等式アーカイブにおける安全な検査

Safely Checking in Equation Archive

玉野 浩嗣 † 西崎 真也 †
Hiroshi TAMANO Shin-ya NISHIZAKI
† 東京工業大学 大学院情報理工学研究科
Dept. of Computer Science, Tokyo Instituteof Technology
tamano@lambda.cs.titech.ac.jp

等式系には語問題 (word problem) が決定可能なものとそうでないものが存在する。前者は等式系と等価な完備な項書換え系があり、この項書き換え系は定理証明系の式の簡略化に大きな効果を発揮する。そのため語問題が決定可能な等式系と対応する完備な項書き換え系のアーカイブをサーバに構築することを目標として、本研究では信頼できないクライアントに対しても安全にこれらを検査する方法を提唱する。

1 はじめに

いくつかの背景から紹介することとする。

- 項書換え系 (Term Rewriting System:TRS)

  λ 項に対する β 簡約や、コンビネータ論理における簡約を一般化して、より一般的な書き換え規則によって計算を記述する理論が項書換え系である。主に等式推論に用いられ、定理証明系などのシステムにおいて式の簡約で重要な役割を果たしている。項書き換え系で重要な性質は停止性と合流性である。前者は書き換え規則の適用により無限の簡約列が存在しないことを保証し、後者は規則の適用順によらず規則適用結果が合流することを保証する。この停止性、合流性の成り立つ項書き換え系のことを完備であるという。完備な項書き換え系では語問題 (word problem) が決定可能であるという重要な性質を満たすことが知られている。

- 定理証明系 (theorem prover)

  定理証明系とは高階論理や型理論などの形式的論理の下で記述された証明をチェックするソフトウェアである。代表的な定理証明系としては HOL[1],Isabelle[2],Coq[3],Mizar[4] などがあげられる。純粋な数学の形式化からハードウェアの検証まで様々な用途に使われている。定理証明系にとって等式変形による式の簡略化は証明の重要な要素の一つである。

- PDIP

  PDIP[5] とは対話的証明器付き証明文書「Proof Document with Interactive Prover」の略で、定理証明スクリプトと HTML 形式の証明文書が一体化した新しい証明文書のアーキテクチャである。証明を読む者にとっては動かしながら証明を読んで行くことができ、また証明を書く者にとっては証明スクリプトが証明を記述する支援を行ってくれる。証明のスタイルは従来の CUI に基づくものと異なり、Web ベースの GUI を利用しマウスによりクリックしながら証明を進めて行くスタイルをとる。2005 年 9 月現在、等式論理と一階述語論理を対象とした PDIP が実装されている。

- 研究の目的：安全な知識と安全な検証

  近年、インターネットの普及によりネットワークを利用した知識の収集が盛んに行われている。例えば、ウォード・カニンガム (Ward Cunningham) により発明された WikiWiki(ウィキウィキ)[6] はコンピュータ上で共同作業により文章の編集を行うツール (および概念) であるが、ネットワークを介して知識を収集するのに便利なツールである。この Wiki を使った有名な例の一つにウィキペディア (Wikipedia[7]) がある。これは様々な人が自由に知識を投稿することにより百科事典を形成していて、利用者は様々な知識をここから得ることができる。

  このような知識を集約することにおいて重要となるのが知識の信頼性である。悪意のある者による間違った知識の投稿や、悪意がなくても人

間のミスによる間違った知識の投稿ができてしまうと利用者が安心してそのシステムを利用することができなくなる。そこで本研究では知識が正しいかどうかを安全に検査する仕組みを考える。

知識が正しいかどうかは形式的に証明されるべきである。したがって知識の対象としては一階述語論理、等式系、型理論などがあげられる。

一階述語論理については、Mizar プロジェクトで代表されるように、すでにたくさんの定理が証明され知識ライブラリができあがっている。等式系については、特に語問題が決定可能な等式系は、対応する完備な項書き換え系が定理証明系の式の簡約において大きな効果を発揮するため重要な知識と考えられる。また後に等式論理版 PDIP との連携も考えているため、今回は等式系とそれに対応する完備な項書き換え系を知識とする。

一般に悪意のある者によりサーバに多大な負荷をかけるような投稿もありうる。そのような投稿を排除するために、これらを安全な計算量で検査する仕組みが不可欠となる。

2 検査方法

2.1 構成

まず構成として、クライアント・サーバモデルを考える。素朴な構成として、クライアントは等式系をサーバへ投稿し、サーバは自動的にこれを検査し受理する構成が考えられる。この構成でサーバが行うことは、クライアントから投稿された等式系に対して完備化手続き行うことである。完備化手続きにより等式系と等価である完備な項書き換え系が生成され、まさにその項書換え系が等式系における語問題が決定可能である証拠となる。しかしここで、完備化手続きがセミアルゴリズムであるということが問題となる。サーバ・クライアントモデルおいて、サーバとクライアントの立場の違いから、クライアントでセミアルゴリズムを動かすことはできてもサーバ側でそれらを動かすことはできない。したがって、より安全な構成にするためにはクライアントは等式系とそれと等価である完備な項書換え系を語問題が決定可能である証拠として投稿する必要がある。クライアントから等式系と、それと等価である項書き換え系が投稿された場合、サーバ側では以下の三つの証明を行う。

- 項書き換え系の停止性の自動証明
- 項書き換え系の合流性の自動証明
- 等式系と項書き換え系の等価性の自動証明

この三つの証明ができれば、その等式系において語問題は決定可能であると言うことができる。 ここで停止性の検査には辞書式経路順序 (lexicographic path order) を使う。また二つ目の合流性については Newman の補題 [8] により停止性が成り立っているものについては局所合流性の検査で十分であることが知られているため局所合流性を示す。停止性の検査に辞書式経路順序を使う理由としては与えられた項書き換え系に対して辞書式経路順序がその停止性を示せるかどうかが決定可能であり、また証明可能な項書き換え系の簡約列の長さに制約が無いことなどがあげられる。

しかし、この構成では次のことが問題となる。

- 与えられた項書き換え系の停止性判定は一般に決定不能である。辞書式経路順序を用いたものに限っても停止性判定は NP 完全問題である [9]。
- 等式系と項書き換え系の等価性の自動証明は一般には決定不能である。

そこで、サーバ側が安全な計算コストで等式アーカイブを検査するには以下のことがクライアント側で必要である。

1. 等式系と等価である完備な項書き換え系を作る。
2. その項書換え系の停止性と局所合流性そして等式系との等価性について前もって証明を行う。
3. 証明の手順をヒントとしてサーバ側に等式系と一緒に送る

以上により、サーバ側では安全な計算コストでヒントを使いその証明を再構築することができる。安全な構成を図 1 に表す。

2.2 節以降では停止性、局所合流性、等式系と項書き換え系の等価性について詳しく見て行く。

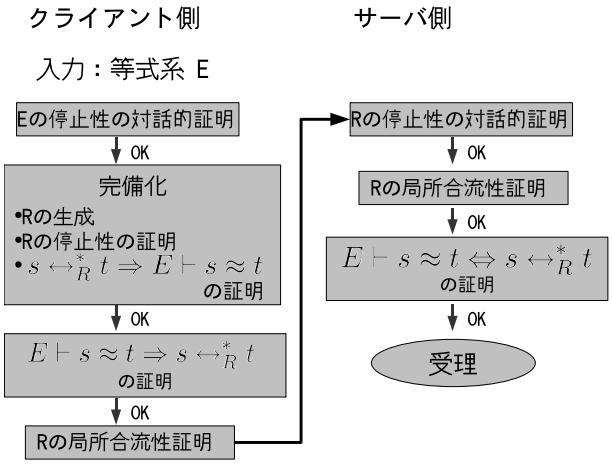

図 1: 安全な構成

2.2 停止性

2.1 節でも述べたように辞書式経路順序を使った停止性の自動証明は NP 完全問題である [9]。したがって、クライアント側ではユーザが関数記号の全順序を与えることにより対話的に停止性を証明するとする。辞書式経路順序において、関数記号が与えられれば $s >_{lpo} t$ であるかを確かめる計算時間は $O(|s| \times |t|)$ となる。したがって、クライアントは関数記号の全順序をサーバに証明のヒントとして送ることにより、サーバは一つの等式につき $O(|s| \times |t|)$ の時間で停止性を証明できることになる。この方法は計算量的に安全といえる。

しかし、この方法ではクライアントとサーバで $s >_{lpo} t$ であるかの計算にオーバーヘッドが多い。よりサーバ側での計算量を削減する方法としてクライアントから項書き換え系の辞書式経路順序による停止性の証明の手順をヒントとしてサーバへ与えることを考える。辞書式経路順序による停止性の証明に用いる推論規則を図 2 に示す。ただし Γ を関数記号の順序対の集合とする。また、証明の手順としてクライアントは (推論規則番号、追加情報) のペアの木をサーバへ送る。ここで追加情報とは LPO_a, LPO_b の場合は p,LPO_c の場合は無し、LPO_d の場合は i である。例えばアッカーマン関数の停止性の証明のヒントは例 2.1 のようになる。

サーバはクライアントから送られてくる証明の手順をヒントとして、停止性の証明を再構築することができる。これにより $O(|s| \times |t|)$ であった計算量は $O(|s| + |t|)$ へと削減される。

例 2.1 (アッカーマン関数)

$$R_{ack} = \begin{cases} a(0,y) & \to & s(y) \\ a(s(x),0) & \to & a(x,s(0)) \\ a(s(x),s(y)) & \to & a(x,a(s(x),y)) \end{cases}$$

$\Gamma = \{a > s\}$ とすると、アッカーマン関数の停止性は以下のように証明され、それに対応するヒントは次のようになる。

$$\dfrac{\dfrac{a(0,y)|_2 = y}{\Gamma \vdash a(0,y) \triangleright y}\ (a) \quad a > s \in \Gamma\ (c)}{\Gamma \vdash a(0,y) \triangleright s(y)}$$

$$\dfrac{\dfrac{s(x)|_1 = x}{\Gamma \vdash s(x) \triangleright x}\ (a) \quad \dfrac{\dfrac{a(s(x),0)|_2 = 0}{\Gamma \vdash a(s(x),0) \triangleright 0}\ (a)}{\Gamma \vdash a(s(x),0) \triangleright s(0)}\ (c)}{\Gamma \vdash a(s(x),0) \triangleright a(x,s(0))}\ (d)$$

$$\dfrac{s(x) = s(s) \quad \dfrac{\dfrac{a(s(x),s(y))|_{21} = y}{\Gamma \vdash a(s(x),s(y)) \triangleright y}\ (a) \quad \dfrac{s(y)|_1 = y}{\Gamma \vdash s(y) \triangleright y}\ (a)}{\backslash\clubsuit/ \quad \Gamma \vdash a(s(x),s(y)) \triangleright a(s(x),y)}\ (d)}{\Gamma \vdash a(s(x),s(y)) \triangleright a(x,a(s(x),y))}\ (d)$$

♣ の証明木

$$\dfrac{s(x)|_1 = x}{\Gamma \vdash s(x) \triangleright x}\ (a)$$

対応するヒント

$$\dfrac{(a,2)}{(c,\_)} \qquad \dfrac{(a,1)\ \dfrac{(a,2)}{(c,\_)}}{(d,1)} \qquad (a,1)\ \dfrac{\dfrac{(a,21)(a,1)}{(d,2)}}{(d,1)}$$

2.3 局所合流性

項書換え系において局所合流性の検査は一般に危険対の合流性を検査すれば十分である。そこでクライアント側では危険対を計算し、危険対のそれぞれの正規形を求めることにより合流性を証明する。図 1 で示したように、局所合流性の検査時には停止性がすでに示されている。従って、正規形は必ず存在する。停止性の場合と同じようにクライアント側とサーバ側のオーバーヘッドを減らすためにクライアントはサーバへ危険対と、危険対合流の証明のヒントを送る。具体的にはある危険対 (s,t) について下のように (適用規則、適用位置) のペアのシーケンスがその役割を果たす。

$$s \to_{(R_1,p_1)} \cdots \to_{(R_n,p_n)} v \ {}_{(R'_m,p'_m)} \leftarrow \cdots {}_{(R'_1,p'_1)} \leftarrow t$$

$$\frac{s|_p = t\ (p \neq \epsilon)}{\Gamma \vdash s \triangleright t}\ (LPO_a)$$

$$\frac{\Gamma \vdash s|_p \triangleright t\ (s|_p \neq t)}{\Gamma \vdash s \triangleright t}\ (LPO_b)$$

$$\frac{\Gamma \vdash f(s_1,\ldots,s_m) \triangleright t_1, \ldots, \Gamma \vdash f(s_1,\ldots,s_m) \triangleright t_n,\ f > g \in \Gamma}{\Gamma \vdash f(s_1,\ldots,s_m) \triangleright g(t_1,\ldots,t_n)}\ (LPO_c)$$

$$\frac{s_1 = t_1, \ldots, s_{i-1} = t_{i-1},\ \Gamma \vdash f(s_1,\ldots,s_m) \triangleright t_{i+1}, \ldots, \Gamma \vdash f(s_1,\ldots,s_m) \triangleright t_n, \Gamma \vdash s_i \triangleright t_i}{\Gamma \vdash f(s_1,\ldots,s_m) \triangleright f(t_1,\ldots,t_m)}\ (LPO_d)$$

図 2: 推論規則

サーバは危険対と、危険対合流の証明ヒントを受け取ると、等式系から危険対を計算し、クライアントから送られてきた危険対が正しいか検査する。その後、証明のヒントにしたがって危険対が合流することを再証明する。以上によりサーバ側での安全な局所合流性の証明が実現できる。

2.4 等式系と項書き換え系の等価性

2.2、2.3 節の停止性、局所合流性の場合と同じように、等式系 $E = \{l_1 \approx r_1, \ldots, l_n \approx r_n\}$ と項書換え系 $R = \{l'_1 \to r'_1, \ldots, l'_m \to r'_m\}$ の等価性のサーバ側での検査にもクライアントからのヒントが必要となる。$s \leftrightarrow^*_R t \Rightarrow E \vdash s \approx t$ のサーバ側での検査にはクライアントから証明を送ってもらい、その証明を検査することとする。また、$E \vdash s \approx t \Rightarrow s \leftrightarrow^*_R t$ のサーバ側での検査には、証明の手順をクライアントから送ってもらい、手順に従いサーバ側で再証明する。それぞれについてみていこう。

- $s \leftrightarrow^*_R t \Rightarrow E \vdash s \approx t$

  $\forall i \in \{1,\ldots,m\}.l'_i \leftrightarrow^*_E r'_i$ を証明すればよい。完備化のステップは危険対を計算しそれらの正規形を計算し合流しない場合は新たな規則として R に加えることにより進む。したがって、どの規則とどの規則から新たな規則が生まれたかを全て記録し、最終的にこれらの新たに加わった規則をたどることにより $\forall i \in \{1,\ldots,m\}.l'_i \leftrightarrow^*_E r'_i$ の証明を構築することができる。図 3

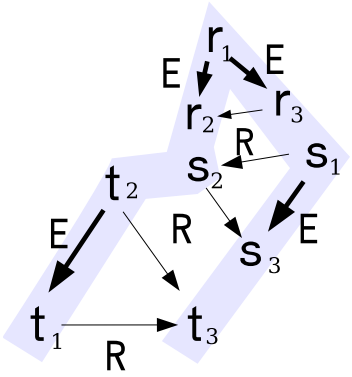

図 3: 完備化により新たに規則が加わる様子

  図 3 の要領でできる証明は以下のようになる。

  $$l'_i = t_1({}_E\!\leftarrow \mid \to_E)\ldots({}_E\!\leftarrow \mid \to_E)t_l = r'_i$$

  したがって具体的には $T = [t_1,\ldots,t_l]$ と、(適用規則、適用位置、方向) のシーケンス

  $$H = [(E_1,p_1,D_1),\ldots,(E_{l-1},p_{l-1},D_{l-1})]$$

  のペア (T,H) を R の要素それぞれについて添付することによりサーバでは安全な証明の検査が可能である。

- $E \vdash s \approx t \Rightarrow s \leftrightarrow^*_R t$

  クライアントでは完備化手続きの後に $E \vdash s \approx t \Rightarrow s \leftrightarrow^*_R t$ の証明を行う。すでに項書換え系は完備であるのでこちら側は簡単である。証明すべきものは $\forall i{\in}\{1,\ldots,n\}\ \exists v.\ l_i \to^*_R v\ \ {}^*\!\leftarrow_R r_i$ であるため局所合流性の証明の手順と同じようにヒントをつくるとしてここでは省略する。

3 計算量

2 節の構成において、サーバ側の計算量を求める。クライアント側から送られる等式系と対応する項書き換え系を

$$E = \{s_1 \approx t_1, \ldots, s_l \approx t_l\}$$

$$R = \{l_1 \to r_1, \ldots, l_n \to r_n\}$$

とし、$|R| = \Sigma_{i=1}^{n}|l_i| + |r_i|$ とする。また、局所合流性の検査のヒント (適用規則、適用位置) のシーケンスを

$$L = \{(l'_1 \to r'_1, p'_1), \ldots, (l'_m \to r'_m, p'_m)\}$$

とし、$|L| = \Sigma_{j=1}^{m}(|l'_j| + |r'_j| + |p'_j|)$ とする。

$s \leftrightarrow_R^* t \Rightarrow E \vdash s \approx t$ の証明は

$$P = \{(T_1, H_1), \ldots, (T_n, H_n)\}$$

$$T_i = \{t_{i_1}, \ldots, t_{i_y}\}$$

$$H_i = \{(l''_1 \to r''_1, p''_1, D_1), \ldots, (l''_z \to r''_z, p''_z, D_z)\}$$

で $|T_i| = \Sigma_{j=1}^{y}|t_{i_j}|, |H_i| = \Sigma_{i=1}^{z}|l''_i| + |r''_i| + |p''_i|, |P| = \Sigma_{i=1}^{n}|T_i| + |H_i|$ また $E \vdash s \approx t \Rightarrow s \leftrightarrow_R^* t$ の検査のヒントを

$$M = \{(l'''_1 \to r'''_1, p'''_1), \ldots, (l'''_x \to r'''_x, p'''_x)\}$$

$|M| = \Sigma_{j=1}^{x}(|l'''_j| + |r'''_j| + |p'''_j|)$

最後に、単一化には線形アルゴリズムを用い、R の危険対の集合を CP(R) とすると計算量は以下のようになる。

- 停止性検査 $O(|R|)$
- 危険対の計算 $O(n \cdot |R|^2)$
- 危険対の照合 $O(|CP(R)|)$
- 危険対の合流性検査 $O(|L|)$
- $E \vdash s \approx t \Rightarrow s \leftrightarrow_R^* t$ $O(|P|)$
- $s \leftrightarrow_R^* t \Rightarrow E \vdash s \approx t$ $O(|M|)$

以上のように最大の計算量でも $O(n \cdot |R|^2)$ である。またその他の計算量はヒントの大きさで抑えられているため、送られてくるヒントを見ればサーバはどれくらい計算が必要かがわかる。サーバはこれを指標にその等式系を受け付けるかどうかを決めれば安全に検査することが可能である。

以上のように、信頼できないクライアントに対してもこの構成は計算量的に安全であることがわかった。

4 関連研究

proof carrying code

George C. Necula らの研究 [10] では信頼できないコードに対して、コード消費者のポリシーを満たしているという証明をつけることにより安全にそのコードを動かすことができる機構を提唱している。コードに証明をつけることによりコード消費者側では自らそのコードが自分のコンピュータのポリシーを越えたことをしないことを検査するよりはるかに簡単に安全性の検査ができる。この証明を付加することにより検査をする側では簡単に検査が可能であるというアイディアにおいて本研究と関連がするところが多い。また、塚田恭章らの研究 Interactive and probablistic proof of mobile code safety[11] において、PCC での証明がプログラムに比べて巨大になるという欠点を簡単な通信により実際に証明を流通させずに、証明の存在を保証する手法により解決している。

5 まとめと今後の課題

われわれは、クライアントから等式系の他に、対応する項書き換え系、各検査のヒントを付加することにより安全な計算量での検査ができる構成を示した。将来の課題として本システムの実装、そして PDIP(Proof Document with Interactive Prover) との連携があげられる。

参考文献

[1] HOL homepage. http://hol.sourceforge.net/

[2] Isabelle homepage. http://www.cl.cam.ac.uk/Research/HVG/Isabelle/

[3] Coq homepage. http://coq.inria.fr/

[4] Mizar homepage. http://mizar.org/

[5] PDIP homepage. http://www.lambda.cs.titech.ac.jp/pdip/

[6] WikiWiki homepage. http://www.c2.com/cgi/wiki?WelcomeVisitors

[7] WikiPedia http://www.ja.wikipedia.org/wiki/

[8] M. H. A. Newman. On theories with a combinatorial definition of 'equivalence'. *Annals of Mathematics*, 43(2):223-243, 1942

[9] Franz Baader and Tobias Nipkow, Term Rewriting and All That.

[10] G Necula. Proof-carrying code. In ACM *Sympoium on Principles of Programing Languages(POPL)*,1997

[11] Yasuyuki Tsukada, Interactive and probablistic proof of mobile code safety,*Automated Software Engineering*,Vol. 12, No2(April 2005)237-257