有限オートマトンとスティッカー系に関するCoqによる形式証明について

溝口, 佳寛
九州大学マス・フォア・インダストリ研究所 : 准教授

田中, 久治
佐賀大学

坂下, 一生
九州大学マス・フォア・インダストリ研究所

井口, 修一
九州大学大学院数理学研究院 : 助教

出版情報：日本数学会2014年度年会応用数学講演アブストラクト, pp.59-62, 2014-03. 日本数学会
バージョン：

有限オートマトンとスティッカー系に関する Coqによる形式証明について

溝口 佳寛　(九大 IMI)[*1]
田中 久治　(佐賀大工)
坂下 一生　(九大 IMI)
井口 修一　(九大数理)

概　　要

有限オートマトンを構成する演算子や文字列の (無限) 集合を Coq 証明支援系を用いて形式的に定義しました. ここでは, Ssreflect (Small Scale Reflection Extension for the Coq system) も利用します. スティッカー系は Păun と Rozenberg により 1998 年に紹介された DNA 計算のための形式モデルです. 彼らは有限オートマトンと同等の能力を持つスティッカー系を構成する具体的な構成方法を示し, スティッカー系の計算能力が有限オートマトンのそれを含むことを示しました. しかし, その構成方法には小さな誤りがありました. それを修正した構成方法を示すことと同時に私たちの構成方法の正当性に対して, コンピュータで検証可能な形式的証明を与えることが本研究の目的です. 現在までに, 諸性質を形式証明するための有限オートマトンとスティッカー系の定義や関数の形式化が出来ました. いくつかの具体例の形式的証明, Păun らの構成に誤りがあることの形式証明を含めて報告します.

1. はじめに

コンピュータを利用した証明で広く知られているものとして平面地図塗り分け問題である「四色問題」があります. 1976 年に Appel と Haken らによってコンピュータの計算を利用して解決されました [2]. しかし, その計算プログラムに誤りがないのかは簡単に人で検証できるものではありませんでした. 2004 年に Gonthier(INRIA, マイクロソフト研究所) は, Coq 証明支援系とその拡張 Ssreflect (Small Scale Reflection Extension[6]) を用いて, プログラムと証明をコンピュータで検証可能な形で与えました [5]. Coq はフランスの INRIA 研究所で開発された証明支援言語で, 型付きラムダ計算言語の一種で, プログラムとその正当性を自動検証可能な形で記述するための言語です [3, 4]. 実社会においては, 例えば周辺機器との通信プロトコルの実装やハードウェア制御等のプログラムを誤りなく実現するために利用されます. プログラムの自動検証支援系では, Coq 以外にも, Isabelle(英国, ケンブリッジ大学) や Agda(スウェーデン,Chalmers 工科大学) などが, また, 数学定理の検証系では, Mizar(ポーランド, Bialystok 大学) などが知られています. 近年では, 数学定理の検証可能証明 (形式化) が注目されており, 2004 年の四色定理の後, 2012 年に奇数位数定理の検証可能証明が Gonthier らにより完成されています [7, 13]. さらに, 3 次元空間の球面充填に関する問題 (ケプラー予想) についても人手では証明の検証が難しく, Hales らによりコンピュータで検証可能な形式証明の構築が進められています [8]. そして, Voevodsky(プリンストン高等研究所) らはホモトピー理論と型理論を融合したホモトピー型理論の Coq を用いた定式化に取り組んでいます [10, 14].

本研究は科研費 (課題番号:25610034) の助成を受けたものである.
2010 Mathematics Subject Classification: 03B35,03D05,68Q45,68T15
[*1] e-mail: ym@imi.kyushu-u.ac.jp

また, Affeldtd と萩原らは情報セキュリティの問題を解決するために, シャノンの定理の形式証明を発表しています [1].

スティッカー系は分子コンピュータの形式モデルで, 1998 年に Păun と Rozenberg によって紹介されました [9, 11]. スティッカー系におけるスティッカー操作は, Watson と Click により提唱された DNA 二重螺旋の分子模型の構造に基づいています. Păun と Rozenberg はこの形式モデルの計算能力について考察しました. 例えば, 与えられた有限オートマトンに対して, それと同等の計算能力を持つスティッカー系を構成し, 有限オートマトンの受理言語とスティッカー系による生成言語の同等性を証明しました.

スティッカー系の構成方法は具体的ですが, その計算能力の同等性の証明の確認 (検証) は容易ではありません. そこで, 私たちは証明支援系を用いてコンピュータで検証可能な証明を構成することを試みました. まず最初に, Coq 証明支援系での有限オートマトンやスティッカー系の定義や基本的な性質の形式証明を構成しました. また, 有限オートマトンから同等なスティッカー系を構成する関数を Coq の関数として実現しました. Coq において取り扱いを容易にするために, ドミノの定義とスティッカー操作関数を改善していますが, 多くの関数の実現は, Perera と溝口による Haskell 言語による実現に準拠しています [12]. 私たちの構成した Coq モジュールは, 文字列, 有限オートマトン, 受理言語を取り扱う **Automaton** モジュールとドミノ, スティッカー系, 生成言語を取り扱う **Sticker** モジュールの 2 つです. これらのモジュールを用いて, 有限オートマトンとスティッカー系に関する様々な性質を形式的に記述し, 形式証明を構成することが出来るようになります. 現在までに, Păun らのスティッカー系の構成方法には小さな誤りがあることの証明. また, その反例を構成することは形式的に行えました.

本発表では, 有限オートマトンとスティッカー系の Coq による形式化の紹介, いくつかの基本性質の形式証明, Păun らの結果への反例の形式証明を紹介します. 今後の課題は, 一般の有限オートマトンに対してのスティッカー系との計算能力の同等性についての形式的な証明を構成することです.

2. 有限オートマトンとスティッカー系

$\mathbf{N}$ を自然数の集合, Σ をアルファベットの有限集合, Σ^* を空文字列 (ε) も含む Σ 上の文字列全体の集合とします. $n \in \mathbf{N}$ に対して, $\Sigma^n = \{w \in \Sigma^* \mid |w| = n\}$, $\Sigma^{\leq n} = \{w \in \Sigma^* \mid 1 \leq |w| \leq n\}$ とします.

定義 2.1 (有限オートマトン) Q を状態の有限集合, Σ をアルファベットの有限集合, $\delta : Q \times \Sigma \to Q$, $q_0 \in Q$, $F \subset Q$ とするとき, 5 つ組 $M = (Q, \Sigma, \delta, q_0, F)$ を有限オートマトンという.

q_0 を初期状態, F を受理集合, δ を状態遷移関数と言います. 状態遷移関数 δ は $q \in Q$, $x \in \Sigma$, $w \in \Sigma^*$ に対して, $\delta^*(q, \varepsilon) = q$, $\delta^*(q, xw) = \delta^*(\delta(q, x), w)$ とすることで, 自然に $\delta^* : Q \times \Sigma^* \to Q$ に拡張することが出来ます.

定義 2.2 有限オートマトン $M = (Q, \Sigma, \delta, q_0, F)$ に対して, $L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$ とし, $L(M)$ を M の受理言語集合という.

定義 2.3 (ドミノ) Σ をアルファベット集合, $\rho \subset \Sigma \times \Sigma$ とする. $l, r \in \Sigma^*$, $x, y \in \mathbf{N}$ に対して, (l, r, x, y) は, 以下の条件 (i), (ii), (iii) を満たすときに, (Σ, ρ) 上のドミノという.

(i) If $x \geq 0$ then $(l[i + x], r[i]) \in \rho$, $1 \leq i \leq min(|r| - x, |l|)$,

(ii) If $y \geq 0$ then $(l[i], r[i+y]) \in \rho, 1 \leq i \leq min(|r| - y, |l|)$,

(iii) $x \times y = 0$.

(Σ, ρ) 上のドミノ全体の集合を D と書くことにします.

定義 2.4 関数 $\mu : D \times D \to D \cup \{\perp\}$ を次のように定義する.

$$\mu((l_1, r_1, x_1, y_1), (l_2, r_2, x_2, y_2)) = \begin{cases} (l_1 l_2, r_1 r_2, x_1, y_1) & \text{(条件 * を満たすとき)} \\ \perp & \text{(それ以外)} \end{cases}$$

$$\text{(条件 *)} \quad (l_1 l_2, , r_1 r_2, x_1, y_1) \in D \text{ and } x_1 + y_2 + |r_1| = x_2 + y_1 + |l_1|$$

定義 2.5 (スティッカー系) Σ をアルファベットの有限集合, $\rho \subset \Sigma \times \Sigma$, D を (Σ, ρ) 上のドミノ全体の集合とする. 有限集合 $A \subset D, R \subset D \times D$ に対して, 4 つ組 $\gamma = (\Sigma, \rho, A, R)$ をスティッカー系と言う.

スティッカー系 $\gamma = (\Sigma, \rho, A, R)$ に対して, 関係 $\Rightarrow$ を次のように定義します.

$$x \Rightarrow y \overset{def}{\Longleftrightarrow} \exists (u, v) \in R, y = \mu(u, \mu(x, u))$$

$\Rightarrow^*$ を関係 $\Rightarrow$ の反射推移閉包とします.

定義 2.6 スティッカー系 $\gamma = (\Sigma, \rho, A, R)$ に対して, $LM(\gamma) = \{(l, r, 0, 0) \mid a \Rightarrow^* (l, r, 0, 0), a \in A, |l| = |r|\}$, $L(\gamma) = \{l \in \Sigma^* \mid (l, r, 0, 0) \in LM(\gamma)\}$ とし, $L(\gamma)$ を γ の生成言語と言う.

定義 2.7 有限オートマトン $M = (Q, \Sigma, \delta, q_0, F_M)$ に対して, スティッカー系 $\gamma_M = (\Sigma, \rho, A, R)$ を次のように定義する.

$$\begin{aligned}
\rho &= \{(a, a) \mid a \in \Sigma\}, \\
A &= A_1 \cup A_2, \\
A_1 &= \{(x, x, 0, 0) \mid x \in L(M), |x| \leq k + 2\}, \\
A_2 &= \bigcup_{i=1}^{k+1} \{(xu, x, 0, 0) \mid x \in \Sigma^{(k+2-i)}, u \in \Sigma^i, \delta^*(0, xu) = i - 1\}, \\
R &= D \cup F, \\
D &= \bigcup_{i=1}^{k+1} \bigcup_{j=1}^{k+1} \{((\varepsilon, \varepsilon, 0, 0), (xu, vx, 0, |v|)) \mid x \in \Sigma^{(k+2-i)}, u \in \Sigma^i, v \in \Sigma^j, \delta^*(j - 1, xu) = i - 1\}, \\
F &= \bigcup_{i=1}^{k+2} \bigcup_{j=1}^{k+1} \{((\varepsilon, \varepsilon, 0, 0), (x, vx, 0, |v|)) \mid v \in \Sigma^j, x \in \Sigma^i, \delta(j - 1, x) \in F_M\}, \\
k &= |Q| - 1.
\end{aligned}$$

定理 2.1

$$L(\gamma_M) = L(M).$$

上記の定理 2.1 は Păun と Rozenberg らにより最初に提示されましたが, 証明には小さな誤りがありました. 彼らの有限オートマトンからスティッカー系への構成方法に誤りがありました. 具体的には F の構成のための添字の範囲がずれていました. 私たちは本形式化により, そのことを検証出来ました.

有限オートマトン $M_1 = (\{0,1\},\{a,b\},\delta_1,0,\{1\})$ $(\delta_1(0,a) = \delta_1(1,b) = 0,\ \delta_1(0,b) = \delta_1(1,a) = 1)$ に対して, 彼らの構成では $L(\gamma_{M_1}) \neq L(M_1)$ となります. 我々の定義では, $(abb, ab, 0, 0) \in A$, $((\varepsilon,\varepsilon,0,0),(aba,baba,0,1)) \in F$, および, $\mu((abb,ab,0,0),(aba,baba,0,1)) = (abbaba, abbaba, 0, 0)$ より, $abbaba \in L(\gamma_{M_1})$ を得ます. 一方, [11] の F の定義では $((\varepsilon,\varepsilon,0,0),(aba,baba,0,1)) \notin F$ であるために, $abbaba \in L(M_1)$ であるにも関わらず, $abbaba \notin L(\gamma_{M_1})$ となります.

参考文献

[1] R. Affeldt and M. Hagiwara, Formalization of Shannon's Therems in SSReflect-Coq, Proc. 3rd Conference on Interactive Theorem Proving, LNCS 7406, 233–249, 2012.

[2] K. Appel and W. Haken, Every map is four colourable. Bulletin of the American Mathematical Society, 82, 711–712, 1976.

[3] Y. Bertot, Interactive Theorem Proving and Program Development, Springer, 2004.

[4] A. Chipala, Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant, The MIT Press, 2013, http://adam.chlipala.net/cpdt/.

[5] G. Gonthier, Formal Proof—The Four-Color Theorem. Notices of the American Mathematical Society, 55(11), 1382–1393, 2008.

[6] G. Gonthier, A. Mahboubi, and E. Tassi, A Small Scale Reflection Extension for the Coq system, Inria Research Report No.6455, 2009.

[7] G. Gonthier, et al., A Machine-Checked Proof of the Odd Order Theorem, Proc. 4th Conference on Interactive Theorem Proving, LNCS 7998, 163–179, 2013.

[8] T. C. Hales, A proof of the Kepler conjecture, Annals of Mathematics, 162(3), 1065–1185, 2005.

[9] L. Kari, G. Păun, G. Rozenberg, A. Salomaa and S. Yu, DNA computing, sticker systems, and universality, Acta Informatica, 35, 401–420, 1998.

[10] Á. Pelayo and M.A. Warren, Homotopy Type Theory and Voevodsky's Univalent Foundations, arXive:1210.5658, 2012.

[11] G. Păun and G. Rozenberg, Sticker systems, Theoretical Computer Science, 204, 183–203, 1998.

[12] K. Perera and Y. Mizoguchi, Implementation of Haskell Modules for Automata and Sticker Systems, Journal of Math-for-industory, 1, 51–56, 2009.

[13] The formalization of the Odd Order theorem has been completed September 20th 2012, http://www.msr-inria.fr/news/the-formalization-of-the-odd-order-theorem-has-been-completed-the-20-septembre-2012/.

[14] The Univelent Foundations Program, Homotopy Type Theory: Univalent Foundations of Mathematics, http://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

追記

本原稿は日本数学会 2014 年度年会応用数学講演アブストラクト (2014 年 3 月) (p.59-62) へ掲載された講演アブストラクトの誤植を訂正し, いくつかの参考文献を追加したものです.

(2014 年 3 月 6 日)