オートマトンと検証手法

北陸先端科学技術大学院大学 先端科学技術研究科（青木研究室）

長谷川 央

2022-6-18

自己紹介

名前
長谷川 央（ハセガワ アキラ）

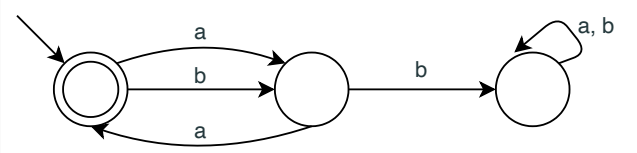
経歴

1997	愛知県豊田市で生まれる
2016	名古屋大学教育学部附属中・高卒
2016–2020	三重大学 総合情報処理センター主催 講習会「パソコン分解講習会」TA
2019	三重大学 総合情報処理センター主催 講習会「Linux 実践入門」講師
2020	北陸先端科学技術大学院大学 入学
2021–	JAIST 情報基盤センター ヘルプデスク

背景

計算機の理論を勉強するときに「オートマトンと形式言語」などの講義で
オートマトンの理論について学ぶ

しかしオートマトンの**実用的な**使用方法については授業では習わない



前回：自作ゲーム

オセロっぽいゲームに対してオートマトンベースのテストを行った

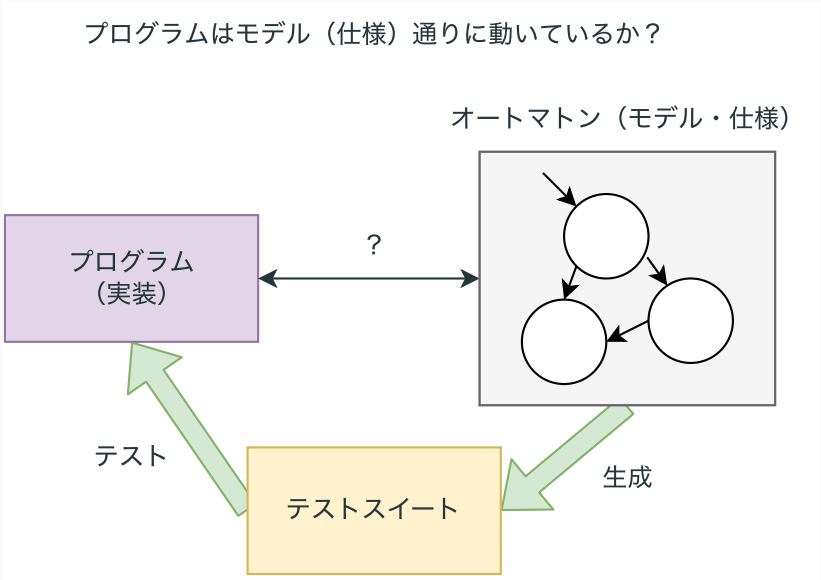
ルール：

1 次元 4 マスのボードでプレイし 相手の石を挟むと取り除ける

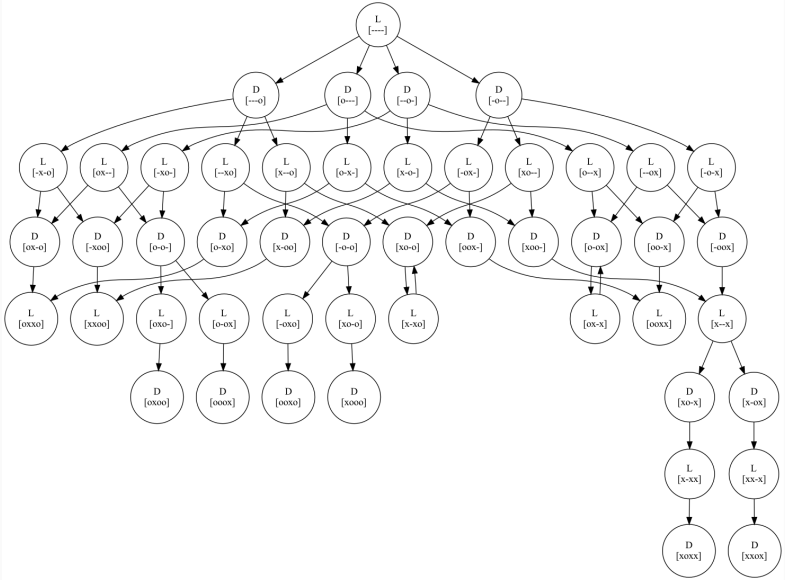
白手番スタートで最終的に石数が多いほうの勝ち

```
→ ./target/release/simple_game
[Light] Choose: [0|1|2|3]
1
[ |○| | ]
[Dark] Choose: [0| |2|3]
0
[●|○| | ]
[Light] Choose: [ | |2|3]
2
[●|○|○| ]
[Dark] Choose: [ | | |3]
3
[●| | |●]
[Light] Choose: [ |1|2| ]
1
[●|○| |●]
[Dark] Choose: [ | |2| ]
2
[●| |●|●]
[Light] Choose: [ |1| | ]
1
[●|○|●|●]
Dark: 3, Light: 1
DARK WIN!
```

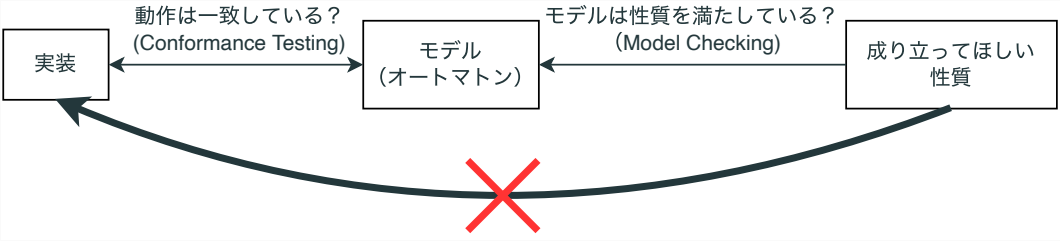
前回：オートマトンを使ったテストの概要図



前回：テストに使用したオートマトン



成り立ってほしい性質と実装の関係



オートマトンを使った形式検証について知る

オートマトンの使い道

前回テストを自動生成するためにオートマトンを作成した

作成したオートマトンは自作ゲームの挙動を模倣している

他に使い道はないか？

→ 手動での解析，形式検証（モデル検査）

手動での解析 (1/2)

数学的なツール等を使わずともモデルから分かる有用なことは多い

例：

- 白・黒の勝利確率
- 最短/平均終了手数（ループなしの場合）

手動での解析 (2/2)

白・黒の勝利確率を同等にしたい

現状：

白 4，黒 2，引き分け 3，ループ 2

改善案：

ループしたら黒の勝ちとする

→ 白 4，黒 4 となる
→ 良いバランス？

モデル検査による検証

モデル検査

- 形式検証（形式手法）の 1 つ
- モデル（オートマトン）自体を検証する手法

例えば…

ゲーム（モデル）は必ず停止するか

しないなら どのような手順のときにループするか

前回のモデル (1/2)

前回使用したモデルの定義

$\mathcal{M} = (S, S_0, \Sigma, \delta)$, where

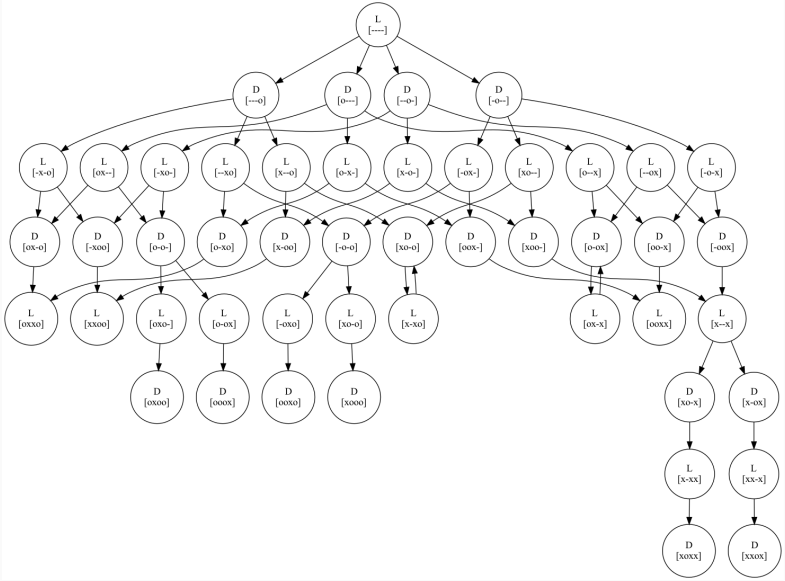
- S is a finite set of states
- $S_0 \in S$ is a initial state
- Σ is the alphabet, which is a finite set of input symbols
- $\delta : S \times \Sigma \to S$ is a transition function

状態は Turn × Board とする
e.g.) 状態 (L,[oxx-]) は現在の手番が Light（白）で盤面が [○|●|●|] であることを示す
また，(D,[- -o-]) は現在の手番が Dark（黒）で盤面が [| |○|] である

初期状態は (L,[- - - -]) とする

アルファベットは左から「n 番目に石を置く」という意味で $\Sigma = \{0, 1, 2, 3\}$ としている
紙面の都合で遷移のアルファベットに関しては省略して記載する

前回のモデル (2/2)


L [----]
D [---o] D [o---] D [--o-] D [-o--]
L [-x-o] L [ox--] L [-xo-] L [--xo] L [x--o] L [o-x-] L [x-o-] L [-ox-] L [xo--] L [o--x] L [--ox] L [-o-x]
D [ox-o] D [-xoo] D [o-o-] D [o-xo] D [x-oo] D [-o-o] D [xo-o] D [oox-] D [xoo-] D [o-ox] D [oo-x] D [-oox]
L [oxxo] L [xxoo] L [oxo-] L [o-ox] L [-oxo] L [xo-o] L [x-xo] L [ox-x] L [ooxx] L [x--x]
D [oxoo] D [ooox] D [ooxo] D [xooo] D [xo-x] D [x-ox]
L [x-xx] L [xx-x]
D [xoxx] D [xxox]

モデル検査の流れ

モデル検査は 3 STEP で行える

1. modeling: モデリング（M の作成）
2. specification: 性質の記述（f の記述）
3. verification: 検証（$M \models f$; モデル検査ツールで自動実行）

モデル検査用のモデル

前回のモデルに AP と L を加えるとモデル検査で使用可能になる

$\mathcal{M} = (S, S_0, \Sigma, \delta, AP, L)$, where

- S is a finite set of states
- $S_0 \in S$ is a initial state
- Σ is the alphabet, which is a finite set of input symbols
- $\delta : S \times \Sigma \to S$ is a transition function
- AP is a set of atomic propositions
- $L : S \to 2^{AP}$ is a function mappping a state to a set of AP that's true on the state

AP の例：

WHITEWIN: 白が勝利，DARKWIN: 黒が勝利，DRAW: 引き分け

（再掲）モデル検査の流れ

モデル検査は 3 STEP で行える

1. modeling: モデリング（M の作成）
2. specification: 性質の記述（f の記述）
3. verification: 検証（$M \models f$; モデル検査ツールで自動実行）

原子命題を使用した性質の記述

時相論理式を使ってモデル上で確かめたい性質を書く

性質の例：

ループしない（停止する）

$\Longleftrightarrow$ 必ずいつか白か黒が勝利するか引き分けとなる

$\Longleftrightarrow \Box\Diamond(WHITEWIN \vee DARKWIN \vee DRAW)$

（線形時相論理式 Linear Temporal Logic による記述）

（再掲）モデル検査の流れ

モデル検査は 3 STEP で行える

1. modeling: モデリング（M の作成）
2. specification: 性質の記述（f の記述）
3. verification: 検証（$M \models f$; モデル検査ツールで自動実行）

検証

$M \models f$

- モデル：前述のモデル
- 性質：ループしない（$\Box\Diamond(WHITEWIN \vee DARKWIN \vee DRAW)$）

今回はモデル上にループするようなパスがないかを調べたい

通常 検証はモデル検査ツールによって自動で（機械的に）行う

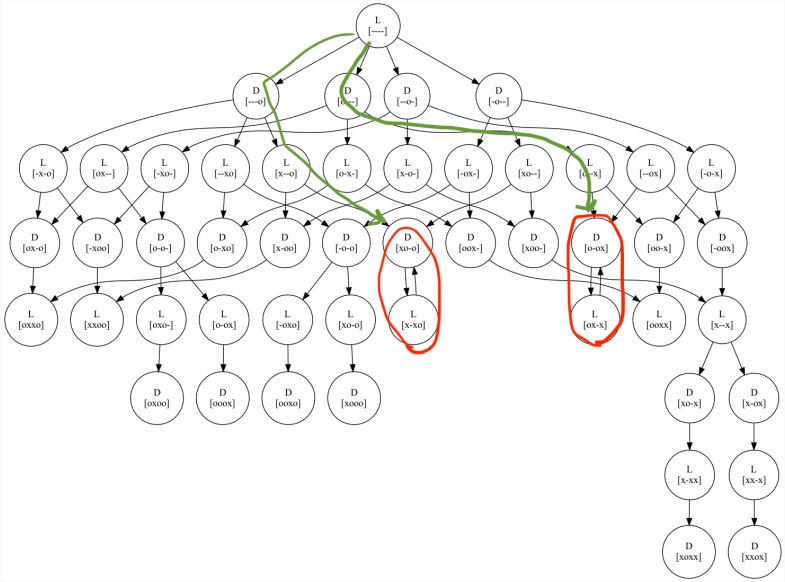
検証の実行過程

モデル検査ツールは記述した性質に違反するパスを探索する

モデルを M，性質を φ とするとき

1. $\neg\varphi$ を受理するモデル A を作成
2. $L(A) \cap L(M) = \emptyset$?
   - True: モデルは性質を満たす
   - False: $L(A) \cap L(M)$ が反例として得られる

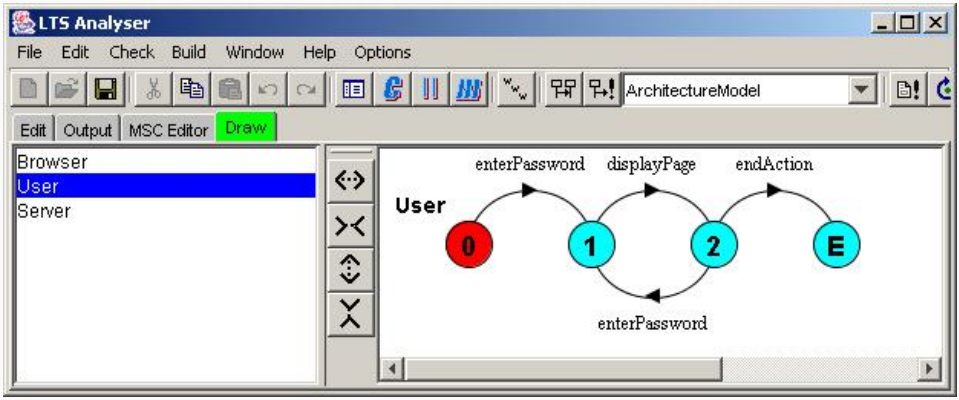
得られる反例


L [----]
D [---o]
D [--o-]
D [-o--]
D [o---]
L [-x-o]
L [ox--]
L [-xo-]
L [--xo]
L [x--o]
L [o-x-]
L [x-o-]
L [-ox-]
L [xo--]
L [o--x]
L [--ox]
L [-o-x]
D [ox-o]
D [-xoo]
D [o-o-]
D [o-xo]
D [x-oo]
D [-o-o]
D [xo-o]
D [oox-]
D [xoo-]
D [o-ox]
D [oo-x]
D [-oox]
L [oxxo]
L [xxoo]
L [oxo-]
L [o-ox]
L [-oxo]
L [xo-o]
L [x-xo]
L [ox-x]
L [ooxx]
L [x--x]
D [oxoo]
D [ooox]
D [ooxo]
D [xooo]
D [xo-x]
D [x-ox]
L [x-xx]
L [xx-x]
D [xoxx]
D [xxox]

ツールについて (1/2)

様々なモデルや性質に対するモデル検査ツールが既に提供されている

今回の場合は LTSA (Labelled Transition System Analyser)，SPIN などが使えそう



ツールについて (2/2)

他にも

- Alloy
- TLA+
- NuSMV
- UPPAAL
- PRISM
- ...

など色々とある

それぞれ対象の性質によって使い分ける
(時間を扱うなら UPPAAL，確率を扱うなら PRISM など)

まとめ

- テスト用に作成したオートマトンは解析や検証にも活用できる
- モデル自体を検証に使いたい場合はモデル検査を使う
- モデル検査ツールは反例探索（検証）を自動で行ってくれるツールで色々と種類がある