経営情報システム学特論１
７．性質の検証（３）

SS専攻 経営情報システム学講座 客員
石川 冬樹
f-ishikawa@nii.ac.jp

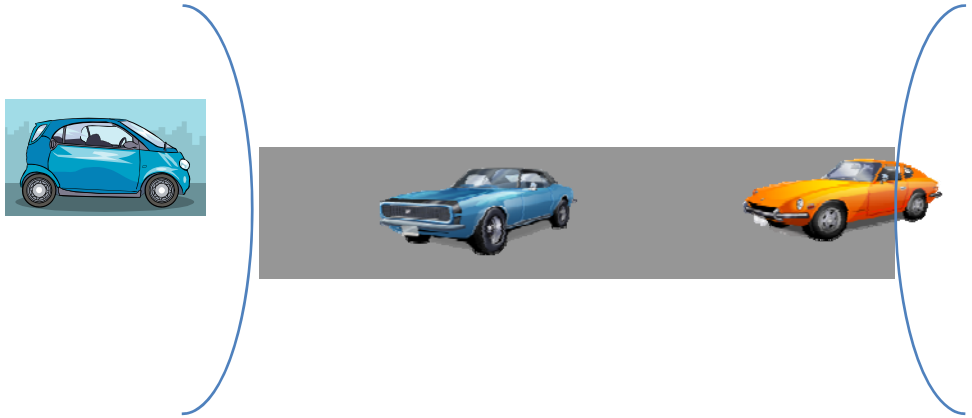
目次

- ここまでの内容の演習

例題

- 一方通行の橋
  - 反対方向に進む車が同時に橋の上に存在してはならない（同じ方向に進む車は複数橋の上に存在してもよい）
  - 橋の上には一定の台数しか乗らない（単にLTSA上の制限と思ってもよい）

➡「橋一式」が車をうまく制御することを考える
（信号などの具体手段は捨象し，どういう条件で止めるかを決めていく）



段階的な試行（一例）

- 下記の手順でLTSAにて進めてみる
  - 橋からの指示に従い行き交う車と，自由な走行を許す橋を表現する
  - 安全性の性質を表現し，「自由に行き交う」場合だと問題があることを検証できるようにする
  - 車が進むことが許されるガード条件を定め，それにより安全性が満たされることを検証し，必要なら修正する
  - ガード条件があっても活性が満たされることを検証し，必要なら修正する

初期モデル

```
const N = 3
range NUM = 0..N
range ID = 1..N

CAR = ( enter -> exit -> CAR ).

BRIDGE = BRIDGE[0][0],
BRIDGE[numtow:NUM][numtoe:NUM] = (
    towest.[id:ID].enter -> BRIDGE[numtow+1][numtoe]
  | towest.[id:ID].exit -> BRIDGE[numtow-1][numtoe]
  | toeast.[id:ID].enter -> BRIDGE[numtow][numtoe+1]
  | toeast.[id:ID].exit -> BRIDGE[numtow][numtoe-1]
).

||CARS = ( [ID]:CAR ).

||SYSTEM = ( towest:CARS || toeast:CARS || BRIDGE ).
```

デバッグ用に
適宜小さくするとよい

橋の上にいる車の台数を数えるときのrangeと
車を指すIDのrange

配布
onewaybridge0.lts

演習：安全性検証

- 下記に関する安全性を定義し，検証し，予想通りの違反が検出されることを確認せよ

  「反対方向に進む車が同時に橋の上に存在してはならない」

  - 方法１：propertyにより，許される遷移をプロセスとして与える
  - 方法２：時相論理式をassertにて与える

    （楽だと思われる方でよい）

演習解答：安全性検証

- 解答例： onewaybridge0-safety.lts

演習：ガード条件の追加

- 車のenter/exitと同期し，ガード条件が成り立つときのみそれらが発火するように制御をする橋のプロセスを合成し，安全性が成り立つか検証せよ

```
BRIDGE = ...
(
    when (...) towest.[id:ID].enter -> ...
  | when (...) towest.[id:ID].exit -> ...
  | when (...) toeast.[id:ID].enter -> ...
  | when (...) toeast.[id:ID].exit -> ...
).
||SYSTEM = ( towest:CARS || toeast:CARS
             || BRIDGE).
```

必要ならばガード条件を付ける

- 次スライドは解答

演習解答：ガード条件の追加

- 解答例
  - enterにだけガード条件を与えればよい
  - 東に向かう車がなければ，西に向かってenterできる（逆も同じ）
  - onewaybridge1.lts

演習：活性検証

- 活性を表す重要な遷移に注目することにより，活性を定義し，LTSA上で検証せよ
  - 方法１：progressにより，到達した無限サイクルに含まれるべきラベルを与える
    （あるいは何も与えずチェックするとすべてのラベルがチェックされる）
  - 方法２：時相論理式をassertにて与える
    （楽だと思われる方でよい）

  次頁の補足も参考にすること

演習：活性検証（続）

- 公平性に関する仮定をどう考えるか？
  - enterばかりが起きる状況（どの車も，自分が橋に入れる状況ならどんどん入ってしまう）は考えるに値しそう
  - 言い換えると，反対方向の車のために自主的に（信号の指示などなく）待ってあげたりはしない
- enterが他ラベルより優先するような設定で検証してみる

演習解答：活性検証

- 解答例： onewaybridge1-liveness.lts

演習：活性の実現

- 活性を保証するように修正せよ（方針は様々）
  - 何も考えず定期的に方向を切り替え？
    （時間での単純な切り替え）
  - 対向方向の待ちの車がいるときは（あるいは溜まったときは）もう橋に入れないように切り替え？
    （センサーを用いた高度な制御）
  - ．．．

※ ただしその結果，また安全性が崩れたり，あるいはデッドロックしたりする可能性があるため，Safety Checkはすること

- 次スライドは解答

演習解答：活性の実現

- 解答例
  - 橋に入ろうとしている車の数もカウントし，対向方向の待ちの車がないときだけ橋に入れる，とする？
    - onewaybridge2-1.lts
  - 車が1台通ったら，あるいは定期的に，優先順位が切り替わるようになる
    - onewaybridge2-2.lts
    - onewaybridge2-3.lts

まとめ

- 性質の検証
  - 「簡単な」設計モデルであっても正しさを保証することは難しい
    （今回の例は簡単だが，コードを抽象化した本質であるという意味では実用的なロジック）
  - 安全性とデッドロックフリーや活性は相反する方向性であるため，片方を満たすための施策がもう片方を崩してしまうことがある
    （繰り返しの検証が必要）

- 次回： 並行・分散システムの様々な実用側面