時間付き多重集合書き換えに対する記号的解析とそのプロトコル解析への応用

Symbolic Analysis of Timed Multiset Rewriting and Its Application to Protocol Analysis

萩谷 昌己 † Masami HAGIYA hagiya@is.s.u-tokyo.ac.jp
山本 光晴 †† Mitsuharu YAMAMOTO mituharu@math.s.chiba-u.ac.jp
ジャン-マリ コタン † Jean-Marie COTTIN jcottin@is.s.u-tokyo.ac.jp

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

多重集合書き換え (multiset rewriting) のフレームワークに時間の経過を付加することにより，時間付き多重集合書き換え (timed multiset rewriting) のフレームワークを定義する．多重集合の各要素がクロックを持ち，書き換え規則もクロックの値を条件としている．さらに，時間付き多重集合書き換えの記号的解析手法について議論し，プロトコル解析への応用について検討する．

1 はじめに

萩谷たちは，認証プロトコルの検証のために，主体 (principal) の局所状態の集合とメッセージの集合の書き換えに基づくフレームワークを提案している [3, 9]．ネットワーク全体の状態は，ネットワーク上を流れたメッセージの集合 M と、主体 (principal) の局所状態の集合 S の組 $\langle M, S\rangle$ によって表される．Cervesato たちも同様のフレームワークを提案している [2] が，彼らは集合の代わりに多重集合を用いている．場合によっては集合よりも多重集合の方がきめ細かい仕様記述や検証が可能である．多重集合の書き換えに基づく汎用的なフレームワークとしては rewriting logic が知られている [7].

プロトコルによっては，タイムアウトなどの時間の概念が入っているものがある．また，暗号解読など，時間によって成功の確率が変わる操作も考えられる．さらに，CPU 時間は DoS 攻撃の対象となるリソースの一つである．このように，ネットワークセキュリティに関してより実際的な状況を解析し，攻撃の可能性を未然に防ぐためには，時間の概念の入った形式的なフレームワークが必要である．

以上のような考えに従い，本論文では，時間概念の入った多重集合書き換えのフレームワークを提案し，それに基づく解析の簡単な例を示す．

近年，timed automaton, hybrid automaton, timed Petri net など，離散系と連続系を融合したハイブリッド系の研究が盛んである [1, 4, 10]．多重集合に関しても，timed rewriting logic が提案されている [8]．多重集合の書き換え系は，多重集合を状態とする状態遷移系であり，一般に状態集合は無限であるので，region 解析などの有限的な状態探索による検証は難しい．また，実際的な状況では，系は不定のパラメータを含んでおり，系の挙動がどのようにパラメータに依存するかを解析したい場合が多い [5].

本論文では，時間付き多重集合書き換え系の形式的なフレームワークを提案し，それに従った解析の事例を示す．特に，不定パラメータが入った書き換え系に対して記号的な解析を行い，特定の要素の多重度を解析する方法を示す．このフレームワークは，証明支援系による形式的証明のベースとして用いるとともに，将来的には有限的な状態探索による検証方法と融合したいと考えている．

2 時間付き多重集合書き換え

本節では，単純化された時間付き多重集合書き換えについて簡単に説明する．この体系は，flow に関する制約を除けば，timed Petri net [10] と等価である．実際のプロトコルの解析に用いる，より本格的な時間付き多重集合書き換えについては，6 節で簡単に紹介する．

時間付き多重集合 (timed mutiset) とは，多重集合

の各要素にクロックが割り当てられたものである．例えば，$\{a_1{:}t_1, a_2{:}t_2, a_3{:}t_3\}$ という時間付き多重集合は a_1, a_2, a_3 という要素から成り，それぞれに t_1, t_2, t_3 というクロックが割り当てられている．クロックの値は非負の実数とする．$\{a_1, a_2, a_3\}$ は多重集合であるので，a_1, a_2, a_3 には重複があっても構わない．重複した要素に対しては別々のクロックが割り当てられる．より厳密には，多重集合の可能な要素の集合を S，自然数 (非負整数) の集合を $\mathbf{N}$，非負実数の集合を $\mathbf{R}^{\geq 0}$ としたとき，時間付き多重集合とは，S から $\mathbf{N}$ への関数 m と，集合 $\{(s,i)|s \in S,\ i \in \mathbf{N},\ 0 \leq i < m(s)\}$ から $\mathbf{R}^{\geq 0}$ への関数 t の組 $\langle m, t\rangle$ と定義される．

時間付き多重集合の書き換えは，二種類の規則によって定義される．jump は，多重集合の要素を置き換える規則であり，次のような形をしている．

$$r : L \ \rightarrow \ R \ \textbf{if} \ C$$

r はこの規則に付けられたラベルである．この規則は，多重集合の部分集合で L にマッチし，C を満たすものを R で置き換えることを意味している．

L は $a{:}t$ という形の式の並びである．ここで，a は S の要素であり，t は非負実数を動く変数である．後者をクロック変数と呼ぶ．本論文では L は空ではないと仮定する．C は L に現れるクロック変数に対する制約であり，算術演算子，比較演算子，論理演算子などによって表現される．R は $a{:}0$ という形の式の並びである．置き換えた後の要素のクロックは 0 にリセットされる．

もう一つの種類の規則は flow である．flow は，時間付き多重集合のクロックを一様にある正実数 d だけ進める規則である．すなわち，時間付き多重集合の中の要素 $a{:}u$ を $a{:}u+d$ で置き換える．ただし，S の各要素に対して，

$$a{:}t \mid C$$

という形の制約を設けることができる．ここで，t はクロック変数，C は t に対する制約である．flow が適用されて $a{:}u$ という要素が $a{:}u+d$ という要素に変化したとき，$u \leq t \leq u+d$ を満たす任意の t に対して C が成り立っていなければならない．そのような条件が成り立っている限りにおいて flow が適用可能である．

以下は，SYN パケットによる攻撃 (SYN Attack) をモデル化したものである．

$\mathtt{R0} : \mathtt{attack}{:}t \rightarrow \mathtt{attack}{:}0, \mathtt{SYN}{:}0 \ \ \textbf{if} \ t \geq \mathtt{C}$

$\mathtt{R1} : \mathtt{SYN}{:}u \ \rightarrow \ \mathtt{SYNACK}{:}0,\ \mathtt{serve}{:}0$
$\qquad \textbf{if} \ \mathtt{L} \leq u \leq \mathtt{U}$

$\mathtt{R2} : \mathtt{serve}{:}t \ \rightarrow \ \ \textbf{if} \ t = \mathtt{T}$

$\mathtt{R3} : \mathtt{serve}{:}t,\ \mathtt{ACK}{:}u \ \rightarrow \ \textbf{if} \ t < \mathtt{T} \ \textbf{and} \ u \geq \mathtt{L}$
$\mathtt{serve}{:}t \mid t \leq \mathtt{T}$

3 証明木

時間付き多重集合書き換え (timed multiset rewriting) の過程を形式化するために，証明木および時間付き証明木を定義する．本節では最後に証明木 (proof tree) の帰納的定義を与える．

一般に jump 規則は以下のような形をしている．

$$r : a_1{:}t_1, \cdots, a_m{:}t_m \rightarrow b_1{:}0, \cdots, b_n{:}0 \ \textbf{if} \ C$$

証明木を作るために，m 個の関数記号 $r_1, \cdots, r_m$ を導入する．r_1 は引数が n 個の関数記号として用いる．$r_2, \cdots, r_m$ は引数が 0 個の関数記号として用いる．

また，多重集合の要素 a_i や b_j は定数記号として用いる．ここでは，定数記号と引数が 0 の関数記号とは区別していることに注意されたい．

D を，関数記号 r_i および定数記号 a_i, b_j から作られる項とする．D における関数記号の position 全体の集合を $RP(D)$ によって表す．ここで，position とは正の自然数の並びで，項の root から部分項へ至る経路を表している．

E を $RP(D)$ 上の同値関係とする．ここでは，E は $RP(D)$ 上の同値類の集合とする．すなわち，E は $RP(D)$ の分割である．

証明木は，以上のような D と E の組 $\langle D, E\rangle$ として定義される．以下に証明木の帰納的定義を与えるが，その前に例を示す．

先の SYN attack の例に対して以下のような項 D を作ることができる．

```
R0(R0(R0(R0(attack,
              R1(SYNACK,serve)),
            SYN),
          R1(SYNACK,serve)),
        R1(SYNACK,R2))
```

ここで，R0，R1，R2 は，$\mathtt{R0}_1$，$\mathtt{R1}_1$，$\mathtt{R2}_1$ の略とする．E を自明な同値関係 (すべての position は自分のみと同値) とすると，$\langle D, E\rangle$ は証明木になる．

定数記号のいくつかを初期定数記号とする．上の例では，attack を初期定数記号としている．

以下は，証明木の帰納的定義である．

- a が初期定数記号ならば，$\langle a, \emptyset \rangle$ は証明木である．
- 次の条件が満たされるとき，
  - $\langle D, E \rangle$ は証明木である．
  - $r : a_1{:}t_1, \cdots, a_m{:}t_m \to b_1{:}0, \cdots, b_n{:}0$ **if** C は jump である．
  - $1 \leq i \leq m$ に対して，p_i は D における a_i の position である．

以下も証明木となる．

$$\langle D[p_1 \leftarrow r_1(b_1, ..., b_n), p_2 \leftarrow r_2, \cdots, p_m \leftarrow r_m], \\ E \cup \{\{p_1, \cdots, p_m\}\}\rangle$$

ここで，$D[p_i \leftarrow r_1(b_1, ..., b_n), \cdots]$ は，D の position p_i の定数記号 (すなわち a_i) を $r_1(b_1, ..., b_n)$ もしくは r_i で置き換えることを表している．

4 時間付き証明木

本節では，時間付き証明木 (timed proof tree) を定義する．まず，証明木における predecessor の概念を定義する．

$\langle D, E \rangle$ を証明木とする．$AP(D)$ を D の中の定数記号の position の集合とする．関数 $pred$ は $RP(D) \cup AP(D)$ から $E \cup \{\bot\}$ への関数で，position の親の position の同値類を与える．position $p \in RP(D) \cup AP(D)$ に対して，$p = qi$ ならば，$pred(p) = [q]$ と定義する．ここで，$[q]$ は q の同値類を表している．また，root position ϵ に対しては，$pred(\epsilon) = \bot$ と定義する．

本節の以下の部分では，

$$r : a_1{:}t_1, \cdots, a_m{:}t_m \to b_1{:}0, \cdots, b_n{:}0 \ \textbf{if} \ C$$

は jump 規則であるとする．$p \in RP(D)$ に対して，$[p] = \{p_1, \cdots, p_m\}$ かつ $D[p_1] = r_1(b_1, \cdots, b_n)$ かつ $D[p_i] = r_i$ $(2 \leq i \leq m)$ が成り立つとき，r は p に対する jump であるという．いうまでもなく，$D[p_1]$ は D の p_1 における部分項を表している．

$time$ を $E \cup \{\bot, \top\}$ から $\mathbf{R}^{\geq 0}$ への関数とする．r が $p \in RP(D)$ に対する jump であるとき，$time([p])$ は r による書き換えが起きた時刻を表している．また，$time(\bot)$ は全体の書き換えが始まった時刻，$time(\top)$ は現在の時刻を表している．

関数 $time$ が与えられたとき，position $p \in RP(D) \cup AP(D)$ に対して，p の持続時間 $dur(p)$ を以下のように定義する．

$$dur(p) = time([p]) - time(pred(p)) \quad (p \in RP(D))$$
$$dur(p) = time(\top) - time(pred(p)) \quad (p \in AP(D))$$

証明木 $\langle D, E \rangle$ と $E \cup \{\bot, \top\}$ から $\mathbf{R}^{\geq 0}$ への関数 $time$ に対して以下の条件が満たされるとき，$\langle D, E, time \rangle$ を時間付き証明木という．

- 任意の $p \in RP(D) \cup AP(D)$ に対して，$dur(p) \geq 0$ が成り立つ．
- r が $p \in RP(D)$ に対する jump であり，$[p] = \{p_1, \cdots, p_m\} \in E$ ならば，C の具体化である $C[t_1 \leftarrow dur(p_1), \cdots, t_m \leftarrow dur(p_m)]$ が成り立つ．
- r が $p \in RP(D)$ に対する jump であり，$a_i{:}t|C$ ならば，任意の $u \leq dur(p)$ に対して $C[t \leftarrow u]$ が成り立つ．
- $p \in AP(D)$，$D[i] = a$，$a{:}t|C$ ならば，任意の $u \leq dur(p)$ に対して $C[t \leftarrow u]$ が成り立つ．

5 position 解析

時間付き証明木の概念は，証明木の帰納的定義と $time$ に関する制約によって厳密に定まる．従って，時間付き証明木に関する推論を行うためには，まず証明木の可能性を網羅して，個々の可能性に関して $time$ の制約を求め，時間付き証明木に対して求めたい性質を $time$ の制約から導く．

個々の証明木は，その position と position 間の同値関係によって定まるので，証明木の可能性を網羅するためには，position の集合の可能性を網羅することが必要である．そのために，証明木の中の定数記号と関数記号を述語とする論理プログラムを考える．先の例に対しては，次のようなプログラムが考えられる．なお，以下の例では R3 は考えないこととする．

$\mathtt{R0}(\epsilon)$.
$\mathtt{attack}(p1) \vee \mathtt{R0}(p1) :- \mathtt{R0}(p)$.
$\mathtt{SYN}(p2) \vee \mathtt{R1}(p2) :- \mathtt{R0}(p)$.
$\mathtt{SYNACK}(p1) :- \mathtt{R1}(p)$.
$\mathtt{serve}(p2) \vee \mathtt{R2}(p2) :- \mathtt{R1}(p)$.

例えば，$\mathtt{RO}(p)$ は position p の記号が RO であることを意味している．

以上の論理プログラムは，:– の左辺である結論部に選言が現れているので，いわゆる disjunctive な論理プログラムになっている [6]．従って，一般に minimal model は一意的には定まらない．minimal model の個々の可能性が証明木の可能性に対応している．

上の例の場合，RO を除いて各述語が結論部に一度しか現れていない．RO についても，結論部のパターンは排他的である．従って，上の例に対しては各節の逆が成り立ち，さらに以下が成り立つ．

$$\mathtt{RO}(p) \;\Rightarrow\; p = \epsilon \vee \exists q.\, p = q1.$$
$$\mathtt{SYN}(p) \;\Rightarrow\; \exists q.\, p = q2.$$
$$\mathtt{SYNACK}(p) \;\Rightarrow\; \exists q.\, p = q1.$$
$$\mathtt{serve}(p) \;\Rightarrow\; \exists q.\, p = q2.$$

以上のような解析をここでは position 解析と呼ぶ．一般には E を表す述語を考える必要がある．また，上の例のような厳密な解析は一般にはできない．

position 解析の結果を用いて，次のような解析を行うことができる．

> $\mathtt{serve}(p)$ ならば，$p = q22 \wedge \mathtt{RO}(q)$ を満たす q が唯一存在する．
> $\mathtt{RO}(q)$ ならば，$q = 1 \cdots 1 = 1^n$．
> $time([1^n]) + \mathtt{C} \le time([1^{n+1}])$．
> $\mathtt{serve}(q22)$，$\mathtt{RO}(q)$ ならば，$\mathtt{L} \le time(\top) - time([q]) \le \mathtt{T} + \mathtt{U}$．
> 従って，$time(\top) - (\mathtt{T} + \mathtt{U}) \le time([q]) \le time(\top) - \mathtt{L}$．

以上の議論より，次のことが導かれる．

> 任意の時間付き証明木に対して，serve の多重度 (定数記号 serve の出現回数) は，高々 $(\mathtt{T} + \mathtt{U} - \mathtt{L})/\mathtt{C} + 1$ である．

以上の解析において，T，U，L は具体的な数値である必要はなく，不定パラメータのままで構わない．

以上の解析においては，自然数から position への適当な関数 f が存在して，関数記号 r の position が $f(i)$ と表され，$time([f(i)]) + c \le time([f(i+1)])$ という制約が成り立っている．また，定数記号 a に対して，a の position と r の position の間に 1:1 の関係 g が存在し，さらに a の position p に対して，$time([g(p)])$ の上下限が定まっている．このような場合，a の多重度の上界を求めることができる．

6 本格的な時間付き多重集合書き換え

実際のプロトコルを表現するためには，jump 規則はメッセージの内容や主体の局所状態などを参照して，新たなメッセージや局所状態を生成できなければならない．また，ノンスなどのデータを新たに生成する必要もある．従って，本格的な時間付き多重集合書き換えの jump 規則は，

$$L \to R\ (\exists \alpha, \beta, \cdots)\ \textbf{if}\ C$$

という形をしている．L は $e{:}t$ という式の並びである．t はクロック変数であり，e は変数を含むパターンとする．R は $e{:}0$ という式の並びであり，$\alpha, \beta, \cdots$ はこの規則によって新たに生成されるデータである．e は L に含まれる変数や $\alpha, \beta, \cdots$ を含むパターンである．特に L に含まれるクロック変数の値を参照することができるとする．C は L に含まれる変数に関する制約である．

7 おわりに

本論文では，クロックの入った多重集合書き換えについて述べ，形式的な証明のベースになるフレームワークを提案した．特に時間付き証明木の定式化を行った．さらに position 解析を提案し，不定パラメータを含む書き換え系の多重度解析が行えることを示した．今後は，より一般的な記号的解析の手法を確立するとともに，最後に述べた本格的な時間付き多重集合書き換えを用いたより実際的な解析へと研究を進めて行きたい．

参考文献

[1] R. Alur, et al. TCS 126(2), 1994, pp.183–235.
[2] I. Cervesato, et. al. IEEE CSFW'00, pp.35–51.
[3] M. Hagiya, et al. 1999. http://nicosia.is.s.u-tokyo.ac.jp/pub/staff/hagiya/ssr99/protveri.ps
[4] T. A. Henzinger. IEEE LICS'96, pp.278–292.
[5] T. S. Hune, et al. TACAS'01.
[6] J. Lobo, et al. *Foundations of Disjunctive Logic Programming,* MIT Press, 1992.
[7] J. Meseguer. TCS 96, 1992, pp.73–155.
[8] P. C. Ölveczky, et al. ENTCS 36, 2001.
[9] 高橋孝一, 戸田洋三, 萩谷昌己. JSSST 2000.
[10] M. Tanabe. LNCS 1248, 1997, pp.156–174.