UPPAAL による交差点における自動運転車群のモデル化と検証

○佐原優衣 中村正樹 榊原一紀 （富山県立大学） 玉置久 （神戸大学）

概要 本研究では自動運転車群による交差点通過の制御アルゴリズムを，UPPAAL ツールにより時間オートマトンとして形式的にモデル化し，モデル検査技術を用いて，追従も含めたアルゴリズムで，最小時間を検証する手法を検討する．

キーワード: 自動運転車制御，形式検証，時間オートマトン，UPPAAL

1 はじめに

近年，自動運転技術が急速に発達している．自動運転は，搭載される技術によってレベル 1 からレベル 5 までに分けられており，現在，日本国内では，運転者支援を主としたレベル 2 までが市販車に採用されている. 今後，高速道路や，限定地域での特定条件下での完全自動運転を行うレベル 4 の車両の普及が目指されている．自動運転技術が普及し，大量の自動運転車が利用される都市空間を考える．道路上の車両密度が高くなるため，渋滞やデッドロックが発生することが想定される. したがって，個々の車両だけではなく，自動運転車群が効率的に走行するアルゴリズムが必要となる.

本研究では群制御アルゴリズムが安全性に関わる衝突回避やデッドロック回避，効率性に関わる時間制約などの性質を満たすかどうかを検証する手法を提案する．自動運転車の群制御アルゴリズムを形式的に記述し，モデル検査を用いて，性質を検証する. モデル検査は，システム上で起こり得る状態を網羅的に調べることにより設計の誤りを発見する自動検証手法の一種である. モデル検査手法は，システムの振る舞いの設計，および検証したい性質をそれぞれモデル化し，ツールを用いて，システムが性質を満たしているかを調べる. 本研究では，時間オートマトン [3] による時間制約検証が行えるモデル検査ツール UPPAAL[1, 2] を採用する. UPPAAL を用いて交差点を通過する 1 台の自動運転車の挙動をモデル化する。交差点は 2 車線対面通行で右折用レーンはなく，信号もない交差点とする.

2 追従のない交差点モデル

本節では先行研究 [4, 5] で扱った同一方向の追従のない交差点を用いて，UPPAAL による交差点のモデル化，シミュレーション，検証の方法を紹介する車両は自分の始点と終点を保持し，挙動を交差点進入前，交差点通過中，交差点通過後の 3 つの状態に分類する．交差点には 5 つの鍵があるとして，その組み合わせで交差点の使用権を管理するモデルを考えた．交差点通過には，同時に通行可能な組み合わせと，そうでない組み合わせがある．例えば，対面の直進同士は同時通行可能だが，対面の直進と右折は同時通行不可である．交差点内に 5 つの地点を設定し，各進入方向からの直進右左折が，どの地点を通過するかを分析することで，交差点の使用権を考える．交差点内の 5 つ地点にそれぞれ鍵を設定し，その組み合わせで交差点を通過可能とする.

2.1 時間オートマトン

進行方向と直進右左折を固定した車両の時間オートマトンを作成する．この時間オートマトンは，パラメータ変数として L1, L2, use を持ち，各車両が管理する時間変数 local_clock を持つ．L1, L2 は，インスタンス生成時に，lock1, lock2, lock3, lock4 のいずれで具象化される．use により，地点 cross を通過するかどうかを 0, 1 で表す．図 1 に，時間オートマトンを示す.

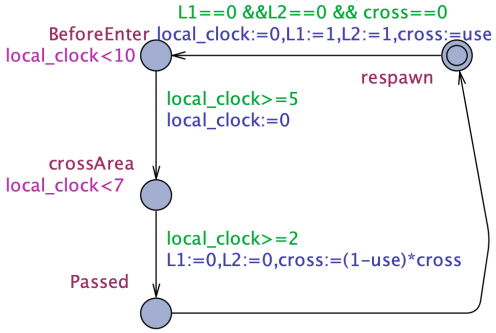

Fig. 1: 交差点を通過する車両の時間オートマトン

パラメータ L1，L2，右折用鍵 cross が取得可能（値が 0）なとき，初期状態 respawn から交差点進入前状態 BeforeEnter への遷移可能とする．遷移時，L1, L2 を他の車両から取得できないようにする（1 に更新）. cross を use で更新する．交差点を使用する車両のときのみ，cross が使用中となる．また，時間変数 local_clock を 0 に更新する．BeforeEnter 状態には，local_clock が 10 秒未満という条件が付与されている．BeforeEnter からの遷移には，5 秒以上という条件が付与されている．直前に時間変数がリセットされているため，これらの条件から，BeforeEnter 状態には 5 秒以上 10 秒未満の期間内のみいられることが記述されている．同様に，crossArea 状態には 2 秒以上 7 秒未満の時間制約が与えられる．crossArea 状態から Passed 状態への遷移時に，L1, L2 の使用権が解放される．また，cross の更新では，地点 cross の仕様の有無に関わらず cross が 0 に更新される.

2.2 シミュレーション

直進は対応する鍵を L1, L2 に設定してる．左折は，L1, L2 に同じ鍵を設定している．右折は，L1, L2 に対応する鍵を与え，use に 1 を与えている．図 2 は，12 のインスタンスのうち 3 台（se, es, wn）が交差点を同時に通過している状態のスクリーンショットである.

2.3 モデル検査

車両全てが交差点を 1 回通過するのにかかる最小時間について検証を行う．交差点の使用権の取得方法は前節と同仕様の 5 つ鍵によって管理する．1 回だけなので循環する時間オートマトンではなく一度通過するだけの時間オートマトンを作成する (図 3)．ループになっ

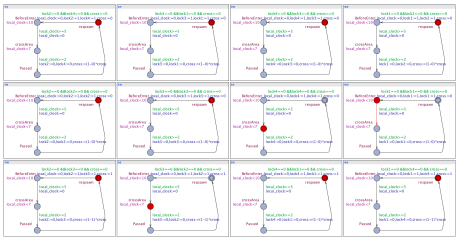

Fig. 2: 交差点を通過する車両の時間オートマトンの合成

ていない以外は，図 1 と同じである．最小時間の検証を行う．検証のために大域時間変数 gc を宣言する．まず，42 秒で全車両が通過できるかどうかを次の検証式を用いて検査する．

```
E<> (gc==42 and ns.finish and sn.finish and
... and ws.finish)
```

この検証式は，経過時間が 42 秒のときに，ns から ws までのすべての時間オートマトンが状態 finish となる実行列が存在することを表している．UPPAAL モデル検査により，上記の検証式が満たされることが確かめられた．次に，以下の検証式を考える．

```
A[] (gc<42 imply not (ns.finish and sn.finish
and ... and ws.finish))
```

この検証式は，すべての実行列で，経過時間が 42 秒未満ならばすべての時間オートマトンが状態 finish にならないことを表している．UPPAAL モデル検査により，上記の検証式が満たされることが確かめられた．UPPAAL モデル検査を用いることで，検証時に具体的

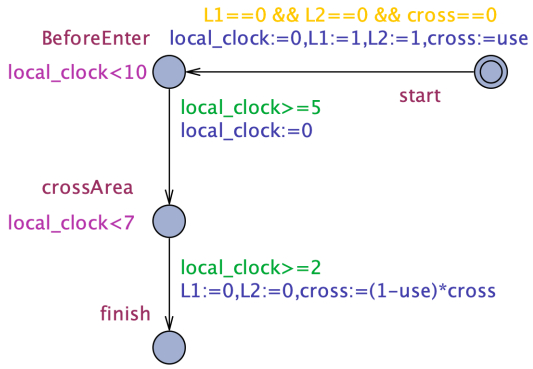

Fig. 3: 交差点を 1 回通過する時間オートマトン

な実行列を得ることができる．42 秒で通過可能な検証式のモデル検査で，図 4 の実行列を得た．この順番で各車両が交差点を通過することで，42 秒で全車両の通過が可能であることが確かめられたことになる．

3　追従を可能した交差点モデル

前章のモデルでは，同方向車両が存在した場合，先行した車両が通行終了するまで，後方車両は交差点を通行できなかった．本章では，例えば同一方向の車両であれば，前方車が時間内に交差点を通過する仮定のもとで，追従して交差点に進入可能なモデルを作成する．制御器は，交差点通過を 2 段階に分けて管理し，通過する各車両がどの段階にあるかを把握しているとする．Fig.5 の外側の円が交差点通過の第一段階，内側の

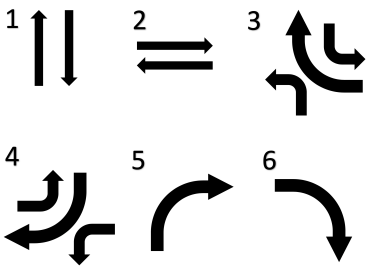

Fig. 4: 最小時間のルート

円が第二段階に対応する．ある車両が北から直進して第二段階で通過中を考える．南から直進または左折する車両は通行可能だが，南から右折する車両は通行できない．一方で西から左折する車両も通行可能である．このとき，北から直進する車両の後続車両が直進や左折の場合，追従可能であるが，右折の場合，追従不可である．

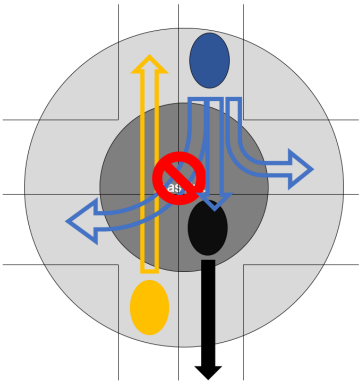

Fig. 5: 交差点の車両の位置範囲

3.1　時間オートマトン

交差点に対して進入する方向と，直進右左折を保持した車両の時間オートマトンを作成する (Fig.6)．この時間オートマトンは，パラメータ変数として，交差点に対して進入する方向 start，直進右左折を示す turn を持つ．各車両が管理する時間変数 time を持つ．グローバルな二次元配列 dir[start][turn] によって，start から turn 方向へ通過する車両の総数を管理する．述語 go() は，各方向，各段階の現在の車両数から，新たな車両が第一段階に移行かどうかを判定する述語として定義される．

3.2　シミュレーション

Fig.7 は 10 インスタンスのうち 2 台 (we,sw) が交差点を通過前状態のスクリーンショットである．検証には大域時間変数 gc を宣言する．

3.3　モデル検査

全ての車両インスタンスが交差点を一回通過するのにかかる最小時間について検証を行う．車両一台の initial から final までプロセスのかかる最小単位時間は 5 である．今回はシミュレーション時の 10 インスタンスで検証を行うので，例えば直進同士でまとめてを 1 セットとすると，2 セットと右折左折のセットが 2 セットで合計 20 単位時間で通行可能ではないかと考える．試しに 20 単位時間で 10 車両が通過できるかを次の検証式を用いて検査する．

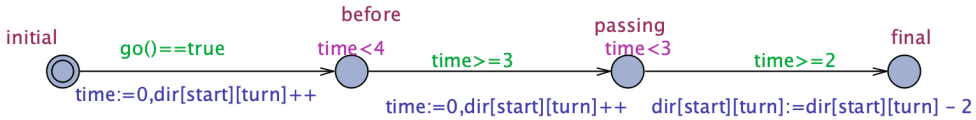

Fig. 6: 交差点を通過する車両の時間オートマトン

Fig. 7: 車両の時間オートマトンの合成

```
E<> (gc==20 and ns.final and sn.final and
... and en2.final)
```

この検証式は経過時間が 20 単位時間の時に ns から en2 までのすべての時間オートマトンが状態 final となる実行例が存在することを示している．UPPAAL モデル検査より，上記の検証式が満たされることが確かめられた．次に以下の検証式を考える．

```
A[] (gc<20 and ns.final and sn.final and
... and en2.final)
```

この検証式は，経過時間がすべての実行例において経過時間が 20 単位時間未満ならばすべての時間オートマトンが状態 final にならないことを示している．UPPAAL モデル検査より，上記の検証式が満たされないことが確かめられた．すなわち，より短い時間で全車両が通過する実行列が存在する．経過時間 20 単位時間よりも小さい値で同様の検証を行った結果，18 単位時間が最小であることが求められた．以下の 2 つの検証式が成り立つ．

```
E<> (gc==18 and ns.final and sn.final and
... and en2.final)
A[] (gc<18 and ns.final and sn.final and
... and en2.final)
```

4　おわりに

本研究では，UPPAAL を用いた自動運転車群制御アルゴリズムのモデル化と検証の手法を提案した．従来のものと比べ，追従を含めた本モデルでは，通過時間を短くなる通過の組み見合わせを作成可能であることを検証した．

謝辞

本研究は JSPS 科研費 JP19K11842 の助成を受けたものです

参考文献

1) Kim Guldstrand Larsen and Paul Pettersson and Wang Yi, UPPAAL in a Nutshell, International Journal of Software Tools for Technology Transfer, Vol.1, No.1-2, pp.134-152, 1997.

2) UPPAAL, `http://www.uppaal.org`

3) Johan BengtssonWang Yi, Timed Automata: Semantics, Algorithms and Tools, Lectures on Concurrency and Petri Nets: Advances in Petri Nets, number 3098 in LNCS, pp.87-124, 2004.

4) 佐原優衣, 中村正樹, 榊原一紀, 玉置久, UPPAAL を用いた自動運転車の群制御アルゴリズムの性能モデル検証, システム制御情報学会研究発表講演会（SCI'19）講演論文集, vol.63, pp.18-24,2019.

5) M. Nakamura, Y. Sahara, C. Kojima, K. Sakakibara and H. Tamaki, Modeling and Verification of Autonomous Vehicle Group Control Algorithms, Proceedings of the SICE Annual Conference 2019 (SICE 2019), pp.115-118,2019.