Copyful Streaming String Transducers

Emmanuel Filiot

ULB, Computer Science Department

Pierre-Alain Reynier

~~LIF~~ LIS, Aix-Marseille University & CNRS

Δ meeting, ~~Porquerolles, Oct. 2017~~
IRIF, March 2018

Overview

Overview

Streaming String Transducers [Alur, Cerny, 2010]

SST= Deterministic Finite-state Automata extended with registers

Streaming String Transducers [Alur, Cerny, 2010]

SST= Deterministic Finite-state Automata extended with registers

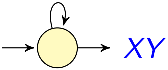
Registers: output words

$w \mapsto w\ .\ mirror(w)$

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

$\sigma,\ upd_\sigma : \begin{cases} X := X.\sigma \\ Y := \sigma Y \end{cases}$



X, Y, Z: registers
u, v: words in Σ^*

Streaming String Transducers [Alur, Cerny, 2010]

SST= Deterministic Finite-state Automata extended with registers

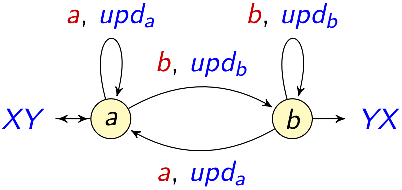
Registers: output words

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

X, Y, Z: registers
u, v: words in Σ^*

$$w \mapsto \begin{cases} w \,.\, mirror(w) \text{ if } last(w) = a \\ mirror(w) \,.\, w \text{ if } last(w) = b \end{cases}$$



Streaming String Transducers [Alur, Cerny, 2010]

SST= Deterministic Finite-state Automata extended with registers

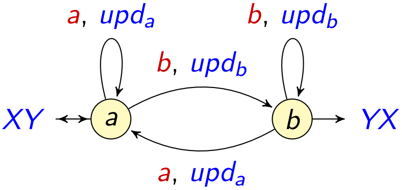
Registers: output words

Register updates:

- $X := u \cdot Y \cdot v$
- $X := YZ$

X, Y, Z: registers
u, v: words in Σ^*

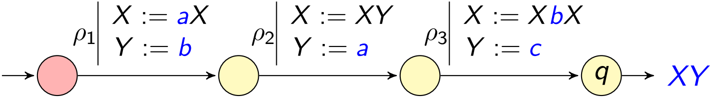
$$w \mapsto \begin{cases} w \,.\, mirror(w) \text{ if } last(w) = a \\ mirror(w) \,.\, w \text{ if } last(w) = b \end{cases}$$



Copyless restriction: a register cannot be copied

$$\text{OK} \begin{cases} X := XY \\ Y := aZb \\ Z := bb \end{cases} \qquad \text{KO} \begin{cases} X := Ya \\ Y := X \\ Z := bY \end{cases} \qquad \text{KO} \begin{cases} X := YaY \\ Y := XZ \\ Z := bb \end{cases}$$

Semantics of SST: two views

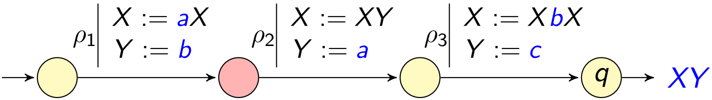


Forward interpretation: register valuations

X: ε

Y: ε

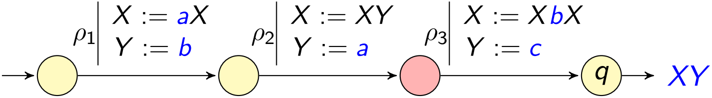
Semantics of SST: two views



Forward interpretation: register valuations

X:	ε	a
Y:	ε	b

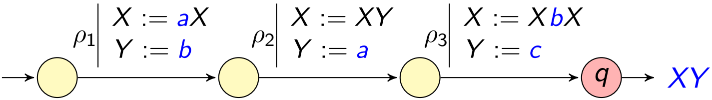
Semantics of SST: two views



Forward interpretation: register valuations

X:	ε	a	ab
Y:	ε	b	a

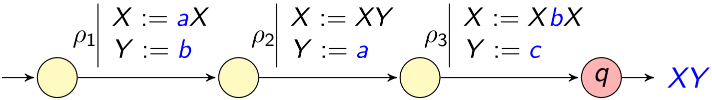
Semantics of SST: two views



Forward interpretation: register valuations

X:	ε	a	ab	abbab
Y:	ε	b	a	c

Semantics of SST: two views

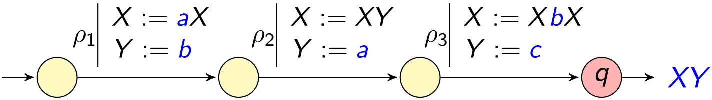


Forward interpretation: register valuations

X:	ε	a	ab	abbab
Y:	ε	b	a	c

➜ the final output is abbabc

Semantics of SST: two views



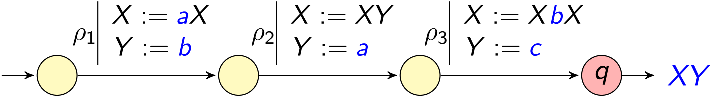
Forward interpretation: register valuations

X:	ε	a	ab	abbab
Y:	ε	b	a	c

➜ the final output is abbabc

Backward interpretation: word over registers

Semantics of SST: two views



Forward interpretation: register valuations

X:	ε	a	ab	abbab
Y:	ε	b	a	c

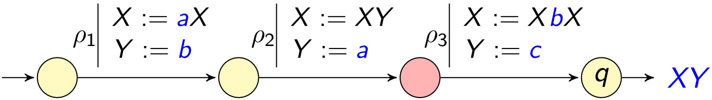
➜ the final output is abbabc

Backward interpretation: word over registers

XY

$\rho_f(q)$

Semantics of SST: two views



Forward interpretation: register valuations

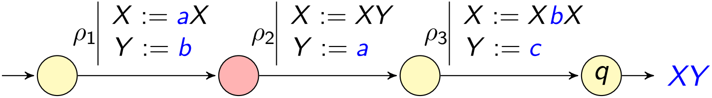
X:	ε	a	ab	abbab
Y:	ε	b	a	c

➜ the final output is abbabc

Backward interpretation: word over registers

XbXc	XY
$\rho_3(\rho_f(q))$	

Semantics of SST: two views



Forward interpretation: register valuations

X:	ε	a	ab	abbab
Y:	ε	b	a	c

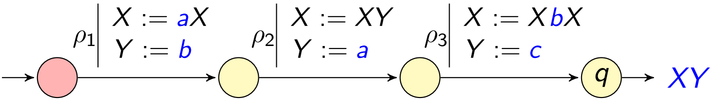
➜ the final output is abbabc

Backward interpretation: word over registers

XYbXYc XbXc XY

$\rho_2(\rho_3(\rho_f(q)))$

Semantics of SST: two views



Forward interpretation: register valuations

X:	ε	a	ab	abbab
Y:	ε	b	a	c

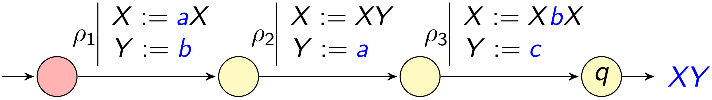
➜ the final output is abbabc

Backward interpretation: word over registers

aXbbaXbc   XYbXYc   XbXc   XY

$\rho_1(\rho_2(\rho_3(\rho_f(q))))$

Semantics of SST: two views



Forward interpretation: register valuations

X:	ε	a	ab	abbab
Y:	ε	b	a	c

➜ the final output is abbabc

Backward interpretation: word over registers

abbabc   aXbbaXbc   XYbXYc   XbXc   XY

$\rho_{ini}(\rho_1(\rho_2(\rho_3(\rho_f(q)))))$

➜ composition of homomorphisms

Copyless SST

"Regular" word-to-word functions:

- copyless SST (and also 1-bounded SST, k-bounded SST)
- deterministic two-way transducers
- MSO-definable word transducers

➜ most simple and intuitive model

Copyless SST

"Regular" word-to-word functions:

- copyless SST (and also 1-bounded SST, k-bounded SST)
- deterministic two-way transducers
- MSO-definable word transducers

➜ most simple and intuitive model

Positive results:

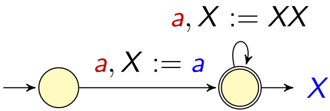
- equivalence of copyless SST is in PSPACE
- functionality of non-deterministic SST is in PSPACE
- type checking is in PSPACE (Given A, B and T, does $T(A) \subseteq B$?)
- closed under composition

What about copyful SST?

Non-linear updates

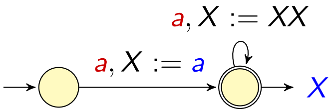
➜ not anymore linear-size increase

➜ strictly increases the expressive power



What about copyful SST?

Non-linear updates
➜ not anymore linear-size increase
➜ strictly increases the expressive power



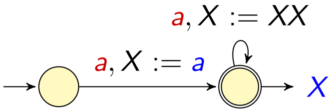
A second example:

$u\#v \mapsto v[a \leftarrow u]$

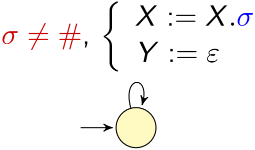
(quadratic blow-up)

What about copyful SST?

Non-linear updates
➜ not anymore linear-size increase
➜ strictly increases the expressive power



A second example:

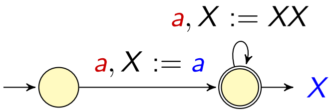


$u\#v \mapsto v[a \leftarrow u]$

(quadratic blow-up)

What about copyful SST?

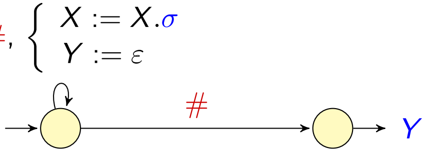
Non-linear updates
➜ not anymore linear-size increase
➜ strictly increases the expressive power



A second example:

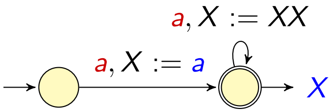
$u\#v \mapsto v[a \leftarrow u]$

(quadratic blow-up)



What about copyful SST?

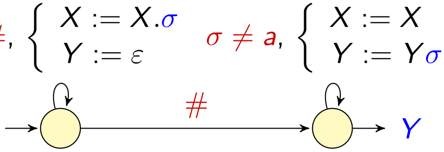
Non-linear updates
➜ not anymore linear-size increase
➜ strictly increases the expressive power



A second example:

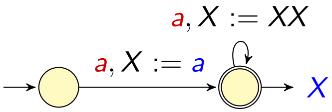
$u\#v \mapsto v[a \leftarrow u]$

(quadratic blow-up)



What about copyful SST?

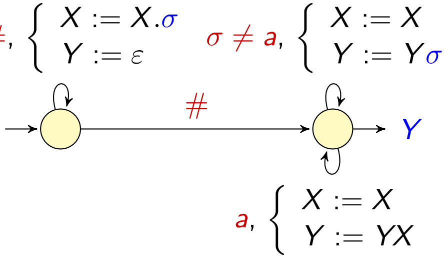
Non-linear updates
➜ not anymore linear-size increase
➜ strictly increases the expressive power



A second example:

$u\#v \mapsto v[a \leftarrow u]$

(quadratic blow-up)



Our results for copyful SST

We obtain decidability results for copyful SST:

- equivalence
- functionality
- copyless-definability: given an SST, does there exist an equivalent copyless one?

Note: decidability of SST equivalence is also a consequence of:

- decidability of the equivalence of top-down tree-to-string transducers [STOC'15]
- recent result on polynomial automata [LICS'17]

Overview

HDT0L systems

An HDT0L system is defined by:

- three alphabets Σ (input), A (working) and Γ (output)
- an initial word $v \in A^*$
- for each $\sigma \in \Sigma$, a orphism $h_\sigma : A^* \to A^*$
- a final morphism $h : A^* \to \Gamma^*$

Behaviour:

$v \qquad \in A^*$

HDT0L systems

An HDT0L system is defined by:

- three alphabets Σ (input), A (working) and Γ (output)
- an initial word $v \in A^*$
- for each $\sigma \in \Sigma$, a orphism $h_\sigma : A^* \to A^*$
- a final morphism $h : A^* \to \Gamma^*$

Behaviour:

$$\sigma_1 \dots \sigma_k \in \Sigma^* \qquad h_{\sigma_1} \dots h_{\sigma_k}(v) \qquad \in A^*$$

HDT0L systems

An HDT0L system is defined by:

- three alphabets Σ (input), A (working) and Γ (output)
- an initial word $v \in A^*$
- for each $\sigma \in \Sigma$, a orphism $h_\sigma : A^* \to A^*$
- a final morphism $h : A^* \to \Gamma^*$

Behaviour:

$$[\![H]\!] : \quad \sigma_1 \dots \sigma_k \in \Sigma^* \quad \mapsto \quad h h_{\sigma_1} \dots h_{\sigma_k}(v) \quad \in \Gamma^*$$

HDT0L systems

An HDT0L system is defined by:

- three alphabets Σ (input), A (working) and Γ (output)
- an initial word $v \in A^*$
- for each $\sigma \in \Sigma$, a orphism $h_\sigma : A^* \to A^*$
- a final morphism $h : A^* \to \Gamma^*$

Behaviour:

$$[\![H]\!] : \qquad \sigma_1 \dots \sigma_k \in \Sigma^* \quad \mapsto \quad hh_{\sigma_1} \dots h_{\sigma_k}(v) \quad \in \Gamma^*$$

Theorem (Culik II & Karhumaki'86, Ruhonen'86, Honkala'00)

Given two HDT0L systems H_1 and H_2, one can decide whether $[\![H_1]\!](w) = [\![H_2]\!](w)$ for every $w \in \Sigma^*$.

HDT0L systems: an example

$\Sigma = \Gamma = \{a, b\}$
$A = \Sigma \cup \{\$_1, \$_2\}$
$v = \$_1\$_2$

h_a : leaves Σ unchanged, $\$_1 \mapsto \$_1 a$, $\$_2 \mapsto a\$_2$
h_b : similar to h_a
h: leaves Σ unchanged, erases $\$_1, \$_2$

HDT0L systems: an example

$\Sigma = \Gamma = \{a, b\}$
$A = \Sigma \cup \{\$_1, \$_2\}$
$v = \$_1\$_2$

h_a : leaves Σ unchanged, $\$_1 \mapsto \$_1 a$, $\$_2 \mapsto a\$_2$
h_b : similar to h_a
h: leaves Σ unchanged, erases $\$_1, \$_2$

Computation of $[\![H_0]\!](baa)$

$$\$_1\$_2$$

HDT0L systems: an example

$\Sigma = \Gamma = \{a, b\}$
$A = \Sigma \cup \{\$_1, \$_2\}$
$v = \$_1\$_2$

h_a : leaves Σ unchanged, $\$_1 \mapsto \$_1 a$, $\$_2 \mapsto a\$_2$
h_b : similar to h_a
h: leaves Σ unchanged, erases $\$_1, \$_2$

Computation of $[\![H_0]\!](baa)$

$$\$_1\$_2$$
$$\$_1 aa\$_2$$

HDT0L systems: an example

$\Sigma = \Gamma = \{a, b\}$
$A = \Sigma \cup \{\$_1, \$_2\}$
$v = \$_1\$_2$

h_a : leaves Σ unchanged, $\$_1 \mapsto \$_1 a$, $\$_2 \mapsto a\$_2$
h_b : similar to h_a
h: leaves Σ unchanged, erases $\$_1, \$_2$

Computation of $[\![H_0]\!](baa)$

$$\$_1\$_2$$
$$\$_1 aa\$_2$$
$$\$_1 aaaa\$_2$$

HDT0L systems: an example

$\Sigma = \Gamma = \{a, b\}$
$A = \Sigma \cup \{\$_1, \$_2\}$
$v = \$_1\$_2$

h_a : leaves Σ unchanged, $\$_1 \mapsto \$_1 a$, $\$_2 \mapsto a\$_2$
h_b : similar to h_a
h: leaves Σ unchanged, erases $\$_1, \$_2$

Computation of $[\![H_0]\!](baa)$

$$\$_1\$_2$$
$$\$_1 aa\$_2$$
$$\$_1 aaaa\$_2$$
$$\$_1 baaaab\$_2$$

HDT0L systems: an example

$\Sigma = \Gamma = \{a, b\}$ $\quad$ h_a : leaves Σ unchanged, $\$_1 \mapsto \$_1 a$, $\$_2 \mapsto a\$_2$

$A = \Sigma \cup \{\$_1, \$_2\}$ $\quad$ h_b : similar to h_a

$v = \$_1\$_2$ $\quad$ h: leaves Σ unchanged, erases $\$_1, \$_2$

Computation of $[\![H_0]\!](baa)$:

$$\$_1\$_2$$
$$\$_1 aa\$_2$$
$$\$_1 aaaa\$_2$$
$$\$_1 baaaab\$_2$$
$$baaaab$$

HDT0L systems: an example

$\Sigma = \Gamma = \{a, b\}$
$A = \Sigma \cup \{\$_1, \$_2\}$
$v = \$_1\$_2$

h_a : leaves Σ unchanged, $\$_1 \mapsto \$_1 a$, $\$_2 \mapsto a\$_2$
h_b : similar to h_a
h: leaves Σ unchanged, erases $\$_1, \$_2$

Computation of $[\![H_0]\!](baa)$:

$$\$_1\$_2$$
$$\$_1 aa\$_2$$
$$\$_1 aaaa\$_2$$
$$\$_1 baaaab\$_2$$
$$baaaab$$

Transformation $w \mapsto w \,.\, mirror(w)$

HDT0L and SST

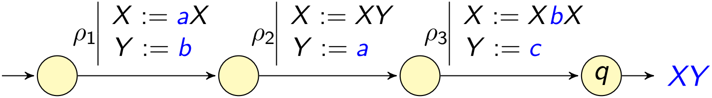
Theorem

HDT0L systems $\equiv$ total copyful SST.

Constructions are effective in both directions, in linear-time.

HDT0L and SST

HDT0L $\rightsquigarrow$ total SST:



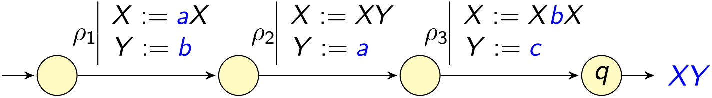
Backward interpretation: word over registers

abbabc aXbbaXbc XYbXYc XbXc XY

$\rho_{ini}(\rho_1(\rho_2(\rho_3(\rho_f(q)))))$

HDT0L and SST

HDT0L $\rightsquigarrow$ total SST:

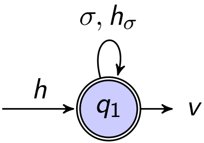


Backward interpretation: word over registers

abbabc aXbbaXbc XYbXYc XbXc XY

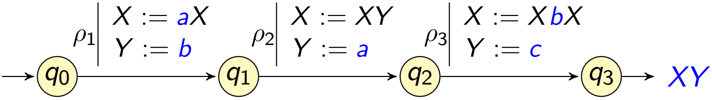
$\rho_{ini}(\rho_1(\rho_2(\rho_3(\rho_f(q)))))$

➜ HDT0L are single-state SST!



HDT0L and SST (2)

total SST $\rightsquigarrow$ HDT0L:



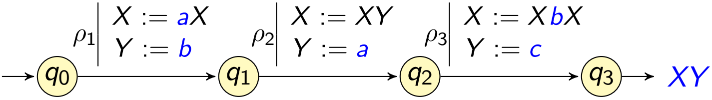
Backward interpretation: word over registers

abbabc   aXbbaXbc   XYbXYc   XbXc   XY

$\rho_{ini}(\rho_1(\rho_2(\rho_3(\rho_f(q_3)))))$

HDT0L and SST (2)

total SST $\rightsquigarrow$ HDT0L:



Backward interpretation: word over registers

abbabc aXbbaXbc XYbXYc $X_{q_2} b_{q_2} X_{q_2} c_{q_2}$ $X_{q_3} Y_{q_3}$

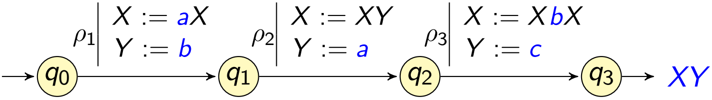
$\rho_{ini}(\rho_1(\rho_2(\rho_3(\rho_f(q_3)))))$

Problem1: states

➜ annotate registers and elements of Σ with Q

HDT0L and SST (2)

total SST $\rightsquigarrow$ HDT0L:



Backward interpretation: word over registers

abbabc aXbbaXbc XYbXYc $X_{q_2} b_{q_2} X_{q_2} c_{q_2}$ $X_{q_3} Y_{q_3}$

$\rho_{ini}(\rho_1(\rho_2(\rho_3(\rho_f(q_3)))))$

Problem1: states

➜ annotate registers and elements of Σ with Q

Problem2: SST are not co-deterministic ($q_1 \xrightarrow{a,\rho_1} q$ and $q_2 \xrightarrow{a,\rho_2} q$)

➜ apply both updates!

$\forall w, \forall q$, there is **exactly one** accepting run on w from q

Final morphism h erases useless computations

Consequences

Theorem

Equivalence of copyful SST is decidable, with same complexity as equivalence of HDT0L systems.

Consequences

Theorem

Equivalence of copyful SST is decidable, with same complexity as equivalence of HDT0L systems.

For non-deterministic SST, we can study the functionality problem: Is it the case that for any two runs on the same input word, the output words are equal?

Using a squaring construction, we reduce it to equivalence:

Theorem

Functionality of non-deterministic copyful SST is decidable.

Overview

Flow transition monoid of SST

Variable flow

$p \xrightarrow{a,\rho} q$ with $\begin{cases} \rho(X) = aYb \\ \rho(Y) = YY \end{cases}$

- Y flows 1 time into X
- Y flows 2 times into Y

Notation:

$$(p, Y) \xrightarrow{a|1} (q, X) \qquad (p, Y) \xrightarrow{a|2} (q, Y)$$

Flow transition monoid of SST

Variable flow

$p \xrightarrow{a,\rho} q$ with $\left\{ \begin{array}{l} \rho(X) = aYb \\ \rho(Y) = YY \end{array} \right.$

- Y flows 1 time into X
- Y flows 2 times into Y

Notation:

$$(p, Y) \xrightarrow{a|1} (q, X) \qquad (p, Y) \xrightarrow{a|2} (q, Y)$$

Matrices with indices (p, X) and coeff. in $\mathbb{N} \cup \{\perp\}$ ($\perp$ is absorbent)

$$M_a[(p, X), (q, Y)] = \left\{ \begin{array}{ll} |\rho(Y)|_X & \text{where } p \xrightarrow{a,\rho} q \\ \perp & \text{if no transition} \end{array} \right.$$

Flow transition monoid of SST

Variable flow

$p \xrightarrow{a,\rho} q$ with $\begin{cases} \rho(X) = aYb \\ \rho(Y) = YY \end{cases}$

- Y flows 1 time into X
- Y flows 2 times into Y

Notation:

$$(p, Y) \xrightarrow{a|1} (q, X) \qquad (p, Y) \xrightarrow{a|2} (q, Y)$$

Matrices with indices (p, X) and coeff. in $\mathbb{N} \cup \{\perp\}$ ($\perp$ is absorbent)

$$M_a[(p, X), (q, Y)] = \begin{cases} |\rho(Y)|_X & \text{where } p \xrightarrow{a,\rho} q \\ \perp & \text{if no transition} \end{cases}$$

- A SST is said *k-bounded copy* iff $\forall u \in \Sigma^*, M_u[\cdot, \cdot] \leq k$

Flow transition monoid of SST

Variable flow

$p \xrightarrow{a,\rho} q$ with $\begin{cases} \rho(X) = aYb \\ \rho(Y) = YY \end{cases}$

- Y flows 1 time into X
- Y flows 2 times into Y

Notation:

$$(p, Y) \xrightarrow{a|1} (q, X) \qquad (p, Y) \xrightarrow{a|2} (q, Y)$$

Matrices with indices (p, X) and coeff. in $\mathbb{N} \cup \{\perp\}$ ($\perp$ is absorbent)

$$M_a[(p, X), (q, Y)] = \begin{cases} |\rho(Y)|_X & \text{where } p \xrightarrow{a,\rho} q \\ \perp & \text{if no transition} \end{cases}$$

- A SST is said k-bounded copy iff $\forall u \in \Sigma^*, M_u[\cdot, \cdot] \leq k$
- A SST is said bounded copy iff it is k-bounded copy for some k

Flow transition monoid of SST

Variable flow

$p \xrightarrow{a,\rho} q$ with $\begin{cases} \rho(X) = aYb \\ \rho(Y) = YY \end{cases}$

- Y flows 1 time into X
- Y flows 2 times into Y

Notation:

$$(p, Y) \xrightarrow{a|1} (q, X) \qquad (p, Y) \xrightarrow{a|2} (q, Y)$$

Matrices with indices (p, X) and coeff. in $\mathbb{N} \cup \{\perp\}$ ($\perp$ is absorbent)

$$M_a[(p, X), (q, Y)] = \begin{cases} |\rho(Y)|_X & \text{where } p \xrightarrow{a,\rho} q \\ \perp & \text{if no transition} \end{cases}$$

- A SST is said *k-bounded copy* iff $\forall u \in \Sigma^*, M_u[\cdot, \cdot] \leq k$
- A SST is said *bounded copy* iff it is k-bounded copy for some k

Theorem

Bounded copy SST $\equiv$ Copyless SST

From copyful to copyless SST

(p, X) is live iff $\exists u \in \Sigma^* \mid p \xrightarrow{u,\rho} p_f$ and X appears in $\rho(\rho_f(p_f))$

From copyful to copyless SST

(p, X) is live iff $\exists u \in \Sigma^* \mid p \xrightarrow{u,\rho} p_f$ and X appears in $\rho(\rho_f(p_f))$

(p, X) is unbounded iff $\{\nu(X) \in \Gamma^* \mid q_0 \to (p, \nu)\}$ is infinite

From copyful to copyless SST

(p, X) is live iff $\exists u \in \Sigma^* \mid p \xrightarrow{u,\rho} p_f$ and X appears in $\rho(\rho_f(p_f))$

(p, X) is unbounded iff $\{\nu(X) \in \Gamma^* \mid q_0 \to (p, \nu)\}$ is infinite

Proposition

$[\![T]\!]$ is copyless definable iff there exists $K \in \mathbb{N}$ s.t. for all $(p, X) \xrightarrow{n} (q, Y)$ with (p, X),(q, Y) live and unbounded, we have $n \leq K$.

- live and unbounded pairs can be computed in PTime
- adapt the construction of matrices

➜ boundedness of products of matrices: decidable in PTime

From copyful to copyless SST

(p, X) is live iff $\exists u \in \Sigma^* \mid p \xrightarrow{u,\rho} p_f$ and X appears in $\rho(\rho_f(p_f))$

(p, X) is unbounded iff $\{\nu(X) \in \Gamma^* \mid q_0 \to (p, \nu)\}$ is infinite

Proposition

$[\![T]\!]$ is copyless definable iff there exists $K \in \mathbb{N}$ s.t. for all $(p, X) \xrightarrow{n} (q, Y)$ with (p, X),(q, Y) live and unbounded, we have $n \leq K$.

- live and unbounded pairs can be computed in PTime
- adapt the construction of matrices

➜ boundedness of products of matrices: decidable in PTime

Theorem

Given a copyful SST, one can decide in PTime whether there exists an equivalent copyless SST.

Overview

Summary and Perspectives

- strong link between copyful SST and HDT0L systems:
- positive results for copyful SST (equivalence, functionality)
- decision of copyless among copyful SST

Summary and Perspectives

- strong link between copyful SST and HDT0L systems:
- positive results for copyful SST (equivalence, functionality)
- decision of copyless among copyful SST

- copyful SST deserve to be studied
- transfer results between copyful SST and HDT0L systems
- Hilbert's basis theorem (Adrien's talk)

Summary and Perspectives

- strong link between copyful SST and HDT0L systems:
- positive results for copyful SST (equivalence, functionality)
- decision of copyless among copyful SST

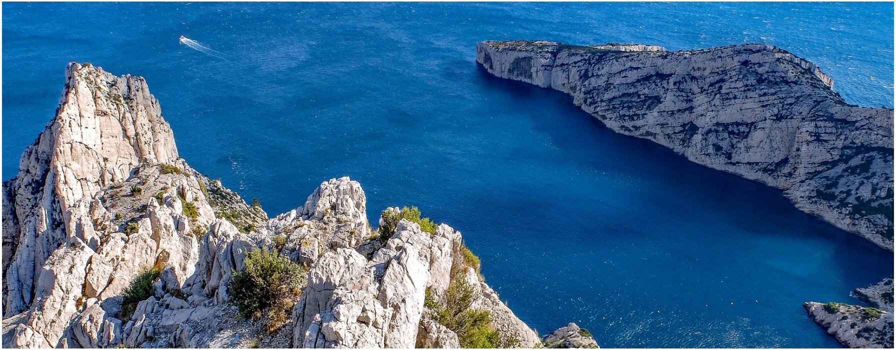
- copyful SST deserve to be studied
- transfer results between copyful SST and HDT0L systems
- Hilbert's basis theorem (Adrien's talk)

Thanks!

RP 2018 Announcement

12th International Conference on Reachability Problems

24 - 26 SEPTEMBER 2018, MARSEILLE, FRANCE



HOME

INVITED SPEAKERS

SUBMISSION

COMMITTEES

ABOUT

The Laboratory of Computer Science and Systems at Aix-Marseille University organizes the 12th International Conference on Reachability Problems (RP'18). This event takes place on the campus of Saint-Charles, close to the train station and to the old harbour of Marseille.

Papers presenting original contributions related to reachability problems in different computational models and systems are being sought.

IMPORTANT DATES

Regular Papers

Abstract: 31 May 2018
Submission: 7 June 2018
Notification: 11 July 2018
Final Version: 17 July 2018