SYRCoSE
Spring/Summer Young Researchers` Colloquium on Software Engineering

Bisimulations in Memory Finite Automata

Aleksandr Delman
`adelman2112@gmail.com`

Antonina Nepeivoda
`a_nevod@mail.ru`

Anna Terentyeva
`mathhyyn@gmail.com`

ISP RAS
St Petersburg University
NCFU NORTH-CAUCASUS FEDERAL UNIVERSITY

Popular Regex Extensions

Regular

Lookaheads or Lookbehinds

Negative Lookaheads

(not in PCRE, but...)

Async. Composition

Non-Regular

Capture Groups &

Backreferences

Problems become EXP-harder
(but are still viable
in many practical cases)

With lookaheads and negations,
haven’t formalised yet

Backreferences Formalism

- Schmid model: only named capture groups.

Backref-regex (ref-words, by Schmid) operations:

$$\begin{cases} [_k \tau]_k & \text{(named capturing)} \\ \&\mathtt{k} & \text{(reading memory cell)} \end{cases}$$

Example: $[_1 \mathtt{a}^+]_1 \mathtt{a}^+ \mathtt{b}\,\&\mathtt{1}$ defines $\{a^m b a^n \mid m > n \,\&\, n > 0\}$

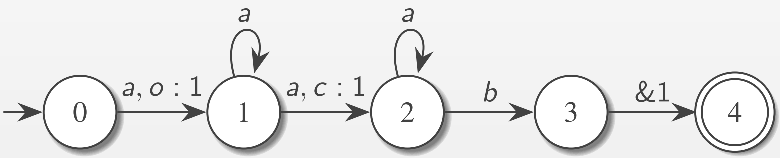
- ε-semantics (Schmid) — uninitialized reference recognizes $\{\varepsilon\}$;
- $\varnothing$-semantics (regex engines) — uninitialized reference recognizes $\varnothing$.

No impact on language properties.

Backreferences Formalism

- Possibly unbalanced and nested (not self-nested) capturing.
- References to k^{th} memory cell — outside k^{th}-capture groups.
- Consequence: extended NFA construction.

Memory Finite Automaton



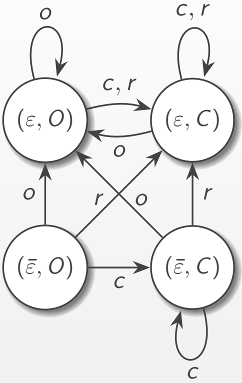
$$\text{Memory operations: } \begin{cases} o & \text{(open)} \\ r & \text{(reset)} \\ c & \text{(close)} \end{cases}$$

Parsing by MFA

MFA Configuration

- a current state;
- accumulated memory values;
- memory cells configurations:

$$\begin{cases} O & \text{(open)} \\ C & \text{(closed)} \end{cases}$$



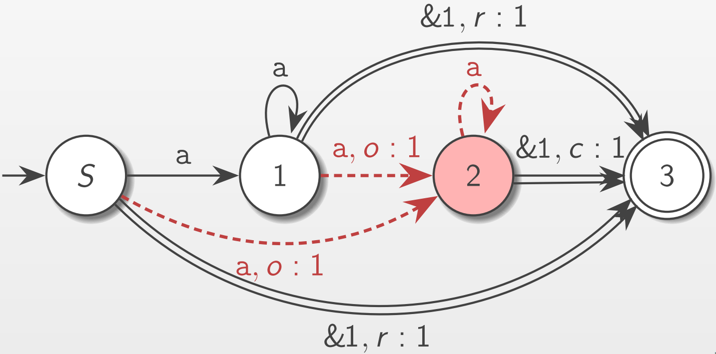
- The initial memory state: $(q_0, w, (\varepsilon, C), \ldots, (\varepsilon, C))$.
- Memory action is performed before tape symbol processing.

MFA and Ref-Words

In any ref-word $\varphi_1 \underbrace{[_k \varphi_2]_k}_{\text{capture group}} \varphi_3$,
φ_2 is again a ref-word

In an MFA, arc sets starting at $o : k$ and ending by $c : k$ instructions should form subautomata

Well-formed capture groups in MFA:



MFA and Ref-Words

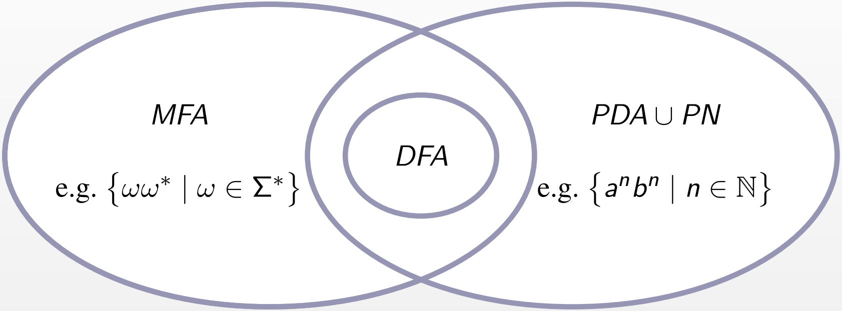
In any ref-word $\varphi_1 \underbrace{[_k \varphi_2]_k}_{\text{capture group}} \varphi_3$, φ_2 is again a ref-word

In an MFA, arc sets starting at $o : k$ and ending by $c : k$ instructions should form subautomata

Any capture group should be useful at least along one path.

$$\Big(\underbrace{[_1 a^*]_1}_{\text{useful}}([_1 \mathtt{a}^*]_1)^* \mid \underbrace{[_1 a^*]_1}_{\text{useless}}[_1 \mathtt{a}^*]_1\Big)(\&1 \mid \mathtt{b})$$

MFA Languages



- Existing models, such as PDA and Petri nets, are not suitable for MFA languages recognition.

Hardness of MFA-related Problems

Jiang T., *et al*
Inclusion is undecidable
for pattern languages (1993).

Inclusion & equivalence
are undecidable
for one-cell ref-words.

Freydenberger D. D.
Inclusion of pattern languages
and related problems (2011).

Seek for candidate over-approximations
of the language inclusion that are simpler to resolve.

Bisimilarity for LTS

Every state machine $\mathscr{A}$ can be represented as a labelled transition system.

- $\mathscr{A}_1$ and $\mathscr{A}_2$ are bisimilar $\Leftrightarrow$ their LTS $\mathcal{T}_1$ and $\mathcal{T}_2$ are bisimilar.

Bisimulation is a relation $\sim$ between states of the systems $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfying the following property:

- If $q_1 \sim q_2$ ($q_1 \in \mathcal{T}_1$, $q_2 \in \mathcal{T}_2$), then for every transition $q_1 \xrightarrow{\gamma} q_1'$ in $\mathcal{T}_1$ there is a transition $q_2 \xrightarrow{\gamma} q_2'$ in $\mathcal{T}_2$ such that $q_1' \sim q_2'$, and vice versa.

Starting and final (if any) states must be bisimilar.

Bisimulation Game

$\mathcal{T}_1$ and $\mathcal{T}_2$ bisimilarity checking technique can be formulated as a two-player game with an initial configuration $\langle q_S, q'_S \rangle$:

Next configuration $\langle q_n, q'_n \rangle$

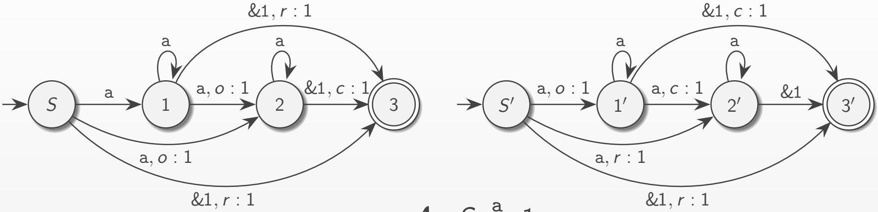
$\mathcal{A}$ chooses any element of the current pair and a transition $q_c \xrightarrow{\gamma} q_n$.

$\mathcal{D}$ responds with a transition $q'_c \xrightarrow{\gamma} q'_n$ from the remaining state respecting q_n finality.

$\mathcal{D}$ cannot choose any matching transition $\Rightarrow$ LTS are not bisimilar.

- Attacker's winning strategy always leads to the fact that any possible play is finite.

Bisimulation Game: an Example



$\mathcal{A} : S \xrightarrow{\mathtt{a}} 1$

$\mathcal{D} : S' \xrightarrow{\mathtt{a}} 1'$

$\mathcal{A} : 1 \xrightarrow{\&1} 3$

$\mathcal{D}$ loses (no way to reset &1)

$\mathcal{D} : S' \xrightarrow{\mathtt{a}} 2'$

$\mathcal{A} : 1 \xrightarrow{\mathtt{a}} 2$

$\mathcal{D}$ loses (no way to change &1)

External Actions in MFA

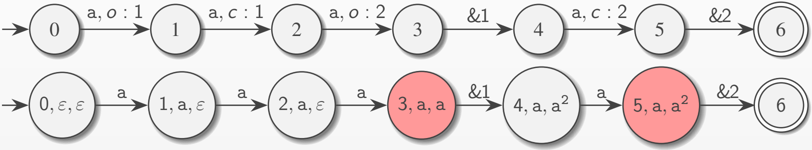
- References are external, i.e. accessible to players.

The action alphabet $\mathcal{A}$ in k-MFA is $\Sigma \cup \{\texttt{\&i}\}_{1\leq i\leq k}$. Given an action $\texttt{\&i}$ referencing to ω in a run of $\mathscr{A}_1$, an equal response action in a run of $\mathscr{A}_2$ is action $\texttt{\&i}$ reading the same memoised string ω.

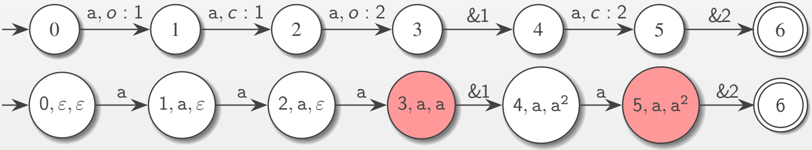
- Capturing is internal, i.e. players control the capture process by state change.

Non-Trivial Bisimulations

- MFA and its LTS for $\mathtt{a}[_1\mathtt{a}]_1\mathtt{a}[_2\&1\mathtt{a}]_2\&2$.



- MFA and its LTS for $[_1\mathtt{a}]_1\mathtt{a}[_2\mathtt{a}\&1]_2\mathtt{a}\&2$.



- The capture groups $[_2\&1\mathtt{a}]_2$ and $[_2\mathtt{a}\&1]_2$ contain distinct action languages.

Word Equations Problem Reduction

- Construct a ref-word ρ using k memory cells;
- Construct the following ref-words:
  - $\rho_1 = \rho\, [_{\mathtt{k+1}} \mathcal{U}(\&1, \ldots, \&\mathtt{k})]_{\mathtt{k+1}} \mathcal{V}(\&1, \ldots, \&\mathtt{k}) \&\mathtt{k} + 1$
  - $\rho_2 = \rho\, \mathcal{U}(\&1, \ldots, \&\mathtt{k}) [_{\mathtt{k+1}} \mathcal{V}(\&1, \ldots, \&\mathtt{k})]_{\mathtt{k+1}} \&\mathtt{k} + 1$
- MFA for ρ_1 and ρ_2 are bisimilar $\Leftrightarrow$ all memory values generated by ρ satisfy equation

$$\mathcal{U}(\&1, \ldots, \&\mathtt{k}) = \mathcal{V}(\&1, \ldots, \&\mathtt{k})$$

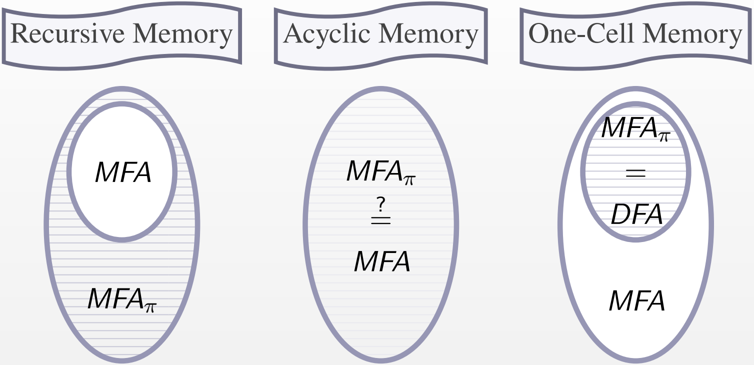
Non-parametrisable solutions to equation

$$\&1\ \mathtt{ab}\ \&2 = \&2\ \mathtt{ba}\ \&1$$

can be generated, *e.g.* by the ref-word

$$[_1 \mathtt{b}^*]_1 \left(\left[_2 (\&1\ \mathtt{ba})^*\ \&1\ \mathtt{b} \right]_2 \left[_1 (\&2\ \mathtt{ab})^*\ \&2\ \mathtt{a} \right]_1 \right)^*$$

Memory Cell Languages $\mathscr{L}(MFA_\pi)$



$\{a^n b^n\} \in \mathscr{L}(MFA_\pi)$ in the recursive case.

Track values of &3 in $\left([_2\mathtt{a}\ \&1\ \mathtt{b}]_2[_1\mathtt{a}\ \&2\ \mathtt{b}]_1\right)^*\left[_3(\&1 \mid \&2)\right]_3$.

One-Cell MFA Reduction

$\{\omega\omega\} \notin \mathscr{L}(PDA)$, however $\{\omega\omega^R\} \in \mathscr{L}(PDA)$, and $\omega_1 = \omega_2 \Leftrightarrow \omega_1^R = \omega_2^R$.

1-MFA to PDA Reduction Idea

- memoize capture groups using special stack symbols,
- and pop the stack symbols when reading tape symbols from a twin "memoized tape alphabet".

Problems

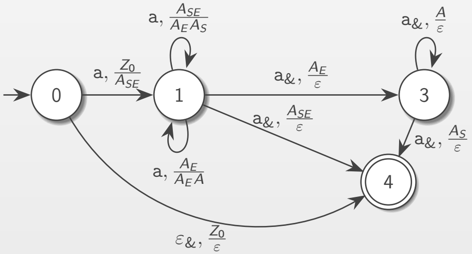
- Implicit (ε) memory bounding markers;
- Mixing reading from tape and from memory;
- Processing resets & multiple references.

Unique Memory Pointer

- Stack alphabet Γ: for any $a \in \Sigma$, add $A_S, A_{SE}, A_E, A \in \Gamma$. Index indicates position of a in the captured string:
$$\begin{cases} A_S & \text{for starting the string;} \\ A_{SE} & \text{for being a unique symbol in the string;} \\ A_E & \text{for ending the string;} \\ A & \text{for being internal in the string.} \end{cases}$$
- Tape alphabet Σ': for any $a \in \Sigma$, add $a_{\&} \in \Sigma'$.
- Reset processing: add $E_{SE} \in \Gamma$ and $\varepsilon_{\&} \in \Sigma'$.

Example

PDA model for $[_1\mathtt{a}^*]_1\&1$



Multiple Referencing

- $(\pi_1 \cup \pi_2)\pi_3$, where π_1, π_3 contain $\&1$, and π_2 does not, is unfolded to $\big(\pi_1\pi_3 \cup \pi_2\pi_3\big)$.

- $(\pi_1 \cup \pi_2)^*\pi_3$, where π_1 contains $\&1$, and π_2 does not, is unfolded to ${\pi_2}^* \left(\pi_3 \cup \big(\pi_1(\pi_1 \cup \pi_2)^*\big)\pi_3\right)$.

- The first occurrence of &1 along any trace is replaced by the pop block; the rest are interpreted symbolically (as tape symbols).

PDA Bisimulation is Hard

The 1-MFA reduction to PDA is of mere theoretical interest.

PDA Bisimulation

- Non-elementary complexity;
- No known implementation.

Model PDAs

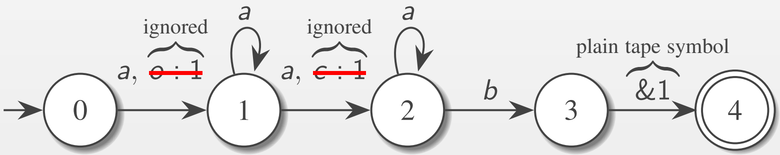
Fall into a narrow subset of PDA:

- Visible pop actions;
- Separate pop/push blocks.

MFA-Bisim Approximations: Action-Bisim

- $\mathscr{A}_1 \sim \mathscr{A}_2 \Rightarrow$ their projections containing only external actions must be bisimilar.
- $\pi_{\mathcal{M}}(\mathscr{A})$ is its *action NFA*; $\mathscr{A}_1$ and $\mathscr{A}_2$ are action-bisimilar $\Leftrightarrow \pi_{\mathcal{M}}(\mathscr{A}_1) \sim \pi_{\mathcal{M}}(\mathscr{A}_2)$.

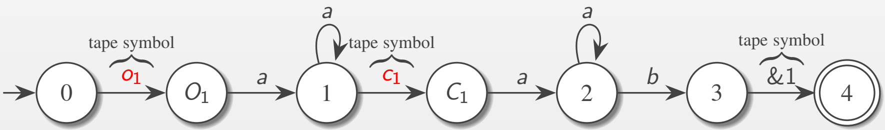
Action Automaton



MFA-Bisim Approximations: Symbolic-Bisim

- If memory actions become external, and the resulting NFA are bisimilar, then so are the initial MFA;
- $\pi^{\mathcal{M}}(\mathscr{A})$ is a *symbolic NFA*; $\mathscr{A}_1$ and $\mathscr{A}_2$ are symbolic-bisimilar $\Leftrightarrow \pi^{\mathcal{M}}(\mathscr{A}_1) \sim \pi^{\mathcal{M}}(\mathscr{A}_2)$.

Symbolic Automaton



Decisive Actions in MFA-based Games

- Given captured branching paths π_1 and π_2 starting in a state q of $\mathscr{A}_i$ s.t. $\pi_1 : q \xrightarrow{\omega_1\gamma_1\omega_2} q_1$, $\pi_2 : q \xrightarrow{\omega_1\gamma_2\omega_2} q_2$, player $\mathcal{D}$ must be able to imitate the captured branching in $\mathscr{A}_j$.

- Given captured loop π in a state q of $\mathscr{A}_i$, player $\mathcal{D}$ must be able to imitate the captured loop in $\mathscr{A}_j$.

- These two sorts of the actions are *decisive*. Any decisive action is not synchronised in $\mathscr{A}_i \Rightarrow$ player $\mathcal{A}$ has a winning strategy.

Action Reachability

- The ordering induced on the decisive actions by their mutual reachability partitions action-bisimilar states in $\pi_{\mathcal{M}}(\mathscr{A}_1)$ and $\pi_{\mathcal{M}}(\mathscr{A}_2)$ into equivalence classes forming a partial order $\preceq$.
- A narrowing of $\sim$ on $\pi_{\mathcal{M}}(\mathscr{A}_i)$ w.r.t. $\preceq$ does not exist $\Rightarrow$ player $\mathcal{A}$ has a winning strategy.

Example

$\mathtt{a}^*[_1\mathtt{a}^*]_1\&1$ and $[_1\mathtt{a}^*]_1\mathtt{a}^*\&1$

Memory Revision

- Word equations induced on the stored strings can be checked trivially, since the strings are parametrisable.
- The stored parameterised words are not equal in $\mathscr{A}_i$ when performing a synchronised reference $\Rightarrow$ player $\boldsymbol{\mathcal{A}}$ has a winning strategy.

Example

MFA for $[_1\mathtt{a}\mathtt{a}^*]_1\mathtt{a}\&1$ and $\mathtt{a}[_1\mathtt{a}^*\mathtt{a}]_1\&1$ are bisimilar, because $\forall n \in \mathbb{N}(aa^n = a^n a)$

Practical Implementation

Calculate $\pi_{\mathcal{M}}(\mathscr{A}_1)$ and $\pi_{\mathcal{M}}(\mathscr{A}_2)$ states bisimulation relation. It must exist.

↓

Ensure that decisive states are bisimilar. Narrow bisimulation of decisive states by taking reachability relation into account.

↓

Find bisimilar states with incoming matching references.

↓

Perform a memory revision in all the closest preceding capture groups. Memory equality will indicate bisimulation.

MFA Generation & Fuzzing

Fuzz module comprises:

- Ref-word generator parameterized by alphabet, length, Kleene stars number, number of memory cells and stars nesting height.
- MFA generator parameterized by alphabet, number of states, edges and memory cells.
- Comparative string parsing by automata.

Testing Suite

The relations hold for MFA and their approximations:

$$\left(\sim^{\mathcal{M}}\right) \subseteq \left(\sim\right) \subseteq \left(\sim_{\mathcal{M}}\right)$$

- Generate arbitrary ref-word-based MFA $\mathscr{A}_1$ and $\mathscr{A}_2$;
- Check whether $\mathscr{A}_1 \sim' \mathscr{A}_2$, $\boldsymbol{\pi}_{\mathcal{M}}(\mathscr{A}_1) \sim \boldsymbol{\pi}_{\mathcal{M}}(\mathscr{A}_2)$, $\boldsymbol{\pi}^{\mathcal{M}}(\mathscr{A}_1) \sim \boldsymbol{\pi}^{\mathcal{M}}(\mathscr{A}_2)$.
- If $\left(\boldsymbol{\pi}_{\mathcal{M}}(\mathscr{A}_1) \sim \boldsymbol{\pi}_{\mathcal{M}}(\mathscr{A}_2)\right) \neq \left(\boldsymbol{\pi}^{\mathcal{M}}(\mathscr{A}_1) \sim \boldsymbol{\pi}^{\mathcal{M}}(\mathscr{A}_2)\right)$, check the bisimulation correctness $\sim'$ manually.

Discussion

- Bisimulation can be a decent approximation of the language equivalence. In addition, bisimulation preserves capture groups, unlike the equivalence.

$$\boxed{[_1\mathtt{a}^*]_1\&1 \xrightarrow{\text{minimize}} (\mathtt{aa})^*}$$

- Generic multiple cells case — decidability is unlikely.
- Acyclic multiple cells case — PDA reduction is not obvious because of recapturing. Memory revision is recursive, yet likely converging.
- The $\sim^{\mathcal{M}}$ and $\sim_{\mathcal{M}}$ approximations are fast filters for non-bisimilar and non-trivially-bisimilar cases.

The project Chipollino:

- **Many more automata functions;**
- **Visualisation in TikZ.**