オートマトンの自動学習手法

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

2022-7-16

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

プログラムのテスト・検証を行う際に事前にオートマトンを作ると

1. 動作が可視化されるので動作について考えやすくなる
2. モデルを使って自動でテストケースが生成できる
3. モデル検査で成り立ってほしい性質が成り立つかを数学的に検証できるようになる

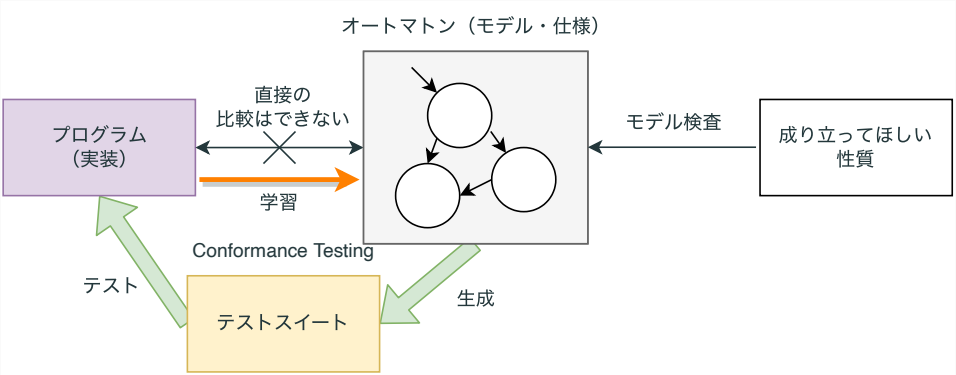
→ しかし，仕様書がない/仕様が分からない場合 オートマトンは作れない

Active Automata Learning (AAL)

プログラムの動作ログからオートマトンを学習する手法を Automata Learning と言う

その中でも動的にログを取得しながら学習する手法を Active Automata Learning (AAL) と言う

AAL にはいくつか種類あるが Angluin's L* algorithm ベースの手法が最も広く使われている



Deterministic Finite Automata (DFA)

$M = (Q, \Sigma, \delta, s, F)$

- Q: 状態の有限集合
- Σ: アルファベット
- $\delta : Q \times \Sigma \to Q$: 遷移関数
- $s \in Q$: 初期状態
- $F \subseteq Q$: 受理状態の集合

例 $M = (\{0, 1, 2, 3\}, \{a, b\}, \delta, 0, \{3\})$ [1]

- $\delta(0, a) = 1$
- $\delta(1, a) = 2$
- $\delta(2, a) = \delta(3, a) = 3$
- $\delta(q, b) = q \in \{0, 1, 2, 3\}$

[1] (D. Kozen, Automata and Computability. Springer-Verlag New York, 1997)

正規文法・正規言語

文法 $G = (V_N, V_T, P, S)$ [2]

- V_N: 変数（非終端記号）の集合
- V_T: 終端記号の集合
- P: 生成規則の集合
- S: 開始記号

P の任意の生成規則が以下のいずれかであるとき G を（右）正規文法と呼び
正規文法で生成される言語を正規言語と言う

- $A \to \varepsilon$
- $A \to a$
- $A \to aB$

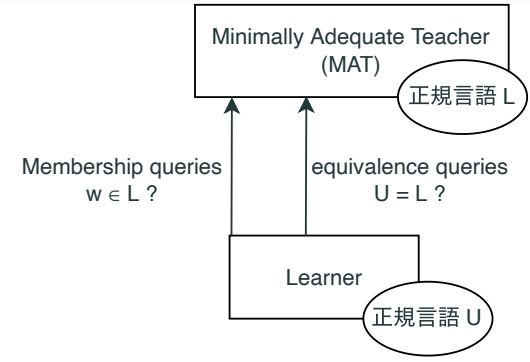
DFA が受理する言語は必ず正規言語であり全ての正規言語はそれを受理する DFA が構築可能

[2] (J.E. Hopcroft, J.D. Ullman, 野崎昭弘, 木村泉, 言語理論とオートマトン. サイエンス社, 1971)

Angluin's L* algorithm

L* algorithm は既知のアルファベット上の未知の正規言語を学習するアルゴリズム [3]

図中 2 つのクエリに正確に答える Minimally Adequate Teacher (MAT) が存在するとき Angluin's L* Algorithm を用いて L を学習することができる



[3] (D. Angluin, "Learning regular sets from queries and counterexamples," Information and Computation, 1987)

Angluin L* algorithm の大まかな流れ

アルゴリズムの大まかな流れ (学習対象の言語: L，学習中の DFA が受理する言語: U)

1. membership queries ($w \in L$?) で学習対象の正規言語 L に含まれる語を探し表にメモする
2. 表を基に DFA を構築し DFA が受理する言語 U と学習対象の言語が等しいかを equivalence queries ($L = U$?) で確かめる
3. 反例が見つかったらそれを表に書き加えて 1 に戻る

学習に使用する表

以下で定義される表 (Observation Tables) を拡張することで学習を行う

Observation Table (S, E, T)

- S: a nonempty finite prefix-closed set of strings
- E: a nonempty finite suffix-closed set of strings
- T: $((S \cup S \cdot \Sigma) \cdot E) \to \{0, 1\}$

関数 row は表の行を返す

表の例

T_4	λ	0
λ	1	0
0	0	1
1	0	0
11	1	0
00	1	0
01	0	0
10	0	0
110	0	1
111	0	0

FIG. 6. $S = \{\lambda, 0, 1, 11\}$, $E = \{\lambda, 0\}$.

表を用いた DFA の構築

表と関数 row を用いて次のように DFA を構築することができる

- $Q = \{\mathrm{row}(s) \mid s \in S\}$
- $q_0 = \mathrm{row}(\varepsilon)$
- $F = \{\mathrm{row}(s) \mid s \in S \text{ and } T(s) = 1\}$
- $\delta(\mathrm{row}(s), a) = \mathrm{row}(s \cdot a)$

正規言語が満たすべき性質

学習後の正規言語は以下の性質を満たしていなければならない

- closed (遷移先の状態がテーブルに存在している)
  $\forall t \in S \cdot \Sigma \ \exists s \in S \, (\mathrm{row}(t) = \mathrm{row}(s))$
- consistent (row が同じなら同じ状態ということなので必ず同じ遷移を行う)
  $\forall s_1, s_2 \in S \ \ (\mathrm{row}(s_1) = \mathrm{row}(s_2) \implies (\forall a \in \Sigma \ \ (\mathrm{row}(s_1 \cdot a) = \mathrm{row}(s_2 \cdot a))))$

[再掲] Angluin L* algorithm の大まかな流れ

アルゴリズムの大まかな流れ (学習対象の言語: L，学習中の DFA が受理する言語: U)

1. membership queries (w ∈ L?) で学習対象の正規言語 L に含まれる語を探し表にメモする
2. 表を基に DFA を構築し DFA が受理する言語 U と学習対象の言語が等しいかを equivalence queries (L = U?) で確かめる
3. 反例が見つかったらそれを表に書き加えて 1 に戻る

追記：1 は closed，consistent を満たすまで継続する

Angluin's L* Algorithm の例: a^*b^+ $(\Sigma = \{a, b\})$

	ε
ε	0
a	0
b	1

	ε
ε	0
b	1
a	0
ba	0
bb	1

c.e.: bab

	ε
ε	0
b	1
ba	0
bab	0
a	0
bb	1
baa	0
$baba$	0
$babb$	0

$\mathrm{row}(\varepsilon) = \mathrm{row}(ba)$

$\mathrm{row}(b) \neq \mathrm{row}(bab)$

	ε	b
ε	0	1
b	1	1
ba	0	0
bab	0	0
a	0	1
bb	1	1
baa	0	0
$baba$	0	0
$babb$	0	0

L*の実用について

equivalence queries は学習中の正規言語と学習対象の正規言語が等しいかを確認するクエリ
→ 実現するためには学習対象の正規言語が分からないといけない
→ したがってそのままでの実用はできない

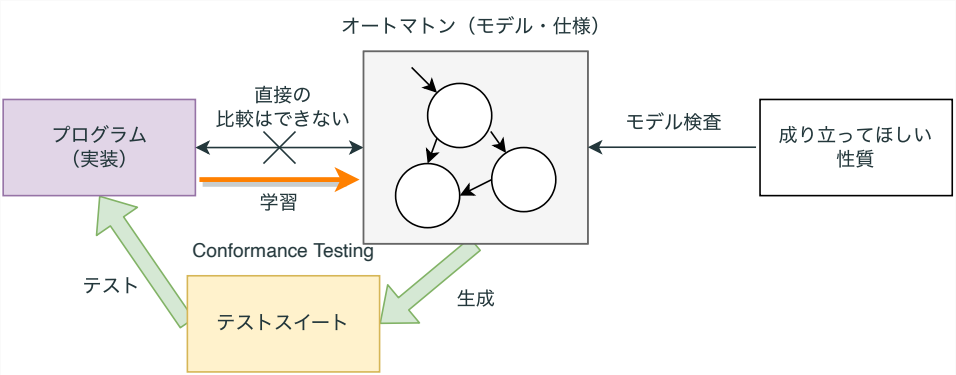
実用時には conformance testing を使って代用している

[再掲] Active Automata Learning (AAL)

プログラムの動作ログからオートマトンを学習する手法を Automata Learning と言う

その中でも動的にログを取得しながら学習する手法を Active Automata Learning (AAL) と言う

AAL にはいくつか種類あるが Angluin's L* algorithm ベースの手法が最も広く使われている



TCP の学習の例

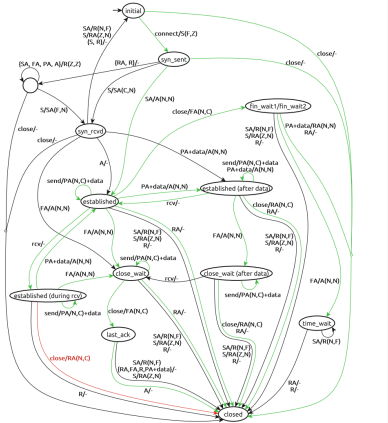


Fig. 2. Learned model for Windows 8 TCP Client. To reduce the size of the diagram, we eliminate all self loops with *timeout* outputs. We replace flags and abstractions by their capitalized initial letter, hence use *s* for *syn*, *a* for *ack*, *n* for *next*, etc. We omit input parameter abstractions, since they are the same for all packets, namely *valid* for both sequence and acknowledgement numbers. Finally, we group inputs that trigger a transition to the same state with the same output. Timeouts are denoted by '-'.

[4]

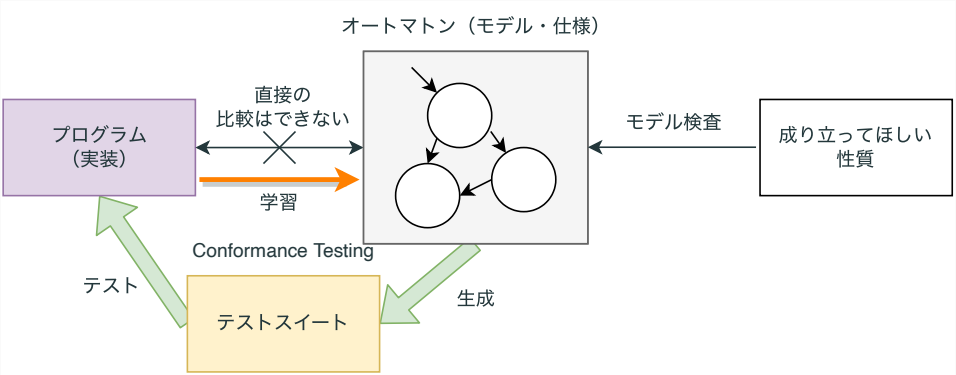
[4] (P. Fiterău-Broştean, R. Janssen, and F. Vaandrager, "Combining model learning and model checking to analyze tcp implementations," in CAV, 2016)

[再掲] Active Automata Learning (AAL)

プログラムの動作ログからオートマトンを学習する手法を Automata Learning と言う

その中でも動的にログを取得しながら学習する手法を Active Automata Learning (AAL) と言う

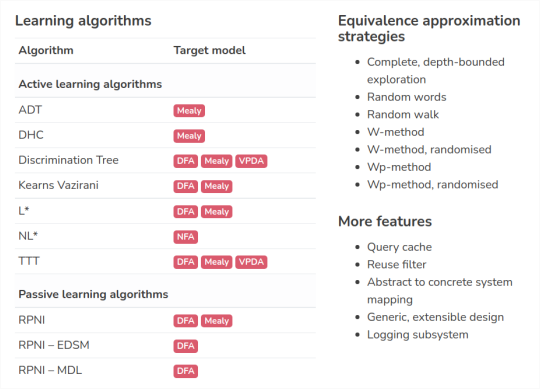
AAL にはいくつか種類あるが Angluin's L* algorithm ベースの手法が最も広く使われている



便利なライブラリ: LearnLib

Java のライブラリ **LearnLib** [5] を使えば簡単に AAL が実行できる

Angluin's L* Algorithm を改良した様々なアルゴリズムが実装されている



Learning algorithms

Algorithm	Target model
Active learning algorithms	
ADT	Mealy
DHC	Mealy
Discrimination Tree	DFA Mealy VPDA
Kearns Vazirani	DFA Mealy
L*	DFA Mealy
NL*	NFA
TTT	DFA Mealy VPDA
Passive learning algorithms	
RPNI	DFA Mealy
RPNI – EDSM	DFA
RPNI – MDL	DFA

Equivalence approximation strategies

- Complete, depth-bounded exploration
- Random words
- Random walk
- W-method
- W-method, randomised
- Wp-method
- Wp-method, randomised

More features

- Query cache
- Reuse filter
- Abstract to concrete system mapping
- Generic, extensible design
- Logging subsystem

[5] (Learnlib, https://learnlib.de/)

まとめ

- L*を用いることで自動でプログラムからオートマトンを得ることができる
- L*により仕様が不明な実装に対してもモデル検査を行うことができる