Finite-Valued Streaming String Transducers

Sarah Winter
IRIF, Université Paris Cité, France

based on joint work with
Emmanuel Filiot, Ismaël Jecker, Christof Löding, Anca Muscholl, Gabriele Puppis

August 22 @ RAMiCS 2024, Prague, Czech Republic

Finite Automata and Languages

Finite automata define a very robust class of languages

- many equivalent automaton models: deterministic, nondeterministic, 2-way, ε-transitions
- various other representations: regular expressions, MSO logic, monoids
- excellent closure-properties: Boolean operations, projection, homomorphisms, reversal, ...
- many interesting problems decidable: equivalence, emptiness, universality, ...

(Word-)Transductions

A **(word-)transduction** is a relation $R \subseteq \Sigma^* \times \Sigma^*$ between words.

Examples.

reverse	$abaabba \mapsto abbaaba$
copy	$abaabba \mapsto abbaabaabbaaba$
sort	$abaabba \mapsto aaaabbb$
delete as	$abaabba \mapsto bbb$
infix	$abb \mapsto \varepsilon, a, ab, abb$
rotate	$abb \mapsto abb, bba, bab$
iterate	$abb \mapsto abb, abbabb, abbabbabb, \ldots$

(Word-)Transductions

A **(word-)transduction** is a relation $R \subseteq \Sigma^* \times \Sigma^*$ between words.

Examples.

reverse	$abaabba \mapsto abbaaba$
copy	$abaabba \mapsto abbaabaabbaaba$
sort	$abaabba \mapsto aaaabbb$
delete as	$abaabba \mapsto bbb$
infix	$abb \mapsto \varepsilon, a, ab, abb$
rotate	$abb \mapsto abb, bba, bab$
iterate	$abb \mapsto abb, abbabb, abbabbabb, \ldots$

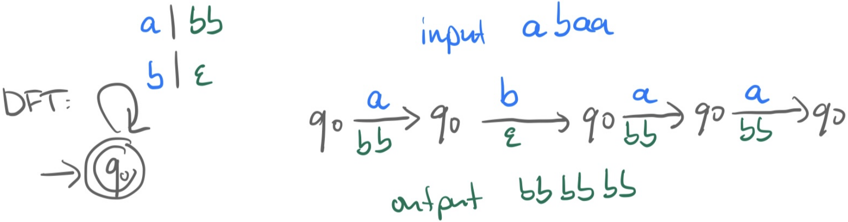
Transductions are defined by (finite) transducers (automata with output). Unlike for automata, the defined classes of transductions vary by transducer model.

Transducer Models

Finite Transducers

A **finite transducer** (**FT**) is a finite automaton that additionally has output words on its transitions.

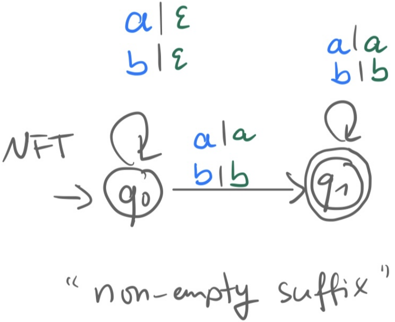
Example. Deterministic FT.



Finite Transducers

A **finite transducer** (**FT**) is a finite automaton that additionally has output words on its transitions.

Example. Nondeterministic FT.



Properties of Finite Transducers

- DFTs define functions, NFTs can define relations.
- It is decidable whether an NFT defines a function (Schützenberger 1975).
- Equivalence is decidable for DFTs (Blattner, Head 1979).
- Equivalence is undecidable for NFTs (Fischer, Rosenberg 1968).
- Fewer closure-properties than finite automata, e.g., FTs are not closed under intersection.

Properties of Finite Transducers

- DFTs define functions, NFTs can define relations.
- It is decidable whether an NFT defines a function (Schützenberger 1975).
- Equivalence is decidable for DFTs (Blattner, Head 1979).
- Equivalence is undecidable for NFTs (Fischer, Rosenberg 1968).
- Fewer closure-properties than finite automata, e.g., FTs are not closed under intersection.

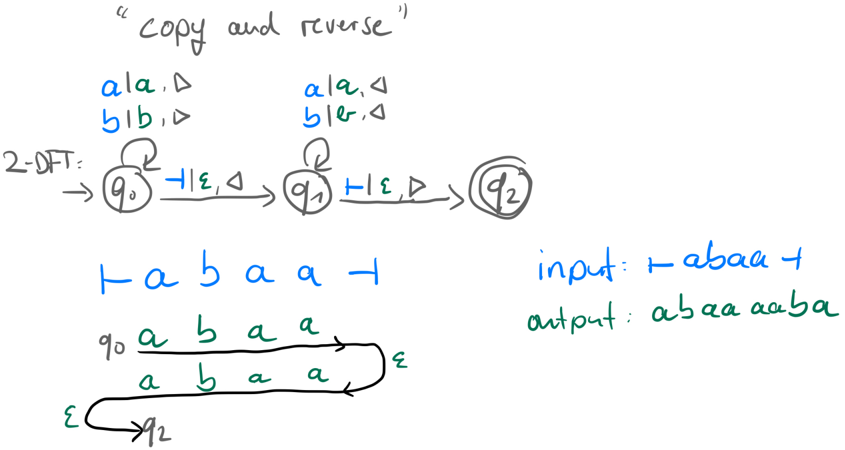
Drawback

- FTs are limited in their expressiveness, for example "copy", "reverse", "sort", ... are not definable.
- Is there a more expressive model?

2-way Finite Transducers

A **2-way finite transducer** (**2-FT**) can move left and right on its input tape and produce output from left to right.

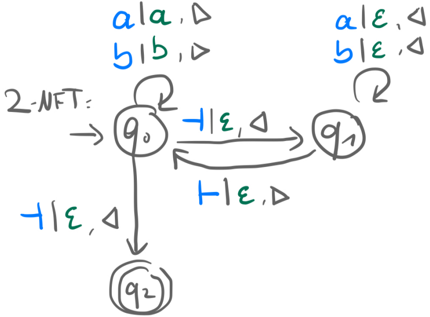
Example. Deterministic 2-FT.



2-way Finite Transducers

A **2-way finite transducer** (**2-FT**) can move left and right on its input tape and produce output from left to right.

Example. Nondeterministic 2-FT.



"iterate"

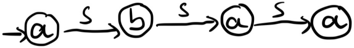
$u \mapsto \{u^n \mid n \geq 1\}$

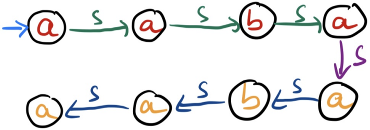
MSO Transductions

A **monadic second order logic transduction** (**MSOT**) takes a fixed number of copies of the universe of the input structure, and defines the relations of the output structure by MSO formulas.

Example. Deterministic 2-FT.

"copy and reverse"





$$\varphi_S^{1,1}(x,y) := S(x,y)$$

$$\varphi_S^{1,2}(x,y) := x = y \wedge \mathrm{last}(x)$$

$$\varphi_S^{2,2}(x,y) := S(y,x)$$

$$\varphi_S^{2,1}(x,y) := \bot$$

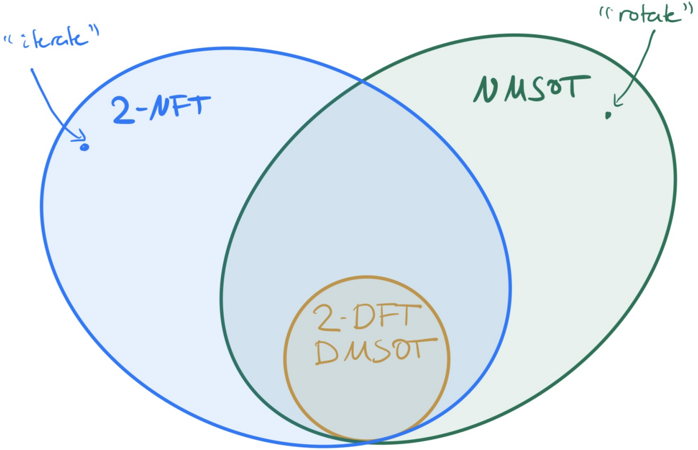
$$\varphi^1_{init}(x) := \mathrm{init}(x), \quad \varphi^2_{init}(x) := \bot$$

$$\varphi^1_\sigma(x) := \sigma(x), \quad \sigma \in \Sigma$$

$$\varphi^2_\sigma(x) := \sigma(x), \quad \sigma \in \Sigma$$

Overview: MSOT and 2-FT

Theorem (Engelfriet, Hogeboom 1988).
DMSOT and 2-DFT have the same expressive power. The classes of transductions defined by NMSOT and 2-NFT differ.



An Equivalent 1-way Model?

- The connection between MSO logic and finite automata is a cornerstone of the analysis of logical specifications.
- We have such a connection between MSO transductions and 2-way finite transducers.
- Unfortunately, reasoning with 2-way models can be quite technical and involved.
- Is there a 1-way model that expresses MSO definable transductions?
- Also, implementation-wise a 1-way model might be preferred.

Streaming String Transducers

A **Streaming String Transducer** (**SST**) is a finite automaton with a set $\mathcal{X}$ of output registers. Transitions are additionally annotated with register updates, one for each register $X \in \mathcal{X}$, of the form

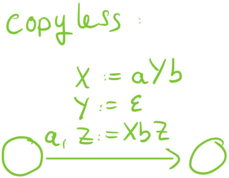
$$X := w_1 X_1 w_2 \cdots w_n X_n w_{n+1} \text{ with } X_i \in \mathcal{X} \text{ and } w_i \in \Sigma^*$$

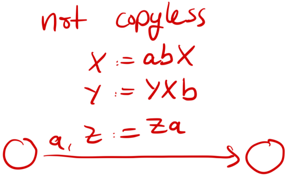
Streaming String Transducers

A **Streaming String Transducer** (**SST**) is a finite automaton with a set $\mathcal{X}$ of output registers. Transitions are additionally annotated with register updates, one for each register $X \in \mathcal{X}$, of the form

$$X := w_1 X_1 w_2 \cdots w_n X_n w_{n+1} \text{ with } X_i \in \mathcal{X} \text{ and } w_i \in \Sigma^*$$

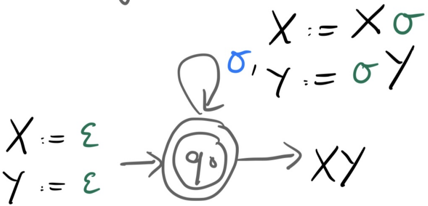
Register updates are required to be **copyless**: each register appears at most once in the right-hand side of the updates in a transition.

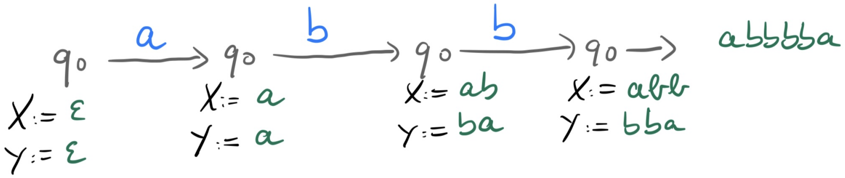




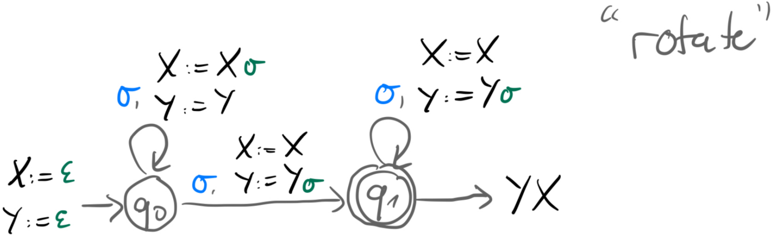
Example: Deterministic SST

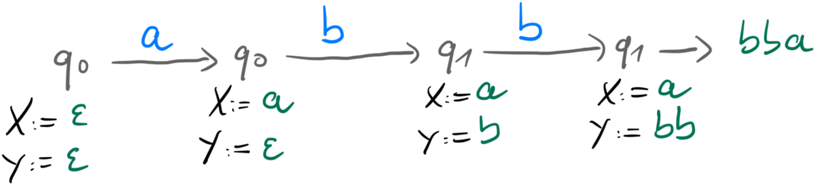
"copy and reverse"





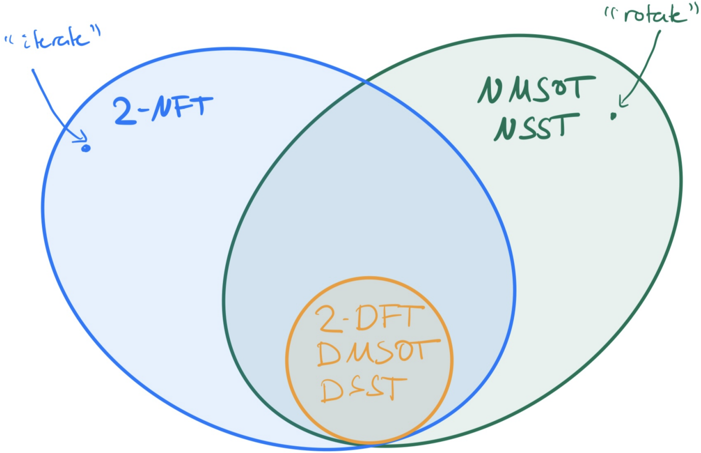
Example: Nondeterministic SST



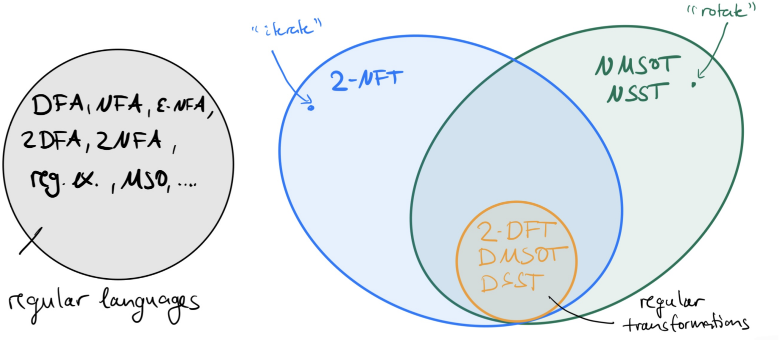


Overview: Adding SSTs

Theorem (Alur, Černy 2010; Alur, Deshmukh 2011). DMSOT and DSST have the same expressive power. So have NMSOT and NSST.



Languages vs. Transductions



Inbetween Functions and Relations

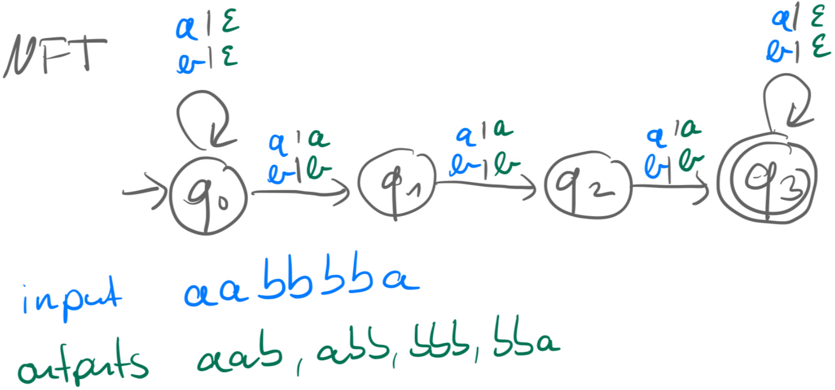
- Equivalence is decidable for 2-DFTs, DMSOTs, and DSSTs, while it is undecidable for their nondeterministic counter-parts.
- For NFTs, there is a robust subclass, namely NFTs defining finite-valued relations.
- Does this robustness extend to finite-valued relations defined by 2-NFTs, NMSOTs, NSSTs?

Finite-valued Transducers

Finite-valued Transducers

A transduction T is called **finite-valued** if there is a bound k such that T associates at most k outputs to each input.

Example.



Properties of Finite-valued FTs

A transduction T is called **finite-valued** if there is a bound k such that T associates at most k outputs to each input.

- Equivalence for finite-valued NFT is decidable (Culik, Karhumäki 1986).
- It is decidable if a given NFT is finite-valued (Weber 1990).
- Every k-valued NFT can be effectively decomposed into a union of k single-valued NFT (Weber 1993).

Properties of Finite-valued FTs

A transduction T is called **finite-valued** if there is a bound k such that T associates at most k outputs to each input.

- Equivalence for finite-valued NFT is decidable (Culik, Karhumäki 1986).
- It is decidable if a given NFT is finite-valued (Weber 1990).
- Every k-valued NFT can be effectively decomposed into a union of k single-valued NFT (Weber 1993).
- Decomposition allows for a new test for equivalence.

Given NFTs T_0, T finite-valued

Check $T_0 \stackrel{?}{\subseteq} \bigcup_{i=1}^{n} T_i$, T_i single-valued

What About Finite-valued SSTs?

NSST were introduced by (Alur, Deshmukh 2011). The authors raised the following questions:

- Is finite-valuedness of NSST decidable?
- Is equivalence for NSST decidable?
- Can every finite-valued NSST be decomposed into a finite union of DSST?

What About Finite-valued SSTs?

NSST were introduced by (Alur, Deshmukh 2011). The authors raised the following questions:

- Is finite-valuedness of NSST decidable?
- Is equivalence for NSST decidable?
- Can every finite-valued NSST be decomposed into a finite union of DSST?

We give positive answers to all these questions and consequently obtain also results about finite-valued 2-NFTs and NMSOTs.

Results

Finite-valuedness

Theorem (FJLMPW 2024). It is decidable (PSPACE-complete) whether a nondeterministic SST is finite-valued.

Results

Finite-valuedness

Theorem (FJLMPW 2024). It is decidable (PSPACE-complete) whether a nondeterministic SST is finite-valued.

Decomposition

Theorem (FJLMPW 2024). Every k-valued SST can be effectively decomposed into a union of k deterministic SST.

Consequences of Decomposition Result

Together with a result of (Alur, Deshmukh 2011), we obtain:

Corollary. Equivalence for k-valued SST is decidable in elementary time.

Decidability was already known (Muscholl, Puppis 2019), but without an elementary upper complexity bound.

Consequences of Decomposition Result

Together with a result of (Alur, Deshmukh 2011), we obtain:

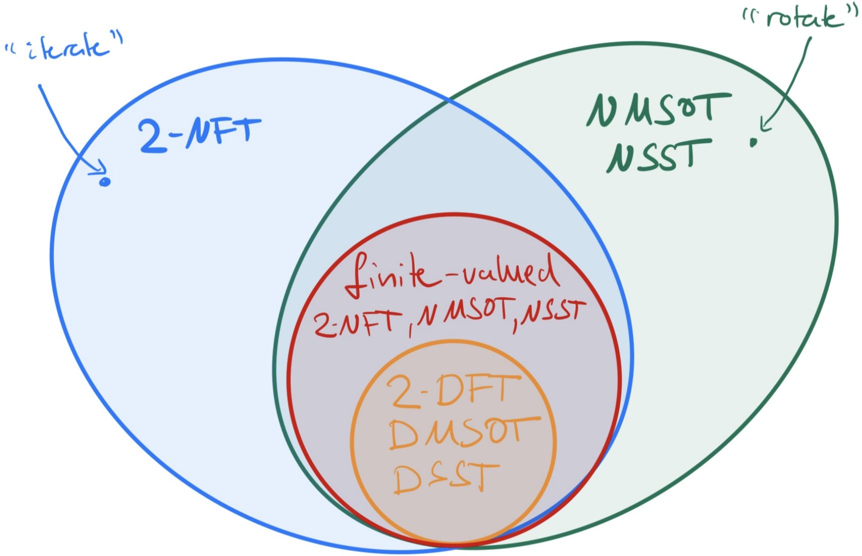
Corollary. Equivalence for k-valued SST is decidable in elementary time.

Decidability was already known (Muscholl, Puppis 2019), but without an elementary upper complexity bound.

Corollary. For finite-valued relations, the classes of 2-NFT, NSST, and NMSOT coincide.

The decomposition entails a translation from finite-valued NSST to 2-NFT. The other direction was already known (Alur, Černy 2011).

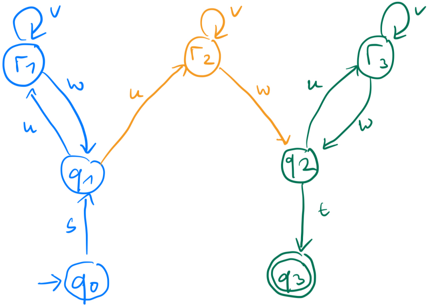
Overview: Adding Finite-valued Relations



Deciding Finite-valuedness

Characterization for Finite-Valuedness

Lemma. An SST is finite-valued iff it does not contain a “simply divergent W-pattern”.



Exist $n_1, \dots, n_5 \in \{1,2\}$ such that the runs below produce different outputs

$$q_0 \xrightarrow{s} q_1 \xrightarrow{uv^{n_1}w} q_1 \xrightarrow{uv^{n_2}w} q_1 \xrightarrow{uv^{n_3}w} q_1 \xrightarrow{uv^{n_4}w} q_2 \xrightarrow{uv^{n_5}w} q_2 \xrightarrow{t} q_3$$

$$q_0 \xrightarrow{s} q_1 \xrightarrow{uv^{n_1}w} q_1 \xrightarrow{uv^{n_2}w} q_2 \xrightarrow{uv^{n_3}w} q_2 \xrightarrow{uv^{n_4}w} q_2 \xrightarrow{uv^{n_5}w} q_2 \xrightarrow{t} q_3$$

Differences between FTs and SSTs

Finite-valuedness is characterized for FTs and SSTs via "divergent W-patterns". Main ingredients to establish the characterization are

- a pumping technique for loops, and
- comparing the "delay" between runs on the same input.

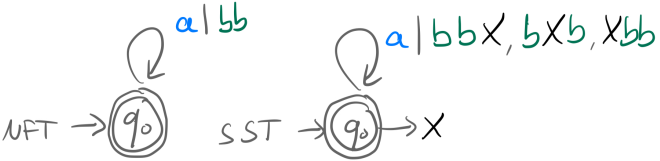
Differences between FTs and SSTs

Finite-valuedness is characterized for FTs and SSTs via "divergent W-patterns". Main ingredients to establish the characterization are

- a pumping technique for loops, and
- comparing the "delay" between runs on the same input.

FTs build their output from left-to-right, while SSTs do not have this restriction.

Example.



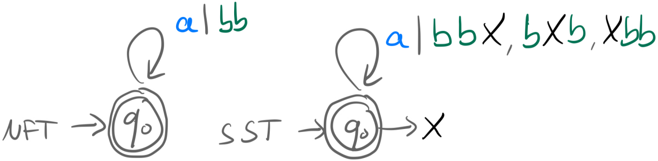
Differences between FTs and SSTs

Finite-valuedness is characterized for FTs and SSTs via "divergent W-patterns". Main ingredients to establish the characterization are

- a pumping technique for loops, and
- comparing the "delay" between runs on the same input.

FTs build their output from left-to-right, while SSTs do not have this restriction.

Example.



This makes it necessary to develop a new pumping technique and a new notion of "delay". A suitable notion of "delay" was introduced in (Filiot, Jecker, Löding, W. 2023).

Skeleton-idempotent Loops

The **skeleton** of an update $\alpha : \mathcal{X} \to (\Sigma \uplus \mathcal{X})^*$ is the update $\hat{\alpha} : \mathcal{X} \to \mathcal{X}^*$ obtained by removing all letters from Σ.

Skeletons and their composition form a finite monoid.

A **skeleton-idempotent loop** is a factor of a run that starts and ends in the same state and induces a skeleton-idempotent update (that is an update α so that α and $\alpha \cdot \alpha$ have the same skeleton).

Example.

$$\alpha : \begin{array}{l} X_1 := a X_1 b X_2 c \\ X_2 := a \end{array} \qquad \hat{\alpha} : \begin{array}{l} X_1 := X_1 X_2 \\ X_2 := \varepsilon \end{array}$$

$$\alpha \cdot \alpha : \begin{array}{l} X_1 := a\, \alpha(X_1)\, b\, \alpha(X_2)\, c = a a X_1 b X_2 c b a c \\ X_2 := a \end{array}$$

Skeleton-idempotent Loops

Example.

α : $X_1 := a X_1 b X_2 c$
$X_2 := a$

$\hat{\alpha}$: $X_1 := X_1 X_2$
$X_2 := \varepsilon$

$\alpha \cdot \alpha$: $X_1 := a\, \alpha(X_1)\, b\, \alpha(X_2)\, c = a a X_1 b X_2 c b a c$
$X_2 := a$

α^3 : $X_1 := a a a X_1 b X_2 c b a c b a c$
$X_2 := a$

Pumping Skeleton-idempotent Loops

Let α be a skeleton-idempotent update. For every $X \in \mathcal{X}$ there exist two words $u, v \in \Sigma^*$ such that

$$\alpha^n(X) = u^{n-1}\alpha(X)v^{n-1}.$$

Pumping Skeleton-idempotent Loops

Let α be a skeleton-idempotent update. For every $X \in \mathcal{X}$ there exist two words $u, v \in \Sigma^*$ such that

$$\alpha^n(X) = u^{n-1}\alpha(X)v^{n-1}.$$

Example.

$$\alpha : \begin{array}{l} X_1 := a X_1 b X_2 c \\ X_2 := a \end{array}$$

$$\alpha^n(X_1) = (a)^{n-1}\alpha(X_1)(bac)^{n-1}$$

$$\alpha^n(X_2) = \alpha(X_2)$$

Pumping Skeleton-idempotent Loops

Let α be a skeleton-idempotent update. For every $X \in \mathcal{X}$ there exist two words $u, v \in \Sigma^*$ such that

$$\alpha^n(X) = u^{n-1}\alpha(X)v^{n-1}.$$

Example.

$$\alpha : \begin{array}{l} X_1 := a X_1 b X_2 c \\ X_2 := a \end{array}$$

$$\alpha^n(X_1) = (a)^{n-1}\alpha(X_1)(bac)^{n-1}$$

$$\alpha^n(X_2) = \alpha(X_2)$$

A Ramsey-type argument shows that in a long enough run a sequence of (pairwise disjoint) skeleton-idempotent loops occur.

Pumping Skeleton-idempotent Loops

Given a run with m such loops, pumping the i-th loop n_i times yields output of the form

$$w_0(u_1)^{k_1-1}w_1(u_2)^{k_2-1}\cdots w_{r-1}(u_r)^{k_r-1}w_r,$$

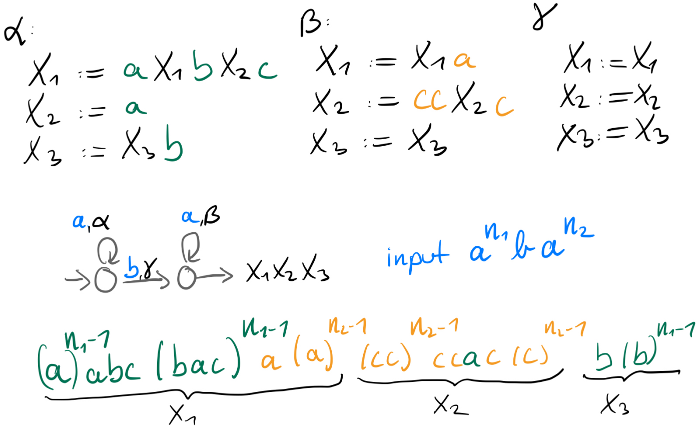
where r is bounded by $2m|\mathcal{X}|$ and $k_1,\ldots,k_1 \in \{n_1,\ldots,n_m\}$.

Pumping Skeleton-idempotent Loops

Given a run with m such loops, pumping the i-th loop n_i times yields output of the form

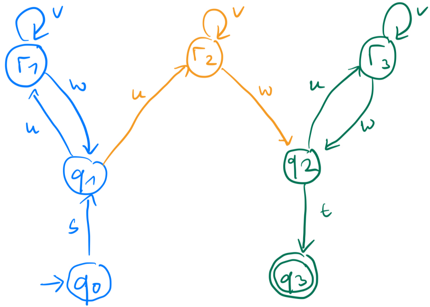
$$w_0(u_1)^{k_1-1}w_1(u_2)^{k_2-1}\cdots w_{r-1}(u_r)^{k_r-1}w_r,$$

where r is bounded by $2m|\mathcal{X}|$ and $k_1,\ldots,k_1 \in \{n_1,\ldots,n_m\}$.



Pumping Skeleton-idempotent Loops

Goal: Use the "simply divergent W-pattern" to create a set of runs (via pumping) with the same input but different outputs.



Exist $n_1, \ldots, n_5 \in \{1, 2\}$ such that the runs below produce different outputs

$$q_0 \xrightarrow{s} q_1 \xrightarrow{uv^{n_1}w} q_1 \xrightarrow{uv^{n_2}w} q_1 \xrightarrow{uv^{n_3}w} q_1 \xrightarrow{uv^{n_4}w} q_2 \xrightarrow{uv^{n_5}w} q_2 \xrightarrow{t} q_3$$

$$q_0 \xrightarrow{s} q_1 \xrightarrow{uv^{n_1}w} q_1 \xrightarrow{uv^{n_2}w} q_2 \xrightarrow{uv^{n_3}w} q_2 \xrightarrow{uv^{n_4}w} q_2 \xrightarrow{uv^{n_5}w} q_2 \xrightarrow{t} q_3$$

Word Inequalities

A **word inequality with parameters** is an inequality of two words in which repetitions of some subwords are parameterized by variables.

A **solution** is an assignment of numbers to the variables such that the resulting words are different.

Word Inequalities

A **word inequality with parameters** is an inequality of two words in which repetitions of some subwords are parameterized by variables.

A **solution** is an assignment of numbers to the variables such that the resulting words are different.

Example.

- $(ab)^{\mathtt{x}} aa (b)^{\mathtt{x}} abba \neq ababaa(bba)^{\mathtt{x}}$ (one parameter)

  The only non-solution is $\mathtt{x} = 2$:

  $ababaabbabba =$
  $ababaabbabba$

Word Inequalities

A **word inequality with parameters** is an inequality of two words in which repetitions of some subwords are parameterized by variables.

A **solution** is an assignment of numbers to the variables such that the resulting words are different.

Example.

- $(ab)^{\mathtt{x}}aa(b)^{\mathtt{x}}abba \neq ababaa(bba)^{\mathtt{x}}$ (one parameter)

  The only non-solution is $\mathtt{x} = 2$:

  $ababaabbabba =$
  $ababaabbabba$

- $b(ab)^{\mathtt{y}}ab(b)^{\mathtt{x}} \neq (ba)^{\mathtt{x}}ba(b)^{\mathtt{y}}b$ (two parameters)

  Non-solutions are all choices such that $\mathtt{x} = \mathtt{y}$,
  Solutions are all choices such that $\mathtt{x} \neq \mathtt{y}$.

Saarela and Consequences

Theorem (Saarela 2015). A word inequality with a single parameter x either has no solutions or the set of solutions is co-finite (the number of non-solutions is bounded by the number of occurences of x in the inequality).

Saarela and Consequences

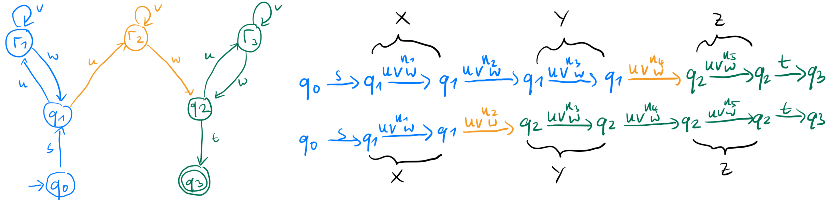
Theorem (Saarela 2015). A word inequality with a single parameter x either has no solutions or the set of solutions is co-finite (the number of non-solutions is bounded by the number of occurences of x in the inequality).

Consequences

- We show properties of the solution space for word inequalities with multiple parameters.
- We show that if each inequality in a finite system of inequalities is solvable, then the system is solvable.

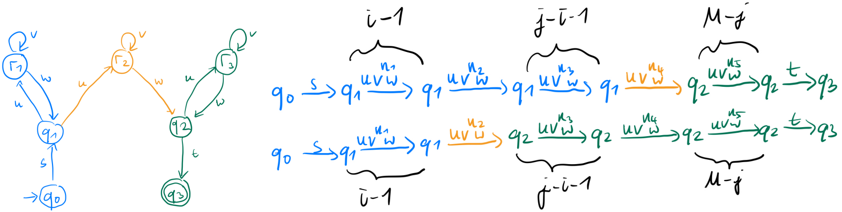
“simply divergent W-pattern” $\Rightarrow$ not finite-valued

- Pattern yields two runs whose outputs have the right format for a word inequality with parameters $\mathtt{x},\mathtt{y},\mathtt{z}$.

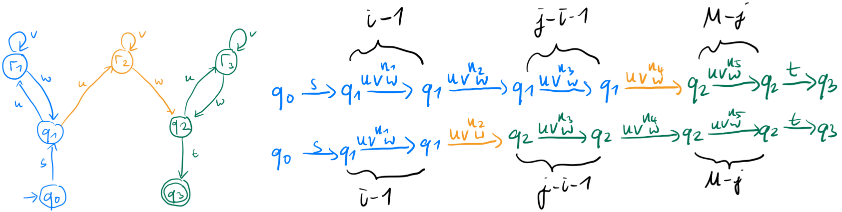


- Since there is one solution ($\mathtt{x} = 1$,$\mathtt{y} = 1$,$\mathtt{z} = 1$), the set of solutions is infinite (and obeys some properties).
- We show that ($\mathtt{x} = i - 1$,$\mathtt{y} = j - i - 1$,$\mathtt{z} = M - j$) for all $i < j$ (with i, j from a specific set) for some arbitrarily large M is a solution.

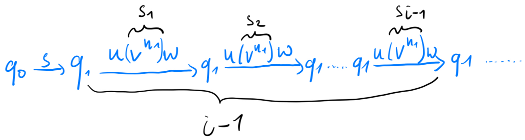
“simply divergent W-pattern” ⇒ not finite-valued



“simply divergent W-pattern” ⇒ not finite-valued



- We now parameterize each v^{n_1} in $(uv^{n_1}w)^{i-1}$, each v^{n_3} in $(uv^{n_3}w)^{j-i-1}$, and each v^{n_5} in $(uv^{n_5}w)^{M-j}$.



- Iterating through all $i < j$ forms a finite system of word inequalities. It has a solution as each inequality has a solution.
- Each inequality is generated by a run with the same input. Hence, the SST is not finite-valued.

Summary

- We completed the picture for finite-valued SSTs concerning their expressive power and answered key decidability questions.
- Future work: Complexities are likely not optimal.