抽象到達可能性検査の Priced Timed Automaton への応用

Application of Abstract Reachability to Priced Timed Automata

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
Graduate School of Information Science and Technology, University of Tokyo

priced timed automaton とは、制御の離散的振舞いと時間の連続的振舞いをモデル化した timed automaton にコストの概念を加えたものである。本稿では我々がこれまで研究してきた抽象到達可能性検査を、priced timed automaton の最小コスト問題におけるリージョン解析やゾーン解析に応用する。また、抽象到達可能性検査アルゴリズムにおいて、A* アルゴリズムに類似した最適化を付加し、これの priced timed automaton の最小コスト問題への適用について考察を行う。

1 はじめに

我々は、文献 [4] において抽象到達可能性検査アルゴリズムを定式化し、それに関して成り立ついくつかの性質を示した。抽象到達可能性検査アルゴリズムは状態間に順序関係を持つラベル付き遷移システム上の抽象アルゴリズムであり、抽象モデル検査アルゴリズムの 1 つである Covering Graph Construction や、最短路問題やフロー解析を含む抽象グラフ探索アルゴリズムを表現可能なものである。種々のアルゴリズムを統一的に扱えるような抽象アルゴリズムの枠組を与えることにより、計算機上の証明検証系での形式的検証を容易にしようというのがねらいである。

本稿では、抽象到達可能性検査アルゴリズムによって表現できるアルゴリズムの範囲をさらに広げるべく、priced timed automaton の最小コスト問題におけるリージョン解析とゾーン解析を表現する。timed automaton[1] とは、制御の離散的振舞いと時間の連続的振舞いをオートマトンの形でモデル化したもので、priced timed automaton はそれにコストの概念を導入したものである。

また、最短路問題の最適化アルゴリズムの一つである A*アルゴリズムを抽象到達可能性検査アルゴリズムの上で定式化し、その正当性証明を行う。A*を抽象アルゴリズムの上で定式化することにより、様々な具体アルゴリズムの上での A*の導入を容易にすることが可能となる。特に proced timed automaton における最小コスト問題に応用することができる。

2 Linearly Priced Timed Automaton

以下の Linearly Priced Timed Automaton(LPTA と略す) の定義は文献 [2] による。

定義 1 C をクロックの有限集合、Act をアクションの集合とする。LPTA はロケーションの有限集合 L, 初期ロケーション $l_0(\in L)$, 辺 $E(\subseteq L\times\mathcal{B}(C)\times Act\times 2^C\times L)$, ロケーションへの不変式の割り当て $I : L\to\mathcal{B}(C)$, ロケーションと辺へのコストの割り当て $P : (L\cup E)\to\mathbb{N}$ による組 (L, l_0, E, I, P) により定義される。$(l, g, a, r, l')\in E$ のことを $l\xrightarrow{g,a,r} l'$ と書く。

ここで、$\mathcal{B}(C)$ はクロックに関する制約であり、$x\bowtie n(n\in\mathbb{N})$ という式の論理積で表現される。($\bowtie$ は $<, >, \leq, \geq$ のいずれか。)

LPTA の状態はロケーション l とクロックの割り当て $u : C\to\mathbb{R}$ の組 (l, u) で表わされる。初期状態ではロケーションが l_0 でクロックは全て 0 に割り当てられる。LPTA の遷移は通常のオートマトンのようにロケーションを移るものと、ロケーションはそのままで時間を一様に経過させるもの 2 種類がある。辺を通ってロケーションを移るときにはその辺に割り当てられたクロックの部分集合の値が 0 にリセットされる。いずれの遷移も、移った先でロケーションに割り当てられた不変条件を満たすようにしなければならない。

- $(l, u)\xrightarrow{a,p}(l', u')$ ただし、$l\xrightarrow{g,a,r} l'$, $u\in g$, $u' = [r\mapsto 0]u$, $u'\in I(l')$, $p = P((l, g, a, r, l'))$
- $(l, u)\xrightarrow{\epsilon(d),p}(l, u+d)$ ただし、$u+d\in I(l)$ かつ $p = P(l) * d$

遷移を表す矢印の上にある二つ目の要素 p は、コストの変化の様子を示している。辺を伝ってロケーションを移るときにはその辺に割り当てられたコストが、ロケーションがそのままで時間を経過させるときには (ロケーションに割り当てられたコスト $\times$ 経過時間) 分のコストがそれぞれかかる。

到達可能なロケーション l に対し、初期状態 l_0 から l に到達するまでのトータルのコストの最小 (一般には下限) がいくつになるか、また、どのような経路を通れば最小コストが実現できるかが問題になる。これを**LPTA の最小コスト問題**と呼ぶ。文献 [3] ではこのような問題の例として、空港に近付きつつある飛行機が着陸するまでの遷移を LPTA で表わし、燃料コスト最小で着陸するにはどのような手順を踏めばよいかという問題を挙げている。

クロックを含む timed automaon の状態は一般に無限集合になるため、そのままでは状態の数え上げによる解析を行うことはできない。そこで、有限の抽象状態上の抽象遷移を考え、その上で解析を行うという手法が用いられる。そのような手法の代表的な例がリージョン解析とゾーン解析である。

リージョン解析ではクロック値の空間を、クロック値の整数部分、および小数部分の大小関係が一致するもので分割したものを抽象状態とし、ゾーン解析では線形な不等式の共通部分で表される部分を抽象状態とする。いずれの場合もオートマトン中の条件に現れる定数を超える部分は同一視することにより、抽象状態が有限になるようにしている。

LPTA についても同様にリージョン解析とゾーン解析の手法が提案されている [2, 3]。これらの手法ではリージョンやゾーンにコストやその時間変化量に応じた線形な重みを付加することにより、コスト付きリージョンやコスト付きゾーンを抽象状態とすることで LPTA の最小コスト問題の解析を行う。文献 [2] におけるコスト付きリージョンを用いたアルゴリズムを図 1 に示す。$\mathsf{mincost}(R)$ はコスト付きリージョン R の中での最小コストである。ゾーン解析についても同様のアルゴリズムが提案されている [3]。

```
COST := ∞
PASSED := ∅
WAITING := {(l_0, R_0)}
while WAITING ≠ ∅ do
  select (l, R) from WAITING
  if l ≠ l_g and mincost(R) < COST then
    COST := mincost(R)
  if for all (l, R') in PASSED: R' ⋢ R then
    add (l, R) to PASSED
    for all (l', R') such that (l, R) → (l', R'):
      add (l', R') to WAITING
return COST
```

図 1: LPTA のリージョン解析

3 抽象到達可能性検査アルゴリズムによる表現

我々は文献 [4] において、状態間に順序関係を持つラベル付き遷移システム上の抽象到達可能性検査アルゴリズム (図 2) を定義し、抽象モデル検査アルゴリズムの 1 つである Covering Graph Construction や、最短路問題やフロー解析を含む抽象グラフ探索アルゴリズムが、抽象到達可能性検査アルゴリズムによって表わせることを示した。

```
 1:  S := I;
 2:  U := S;
 3:  while S ≠ ∅ do
 4:    S から一つ要素 s を取り除く;
 5:    for each s --a--> t do
 6:      if ∃t_C ∈ U. t_C ≥ t then (* Cover *)
 7:        (* 何もしない *);
 8:      else if C_G then (* Generate *)
 9:        t_G ∉ U, t ≤ t_G を満たす t_G を作る;
10:        t' ≤ t_G となる t' ∈ U を
           U と S からいくつか取り除く;
11:        t_G を S と U に加える;
12:      else (* Step *)
13:        t' ≤ t となる t' ∈ U を
           U と S からいくつか取り除く;
14:        t を S と U に加える;
```

図 2: 抽象到達可能性検査アルゴリズム

上記アルゴリズムについて、以下のような性質が成り立つ [4]。

性質 1 ラベル付き遷移システム上の初期状態から到達可能な遷移列

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$$

$(s_0 \in \mathcal{I})$ が,「$\forall i.\ 0 \le i \le n-1 \Longrightarrow (\forall t.\ s_i \le t \Longrightarrow (\exists u.\ t \xrightarrow{a_i} u \text{ and } s_{i+1} \le u))$」という条件 (*) を満たすならば，アルゴリズム終了時に $s \in U$ となる s が存在して，$s_n \le s$ が成立する.

すなわち、到達可能な状態そのものを網羅するのではなく、到達可能な状態 s をそれよりも一般的な状態 ($s \le s'$ となる s') で代表させることにより、到達可能な状態を網羅したものを表現しようとするものである。

前節で述べた LPTA のリージョン解析アルゴリズムを、抽象到達可能性検査アルゴリズムによって表現すると次のようになる。

- ラベル付き遷移システム

$$\begin{cases} \text{状態 } \mathcal{S} & \stackrel{\text{def}}{=} & \text{リージョン解析の抽象状態} \\ \text{遷移 } \xrightarrow{a} & \stackrel{\text{def}}{\Longleftrightarrow} & \text{リージョン解析の抽象遷移} \\ \text{初期状態 } \mathcal{I} & \stackrel{\text{def}}{=} & \{(l_0, R_0)\}\ (\subseteq \mathcal{S}) \\ s \le s' & \stackrel{\text{def}}{\Longleftrightarrow} & \exists l\, R\, R'.\ s = (l, R), \\ & & \quad s' = (l, R'), \text{ and } R \ge R' \end{cases}$$

- Generate について

$$C_G \stackrel{\text{def}}{\Longleftrightarrow} \mathit{false}$$

- U と S からの要素の削除

  Generate, Step において，U と S からは何も取り除かない

リージョン解析の抽象状態はロケーション l とコスト付きリージョン R の組 (l, R) で表わされる。また、$R \ge R'$ はコスト付きリージョン間の順序で、R と R' がコストを無視したリージョンとしては一致し、コストについてはリージョン上の各点で「(R 上でのコスト)$\ge$(R' 上でのコスト)」という順序がついていることを意味する。より少ないコストを達成する状態を「より一般的」としているため、状態間の大小関係とコストの大小関係の間に逆転が生じていることに注意が必要である。

なお、スペースの都合でゾーン解析の場合の定義は省略するが、リージョン解析の場合と同様、抽象状態が状態に、抽象遷移が遷移にそのまま対応する。

補題 1 抽象グラフ探索、LPTA のリージョン解析・ゾーン解析のいずれも、抽象到達可能性検査で表したときに性質 1 における条件 (*) を満たす。

以下では簡単のために性質 1 における条件 (*) が成り立つ場合のみを考える。

ここまでで LPTA のリージョン解析アルゴリズム (図 1) の動きを抽象到達可能性検査アルゴリズム (図 2) でシミュレートすることはできた。性質 1 より、アルゴリズム終了後に求まるものは次の条件を満たす状態集合 U である。

$$\forall s \in \mathcal{S}.\ s \text{ が初期状態から到達可能} \Rightarrow \exists s' \in U.\ s \le s'$$

しかし実際に求めたいのは、ゴールのロケーション l_g に対して、$\min\{\mathsf{mincost}(R) \mid (l_g, R) \in U\}$ であり、U そのものではない。すなわち、アルゴリズムの動きは表現できていたものの、実際に求めるべき値自身はまだ表現できていなかったのである。

さらに考察すると、アルゴリズム中で状態の比較に用いていた順序 $\le$ を直接用いたのでは、ゴールに到達するまでの最小のコストを表現できないことがわかる。これは、状態 (l, R) と (l', R') は R と R' がコストを無視した場合のリージョンとして同一のものでないと比較できないのに対し、$\min\{\mathsf{mincost}(R) \mid (l_g, R) \in U\}$ では $l = l_g$ となるもの全てを比較対象としなければならないためである。しかし状態間の順序 $\le$ で比較可能なものを増やしてしまうと、今度はアルゴリズム自体が正しくなくなってしまう。

そこで、ゴールとなる状態の比較を行うために、新たに半束 $\mathcal{L}$ と、状態から半束への単調関数 γ を導入する。$\mathcal{L}$ は探索において求めるべき値の集合を表すものであり、最短路問題や LPTA の最小コスト問題では $\mathbb{R} \cup \{\bot\}$ になる。$\gamma(s)$ は、状態 s がゴールに対応するものであれば、その状態における値となり、それ以外の場合は $\bot$ になるように設定する。例えば、LPTA の最小コスト問題では、

$$\gamma(l, R) \stackrel{\text{def}}{=} \textbf{if } l = l_g \textbf{ then } -\mathsf{mincost}(R) \textbf{ else } \bot$$

となる。(大小関係を逆転させるために符号を反転させている。) このように定義すると、アルゴリズム終了時に求めるべき値は

$$\Gamma \stackrel{\text{def}}{=} \bigvee \{\gamma(t) \mid t \in U\}$$

と表現できる ($\bigvee$ は半束の要素の結び)。

定理 1

$$\forall s \in \mathcal{S}.\ s \text{ が初期状態から到達可能} \Rightarrow \gamma(s) \le \Gamma$$

[証明] 初期状態から到達可能な状態 s について性質 1 から、$s \le s'$ かつ $s' \in U$ となる s' が存在。γ の単調性から、$\gamma(s) \le \gamma(s') \le \bigvee\{\gamma(t) \mid t \in U\} = \Gamma$ □

4 A*アルゴリズム

通常の最短路問題では、A*という最適化アルゴリズムがよく知られている。我々は最短路問題を含む抽象グラフ探索アルゴリズムにおいても同様に A*アルゴリズムを定義できることを示し、さらにその正当性証明を証明検証系 HOL の上で与えた [4]。しかし抽象グラフ探索アルゴリズムを抽象到達可能性検査アルゴリズムに変換したものの上では A*を表現できていなかった。本節では抽象到達可能性検査アルゴリズムの上で A*を表現するための枠組を与える。

通常の最短路問題の A*では、各ノードにゴールまでの距離の下界を与えることにより、ゴールまでの最短路を見つけるのに必要な探索空間を狭めていた。抽象到達可能性検査では以下の条件を満たす関数 $g : \mathcal{S} \to \mathcal{L}$ を与える。

$$\forall s \in \mathcal{S}.\ \bigvee\{\gamma(s') \mid s' \text{は } s \text{から到達可能 }\} \leq g(s)$$

これは [4] の抽象グラフ探索における A*アルゴリズムでの g の条件と自然に対応する。

定理 2 上記のような条件を満たす g が与えられたとき、図 2 の 4 行目の直前の時点で、

$$\forall s \in S.\ g(s) \leq \bigvee\{\gamma(t) \mid t \in U\}$$

が成り立つとき、その時点での U に対し、初期状態から到達可能な任意の状態 s について

$$\gamma(s) \leq \bigvee\{\gamma(t) \mid t \in U\}$$

[証明] 文献 [4] の性質 1 の「証明の概略」にある通り、「$\forall s\ t\ a.\ s \in U \setminus S$ and $s \xrightarrow{a} t \Rightarrow \exists t'.\ t \leq t'$ and $t' \in U$」がループ不変式である。いま、初期状態から到達可能な状態 s について、$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$ という経路で初期状態 s_0 から $s_n = s$ に到達するとする。

次に、上記のループ不変式を用いて、以下のように状態の列 $\{s'_i\}_{0 \leq i \leq k} (k \leq n)$ を構成する。

1. $s'_0 = s_0$; $\quad i := 0$
2. $i = n$ か $s'_i \notin U \setminus S$ となったら $k := i$ として終わり。そうでなければ $s'_i \xrightarrow{a_i} t$ について「$t \leq t'$ and $t' \in U$」となる t' をとり、これを s'_{i+1} とする。$i := i + 1$ とし、ステップ 2 へ。

もし $k = n$ であれば $s = s_n \leq s'_n \in U$ より $\gamma(s) \leq \gamma(s'_n)$ であるから、$\gamma(s) \leq \bigvee\{\gamma(t) \mid t \in U\}$ となる。そうでなければ、$s'_k \in S$ である。条件 (*) を利用し、$s'_{k+1}, \ldots, s'_n$ を「$s_i \leq s'_i$ and $s'_{i-1} \xrightarrow{a_{i-1}} s'_i$ $(i = k+1, \ldots, n)$」を満たすように構成する。すると、

$$\begin{array}{rcll} \gamma(s) & \leq & \gamma(s'_n) & \text{条件 (*)} \\ & \leq & \bigvee\{\gamma(s') \mid s' \text{は } s'_k \text{から到達可能 }\} & \\ & \leq & g(s'_k) & (g \text{ の条件}) \\ & \leq & \bigvee\{\gamma(t) \mid t \in U\} & (s'_k \in S) \end{array}$$

が成り立つ。 □

5 おわりに

本稿では LPTA のリージョン解析・ゾーン解析アルゴリズムを、我々が既に提案した抽象到達可能性検査アルゴリズムで表現した。また、A*を抽象到達可能性検査アルゴリズム上で定式化することにより、それで表現できる種々のアルゴリズムの A*版を容易に求めることができるようにした。これらの定式化を計算機上の証明検証系で形式化し、検証を行うことで多くのアルゴリズムを包括する抽象アルゴリズムの検証を効率的に行うことが将来の課題である。

A*アルゴリズムを使用するには、ある条件を満たす g という関数を用意しなければならない。例えば最短路問題であれば各点からゴールまでの地図上の最短距離を下界とする方法がある。LPTA の最小コスト問題の場合は、ロケーションを移る遷移のみによってゴールに到達するまでの最小コストを下界とすることができ、これ自体は通常の最短路問題で解くことができる。しかし、この方法が A*を使わずに解く方法と比べて有利になるかどうかは吟味しなければならない。その他の手法としては、物理的にどうやってもこのコストを下回れないという値を与えるということも考えられる。LPTA の問題の例として挙げた飛行機の着陸問題では、最も燃費のいい状態を最も短い時間続けられたときのコストをオートマトンとは別に与えることができるであろう。

参考文献

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *TCS*, 126(2):183–236, April 1994.

[2] Gerd Behrmann, *et al.* Minimum-cost reachability for priced timed automata. In *HSCC*, *LNCS* 2034, pages 147–161, 2001.

[3] Kim Larsen, *et al.* As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *CAV*, *LNCS* 2102, pages 493–505, 2001.

[4] 山本 光晴, 高橋 孝一, 萩谷 昌己, 西崎 真也, 玉井 哲雄. グラフ探索アルゴリズムの発展とその検証. **コンピュータソフトウェア**, 18(0):92–108, 2001.