が, 単一充足関係を使用することにより自然に定義されている.

3. 提案した具体例

本研究では, 上記拡張時間論理 sLTL と sCTL およびそれらの翻訳関数に基づいた, 階層モデル検査の簡単な応用例としてのいくつかのトイ・イグザンプルを提案した. それらトイ・イグザンプルはモデル検査器 SPIN と NuSMV の仕様記述言語を用いて実装し, 検証することができる. 提案したトイ・イグザンプルの一つでは, 臨床推論に現れる状況を記述する際に, alcoholic $\subseteq$ patient $\subseteq$ drinker $\subseteq$ human のような人間の階層を扱った. これらの階層は, sLTL および sCTL における列様相演算子を用いて記述される. 例えば, sLTL を使用することにより, “Is there an alcoholic in the hospital who is not drinking in the second stage of the disease AUD (alcohol use disorder)?” のようなステートメントを以下のように記述し, 検証できる.

$$[human; drinker; patient; alcoholic]\mathrm{F}(inHospital \wedge \neg drinking \wedge 2nd).$$

この記述において, $[human; drinker; patient; alcoholic]$ の部分が, 階層性を表現した列様相演算子である. また, sCTL を使用することにより, “Is there an alcoholic out of the hospital who is drinking in the second stage of the AUD?” のようなステートメントを以下のように記述し, 検証できる.

$$[human; drinker; patient; alcoholic]\mathrm{EF}(\neg inHospital \wedge drinking \wedge 2nd).$$

加えて, sCTL を使用することにより, “Is there no dead patient who will not be alive again?” のようなステートメントも以下のように記述し, 検証できる.

$$[human; drinker; patient]\mathrm{EF}(died \wedge \neg\mathrm{EF}\neg died).$$

謝辞 本研究は JSPS 科研費 JP26330263 の助成を受けた.

参考文献

[1] R. Cavada, A. Cimatti, C.A. Jochim, G. Keighren, E. Olivetti, M. Pistore, M. Roveri and A. Tchaltsev, NuSMV 2.6 user manual, 144 pages, 2015.
[2] E.M. Clarke and E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, LNCS 131, pp. 52-71, 1981.
[3] E.M. Clarke, O. Grumberg, and D.A. Peled, Model checking, The MIT Press, 1999.
[4] E.A. Emerson and P. Sistla, Deciding full branching time logic, Information and Control 61, pp. 175-201, 1984.
[5] E.A. Emerson and J.Y. Halpern, “Sometimes” and “not never” revisited: on branching versus linear time temporal logic, Journal of the ACM 33 (1), pp. 151-178, 1986.
[6] G.J. Holzmann, The SPIN model checker: Primer and reference manual, Addison-Wesley, 2006.
[7] N. Kamide, Inconsistency-tolerant temporal reasoning with hierarchical information, Information Sciences 320, pp. 140-155, 2015.
[8] N. Kamide and K. Kaneiwa, Extended full computation-tree logic with sequence modal operator: Representing hierarchical tree structures, Proceedings of the 22nd Australasian Joint Conference on Artificial Intelligence, LNAI 5866, pp. 485-494, 2009.
[9] N. Kamide and R. Yano, Logics and translations for hierarchical model checking, Proceedings of the 21st International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, To appear.
[10] K. Kaneiwa and N. Kamide, Sequence-indexed linear-time temporal logic: Proof system and application, Applied Artificial Intelligence 24 (10), pp. 896-913, 2010.
[11] K. Kaneiwa and N. Kamide, Conceptual modeling in full computation-tree logic with sequence modal operator, International Journal of Intelligent Systems 26 (7), pp. 636-651, 2011.
[12] A. Pnueli, The temporal logic of programs, Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46-57, 1977.