Transducers in Automata Library Mata

BRNO UNIVERSITY OF TECHNOLOGY — FACULTY OF INFORMATION TECHNOLOGY

Author: *Ing. David Chocholatý* Supervisor: *doc. Mgr. Lukáš Holík Ph.D.*

Brno University of Technology, Faculty of Information Technology, Brno, Czech Republic

Example of a JavaScript code vulnerable to XSS without proper string sanitization.

```
var text = htmlEscape(goal);
var action = escapeString(text);
element.innerHTML = '<button \
    onclick="performAction(\''
    + action + '\')">' + text
    + '</button>';
```

```
<button onclick="performAction(
    '&#39;);alert(1);//')">
     &#39;);alert(1);//')</button>
```

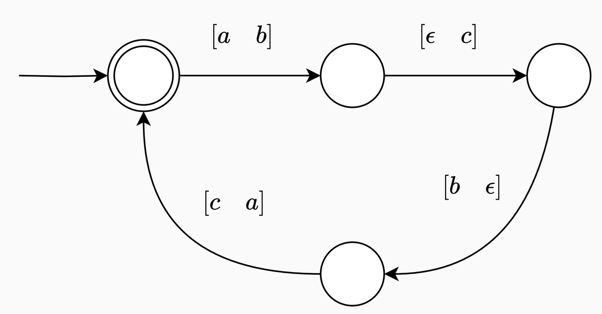
A malicious user input on a web application can lead to a XSS attack executing the malicious action `alert(1)`.

Motivation

- Transducers are very useful in **SMT string solving** to, e.g. **verify security properties of web applications**.
- Transducers can be used to model string functions, such as `replace` and *ReplaceAll*() used to describe string constraints for **secure handling of string inputs** given by the user of web applications.
- String solving **prevents attacks** such as **cross-site scripting (XSS)** by using transducers to model *string sanitization* or *implicit browser transductions*.

Finite Transducers

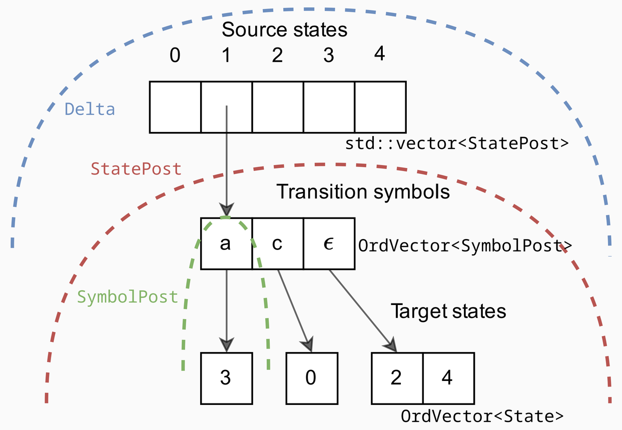
- (Non)Deterministic Finite Tranaducers (NFTs) are **finite state machines with n memory tapes**.
- Each run of a finite transducer represents an n-tuple of words. While a finite automaton models a regular language, finite transducers model rational relations, subsets of the Cartesian product of regular languages. That is, a finite transducer models a set of n-tuples of words.
- 2-tape NFT tapes are usually called an *input* and an *output* tape.
- 5-tuple $\mathcal{T} = (Q, \Gamma, post, I, F)$ where
  - $\Gamma = (\Sigma_\epsilon)^n$ is an n-tape alphabet of $\mathcal{T}$,
  - $post : Q \times \Gamma \to 2^Q$ is a *symbol-post function*
- A 2-tape NFTs can be percieved as a translator translating from one language (on the first tape) to another (the second tape).
- A NFT accepting $(abc)^*$ on the first tape which translates all a symbols to bc, removes all b symbols, and replaces all c symbols with a can be represented as follows.



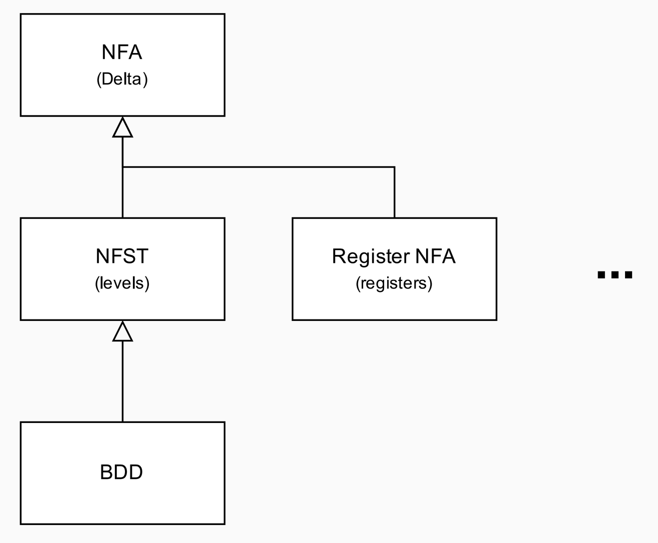
- Word $abcabc$ is translated to $bcaba$.

Automata Library Mata: A Fast and Simple Finite Automata Library

- Mata is a C++ library for **manipulating finite automata**.
- Mata is designed to be **simple** to use for researchers and students, yet provide an **efficient** library for automata operations.
- The efficiency of Mata is achieved by using algorithms utilizing the features of the underlying **data structures**. E.g. the 3-level `Delta` representing the transition relation.
- The goal for NFTs is to **reuse the existing automata operations and data structures** in Mata with minimal modifications, namely to reuse `Delta`.



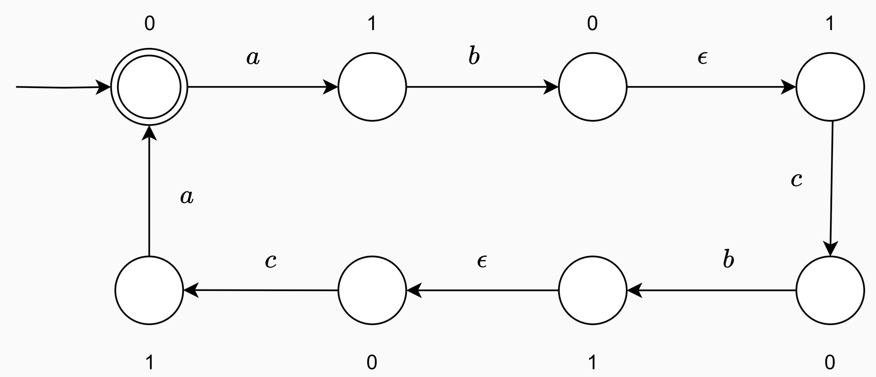
Inheritance hierarchy of automata models in Mata.



Idea

- We have implemented the representtion for NFT in Mata as a **simple finite automaton with states annotated by levels where each level corresponds to a single tape in the NFT**.
- **Transition unfolding**: NFT transition is a sequence of NFA transitions.
- `Nft` inherits from the base class `Nfa` for the nondeterministic finite automata.
- This ensures most of the existing operations can be **reused and only the operations specific to NFTs need to be implemented.**
- This also gives rise to an implementation for **automata with BDD-like transitions** represented as an NFT with different interpretation for each NFT transition.

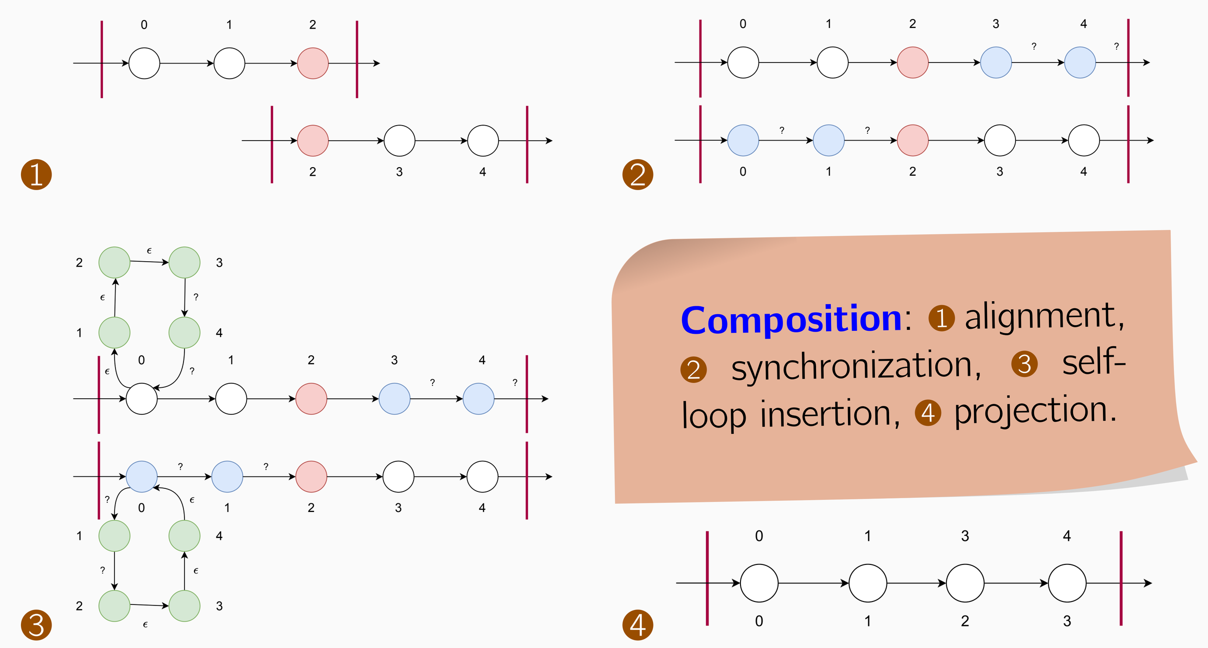
Key Contribution: Design and Implementation of Finite Transducers in Mata



- Representation of the same NFT as earlier in Mata.
- **word unfolding**: The accepted word is an unfolded sequence of words from the NFT.

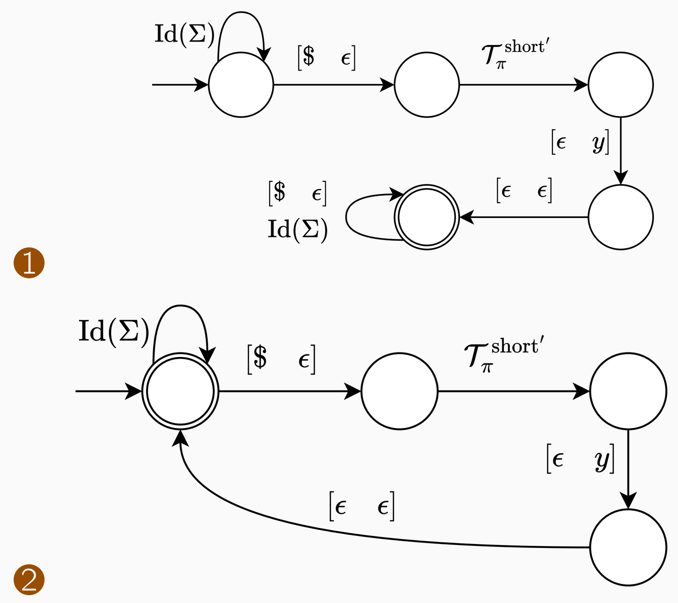
$$\bar{w} = (a_1^1, a_1^2, \ldots, a_1^n) \circ (a_2^1, a_2^2, \ldots, a_2^n) \circ \ldots \circ (a_m^1, a_m^2, \ldots, a_m^n)$$
$$\bar{w} = a_1^1 \cdot a_1^2 \cdot \ldots \cdot a_1^n \cdot a_2^1 \cdot a_2^2 \cdot \ldots \cdot a_2^n \cdot \ldots \cdot a_m^1 \cdot a_m^2 \cdot \ldots \cdot a_m^n$$

- States with levels 0 represent the beginning of an NFT transition.
- NFTs support **epsilon transitions** and **don't care symbols** on the tapes as special symbols.
- These special symbols can be composed into **long jumps over several tapes**, but only up to the end of the current NFT transition.
- NFTs support operations such as: **synchronization on tapes**, **application** on a word or a language, **composition**, and various **projections**.
- Thanks to our approach, the NFT operations can be implemented as a simple combination of existing NFA operations.



Composition: ❶ alignment, ❷ synchronization, ❸ self-loop insertion, ❹ projection.

Support for String Solving



- NFTs can be used to model string functions such as `replace` and *ReplaceAll*().
- Such replace operations can model **string sanitization** or **implicit browser transductions**.
- NFTs in Mata support replacement operation `replace(word, regex, replacement)` where
  - ❶ **single** or ❷ **all** occurrence(s) can be replaced, and
  - the string being replaced can be matched as a **literal** or a **regex**.

Experimental Evaluation

- Benchmarks from **SMT-LIB** (`QF_S/{webapp, transducer-plus}`) and **pattern matching** (`SymbolFromEnd`).
- Comparison with automata library Mona.
- NFTs in Mata have support for **nondeterminism**.

Thesis

vut.cz/en/students/final-thesis?zp_id=155757

- **SymbolFromEnd**: `#.*a.{i}$` $\to \epsilon$, $i \in \{1, \ldots, 150\}$

method	TOs	min	max	mean	q(0.25)	median	q(0.75)	std. dev
mata	0	0.97	8070.18	2397.81	461.98	1743.10	4035.48	2233.84
mona	272	0.29	13095.06	1614.42	2.39	39.25	970.33	3485.11

method	TOs	min	max	mean	q(0.25)	median	q(0.75)	std. dev
mata	0	1.79	7900.85	2211.33	446.27	1623.44	3680.46	1983.68
mona	232	1.04	12664.70	2987.85	79.46	1304.86	4614.69	3791.32

Times are in milliseconds.