Regular transformations of data words

through origin information

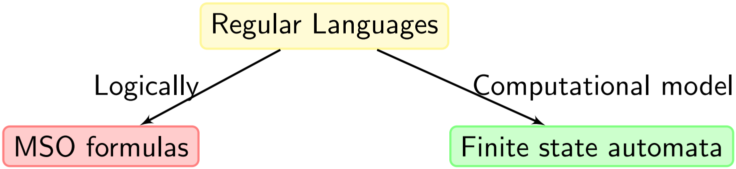
Antoine Durand-Gasselin and Peter Habermehl

LIF, Aix-Marseille Université & IRIF, Univ Paris Diderot

12-04-2016

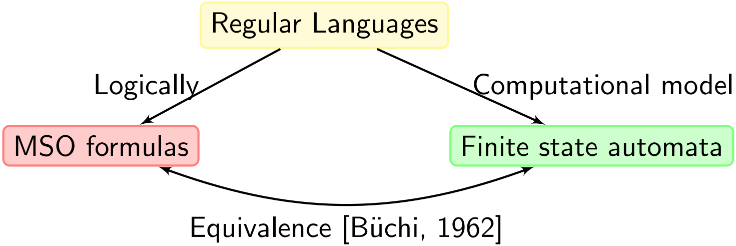
Formal Languages: Qualitative properties on words

Qualitative properties over words
(over finite alphabet)
$\varphi : \Sigma^* \to \{0, 1\}$



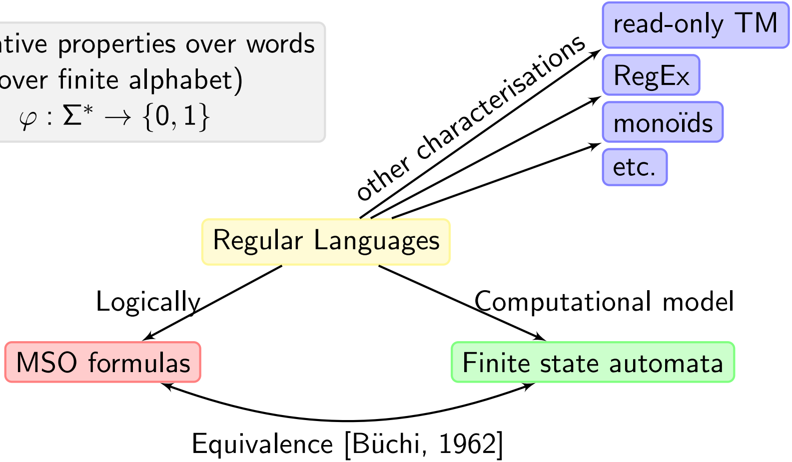
Formal Languages: Qualitative properties on words

Qualitative properties over words
(over finite alphabet)
$\varphi : \Sigma^* \to \{0, 1\}$



Formal Languages: Qualitative properties on words

Qualitative properties over words
(over finite alphabet)
$\varphi : \Sigma^* \to \{0,1\}$



Word Transformations

Words Transformations
$\varphi : \Sigma^* \to \Sigma^*$

Regular Word transformations

Logically

Computational models

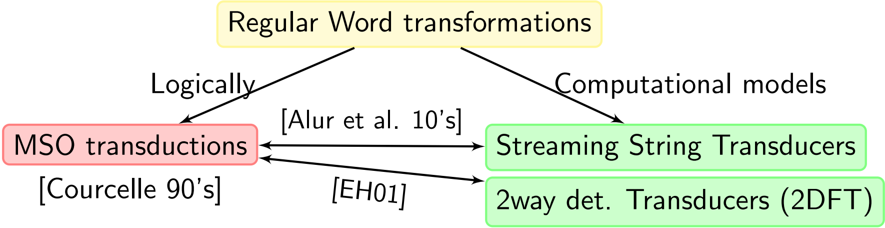
MSO transductions

Streaming String Transducers

2way det. Transducers (2DFT)

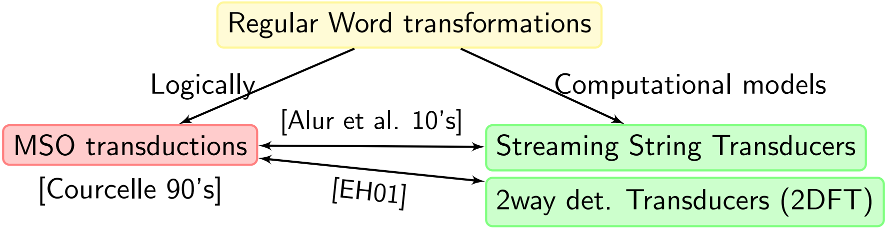
Word Transformations

Words Transformations
$\varphi : \Sigma^* \to \Sigma^*$



Word Transformations

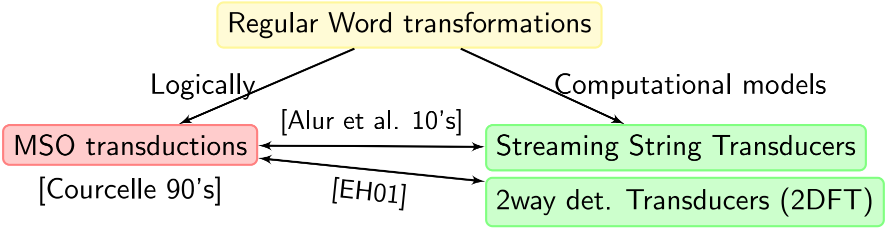
Words Transformations
$\varphi : \Sigma^* \to \Sigma^*$



Key fact: equivalence between a logical definition and
two **deterministic** computational models, one way and two way

Word Transformations

Words Transformations
$\varphi : \Sigma^* \to \Sigma^*$



Key fact: equivalence between a logical definition and
two **deterministic** computational models, one way and two way

Our contribution

An extension of this picture to the setting of data words.

Contribution

Data word transformations
$\varphi : (\Sigma \times \Delta)^* \to (\Sigma \times \Delta)^*$

Contribution

Data word transformations
$\varphi : (\Sigma \times \Delta)^* \to (\Sigma \times \Delta)^*$

Logical definition

Contribution

Data word transformations
$\varphi : (\Sigma \times \Delta)^* \to (\Sigma \times \Delta)^*$

Logical definition

Two-way machine

Contribution

Data word transformations
$\varphi : (\Sigma \times \Delta)^* \rightarrow (\Sigma \times \Delta)^*$

Logical definition

Two-way machine

Contribution

Data word transformations
$\varphi : (\Sigma \times \Delta)^* \to (\Sigma \times \Delta)^*$

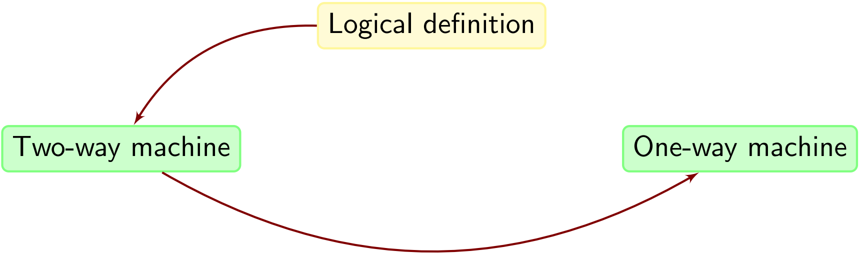
Logical definition

Two-way machine

One-way machine

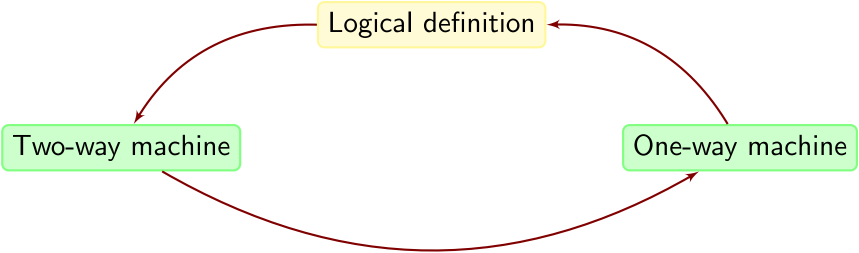
Contribution

Data word transformations
$\varphi : (\Sigma \times \Delta)^* \to (\Sigma \times \Delta)^*$

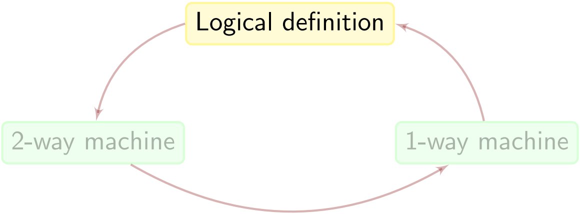


Contribution

Data word transformations
$\varphi : (\Sigma \times \Delta)^* \to (\Sigma \times \Delta)^*$



Contents



Definition of finite words transformations

MSO transductions

- Definition by Courcelle
- Words can be seen as (node-labeled) graphs
- MSO graph transductions
  - A graph is an interpreted structure
  - MSO interpretation of such structures
  - Introduction of a fixed finite number of copies of the "input" structure
- Restriction to words

Running example

input: a b c b b $\#$ a a b $\#$ c c a a $\#$ b

Definition of an MSO transduction

- input and output alphabets $\Sigma = \Gamma = \{a, b, \#\}$

Running example

input:	a	b	c	b	b	$\#$	a	a	b	$\#$	c	c	a	a	$\#$	b
copy 1:	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
copy 2:	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o

Definition of an MSO transduction

- input and output alphabets $\Sigma = \Gamma = \{a, b, \#\}$
- finite set C of copies (here $C = \{1, 2\}$)
- formula φ_{indom}

Running example

input:	a	b	c	b	b	$\#$	a	a	b	$\#$	c	c	a	a	$\#$	b
copy 1:	o	o	o	o	o		o	o	o		o	o	o	o		
copy 2:	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o

Definition of an MSO transduction

- input and output alphabets $\Sigma = \Gamma = \{a, b, \#\}$
- finite set C of copies (here $C = \{1, 2\}$)
- formula φ_{indom}
- formulas $(\varphi^c_{dom})_{c \in C}$

Running example

input:	a	b	c	b	b	$\#$	a	a	b	$\#$	c	c	a	a	$\#$	b
copy 1:	a	b	c	b	b		a	a	b		c	c	a	a		
copy 2:	a	b	c	b	b	$\#$	a	a	b	$\#$	c	c	a	a	$\#$	b

Definition of an MSO transduction

- input and output alphabets $\Sigma = \Gamma = \{a, b, \#\}$
- finite set C of copies (here $C = \{1, 2\}$)
- formula φ_{indom}
- formulas $(\varphi^c_{dom})_{c \in C}$
- formulas $(\varphi^c_{\alpha})_{\gamma \in \Gamma}$

Running example

input: $a \quad b \quad c \quad b \quad b \quad \# \quad a \quad a \quad b \quad \# \quad c \quad c \quad a \quad a \quad \# \quad b$

copy 1: $a \leftarrow b \leftarrow c \leftarrow b \leftarrow b \qquad a \leftarrow a \leftarrow b \qquad c \leftarrow c \leftarrow a \leftarrow a$

copy 2: $a \rightarrow b \rightarrow c \rightarrow b \rightarrow b \rightarrow \# \quad a \rightarrow a \rightarrow b \rightarrow \# \quad c \rightarrow c \rightarrow a \rightarrow a \rightarrow \# \rightarrow b$

Definition of an MSO transduction

- input and output alphabets $\Sigma = \Gamma = \{a, b, \#\}$
- finite set C of copies (here $C = \{1, 2\}$)
- formula φ_{indom}
- formulas $(\varphi^{c}_{dom})_{c \in C}$
- formulas $(\varphi^{c}_{\alpha})_{\gamma \in \Gamma}$
- formulas $(\varphi^{c,c'}_{<})_{c,c' \in C}$

Properties [Courcelle 90's]

- Output linearly larger
- Regular input domain
- Any MSO formula over v can be translated to an MSO formula over u
- MSO Typechecking
- Functional composition
- Functional equivalence is decidable

Data words

Definition

- A data word is a word over alphabet $\Sigma \times \Delta$ (Σ finite, Δ infinite)
- We see a data word as a finite word and a mapping from pos. to Δ

Example (cont'd)

input: $\underset{3}{a}\ \underset{10}{b}\ \underset{2}{a}\ \underset{8}{b}\ \underset{2}{b}\ \underset{4}{\#}\ \underset{3}{a}\ \underset{7}{a}\ \underset{3}{b}\ \underset{4}{\#}\ \underset{5}{b}\ \underset{19}{b}\ \underset{17}{a}\ \underset{1}{a}\ \underset{2}{\#}\ \underset{3}{b}$

Data words

Definition

- A data word is a word over alphabet $\Sigma \times \Delta$ (Σ finite, Δ infinite)
- We see a data word as a finite word and a mapping from pos. to Δ

Example (cont'd)

input: $a \quad b \quad a \quad b \quad b \quad \# \quad a \quad a \quad b \quad \# \quad b \quad b \quad a \quad a \quad \# \quad b$

Data words

Definition

- A data word is a word over alphabet $\Sigma \times \Delta$ (Σ finite, Δ infinite)
- We see a data word as a finite word and a mapping from pos. to Δ

Example (cont'd)

input: $a \quad b \quad a \quad b \quad b \quad \# \quad a \quad a \quad b \quad \# \quad b \quad b \quad a \quad a \quad \# \quad b$

copy 1: $a \leftarrow b \leftarrow a \leftarrow b \leftarrow b \qquad a \leftarrow a \leftarrow b \qquad b \leftarrow b \leftarrow a \leftarrow a$

copy 2: $a \rightarrow b \rightarrow a \rightarrow b \rightarrow b \rightarrow \# \quad a \rightarrow a \rightarrow b \rightarrow \# \quad b \rightarrow b \rightarrow a \rightarrow a \rightarrow \# \rightarrow b$

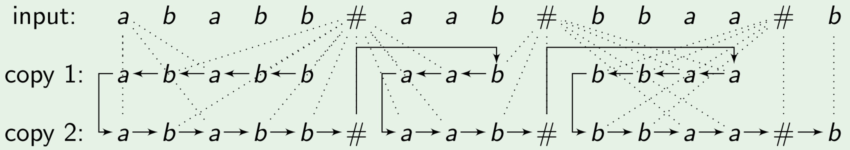
- We give formulas $\varphi^c_{orig}(x, y)$ stating that x in copy c has the same data value as y
- We impose functionality of φ^c_{orig} (can be done in MSO)

Data words

Definition

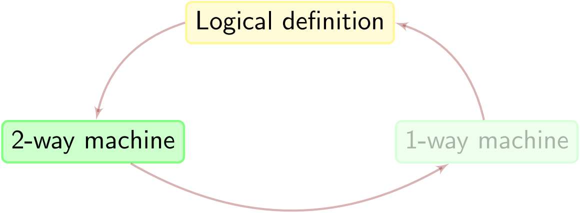
- A data word is a word over alphabet $\Sigma \times \Delta$ (Σ finite, Δ infinite)
- We see a data word as a finite word and a mapping from pos. to Δ

Example (cont'd)



- We give formulas $\varphi^c_{orig}(x, y)$ stating that x in copy c has the same data value as y
- We impose functionality of φ^c_{orig} (can be done in MSO)

Contents



Two-way model

Two way DFT

- Two way deterministic automaton with transitions labeled over Γ^*
- The image is defined if the run is successful as the concatenation of labels of transitions taken along the run

Theorem

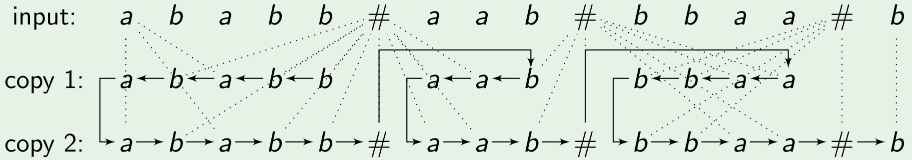
Two way deterministic transducers capture MSO transductions of finite words over finite alphabet

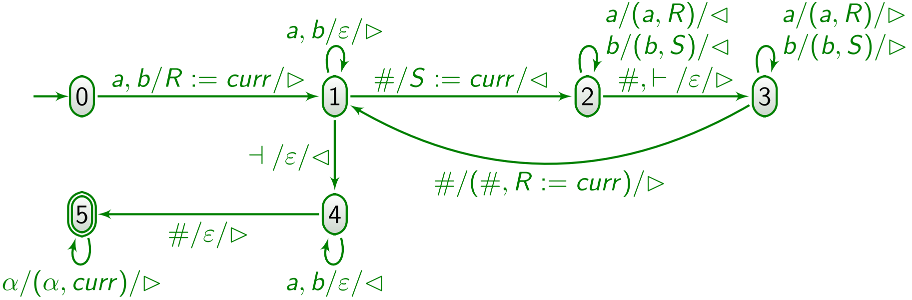
With registers

- We add a set of registers R that store data values
- Their value is updated deterministically from data values and the *current* data value
- Transitions are labeled by words in $(\Gamma \times R)^*$

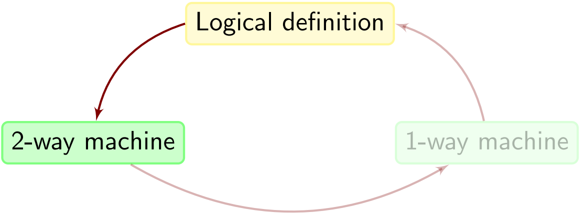
Our example

Example (cont'd)

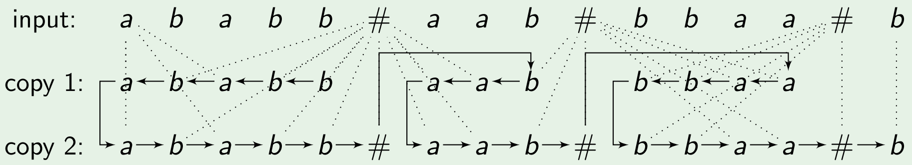




Contents



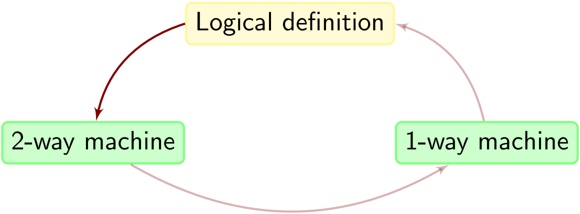
Example (cont'd)



The algorithm

- The two-way model is closed by composition [Chytil & Jákl 1977]
- We relabel the word by adding the information of $(k+3)$-types
  - i.e. the set of MSO formulas of quantifier depth at most $(k+3)$ that are satisfied by the prefix and by the suffix
- We will output symbols at the corresponding position
  - This already handles the fine word part [EH01]
- At any moment the data values that are stored are those which are used left and appear right or vice versa
  - That's only a finite number of data values to store

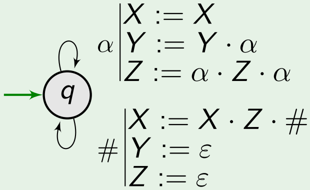
Contents



Streaming string transducers

Finite part

$X = Y = Z = \varepsilon$

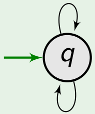


$\mathcal{F}(q) = X \cdot Y$

Streaming string transducers

Finite part

$X{=}Y{=}Z{=}\varepsilon$

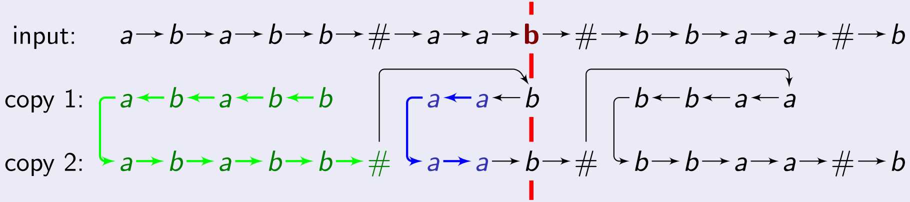


$$\alpha \left| \begin{array}{l} X := X \\ Y := Y \cdot \alpha \\ Z := \alpha \cdot Z \cdot \alpha \end{array} \right.$$

$$\# \left| \begin{array}{l} X := X \cdot Z \cdot \# \\ Y := \varepsilon \\ Z := \varepsilon \end{array} \right.$$

$$\mathcal{F}(q) = X \cdot Y$$

What to store in string variables

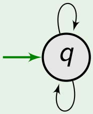


How to handle data values

- We need data registers
- and data parameters

Data registers and data parameters

$X{=}Y{=}Z{=}\varepsilon$



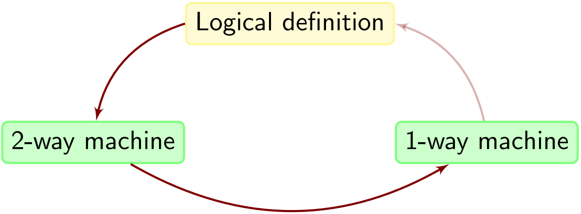
$a \left| \begin{array}{l} X := X \\ Y := Y \cdot (a, r) \\ Z := (a, r) \cdot Z \cdot (a, r) \end{array} \right.$

$b \left| \begin{array}{l} X := X \\ Y := Y \cdot (b, p) \\ Z := (b, p) \cdot Z \cdot (b, p) \end{array} \right.$

$\# \left| \begin{array}{l} X := X \cdot Z[p \leftarrow curr] \cdot (\#, curr) \\ Y := \varepsilon \\ Z := \varepsilon \\ r := current \end{array} \right.$

$\mathcal{F}(q) = X \cdot Y[p \leftarrow curr]$

Contents



From two-way to one-way

A one-way cannot go back !

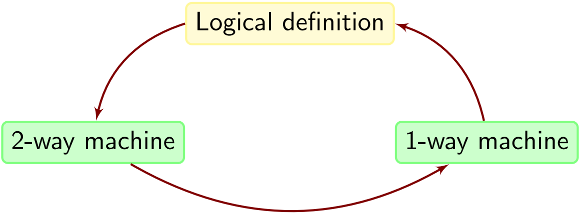
- A valid run of a two way never visits a position more than $|Q|$ times
- In which state does the two-way first reach i ?
- At position i, from state q in which state does the 2way first reach position $i+1$?

How to build the one-way ? (Shepherdson)

At position i

- Easy to keep track of what happened until the 2way first reached i
- String variable X_q will contain what the two-way produces from position i in state q until it first reaches position $i+1$ ...
- ... with content of data registers "being" fresh parameters
- $r_{R,q}$ at position i will contain (if it exists) the last data value stored by the 2way in register R from state q in position i until it reaches position $i+1$

Contents



A semantic restriction for SST's

Restriction on copying

- Automaton + String variables + variables update function
- And the following semantic restriction:
  - The content of some register may not flow more than once in the output

One-way to Logic

The expressive power of MSO allows quite naturally to describe the behaviour of such a finite-state system.

Conclusion

Conclusion

Extension of string transformations to data strings

- Equivalence between 3 models:
  - Logical definition using MSO
  - Extension of 2DFT with data registers
  - Extension of SST with data registers **and** data parameters
- Determistic models
- Typechecking and functionnal equivalence are decidable

Conclusion

Extension of string transformations to data strings

- Equivalence between 3 models:
  - Logical definition using MSO
  - Extension of 2DFT with data registers
  - Extension of SST with data registers **and** data parameters
- Determistic models
- Typechecking and functionnal equivalence are decidable

Future work

- Challenges for extension to testing input data values:
  - Deterministic computational models
  - Equivalence with a logical framework
- Efficient conversions
- Transformations of other classes of objects
- Canonical objects, minimization

Conclusion

Extension of string transformations to data strings

- Equivalence between 3 models:
  - Logical definition using MSO
  - Extension of 2DFT with data registers
  - Extension of SST with data registers **and** data parameters
- Determistic models
- Typechecking and functionnal equivalence are decidable

Future work

- Challenges for extension to testing input data values:
  - Deterministic computational models
  - Equivalence with a logical framework
- Efficient conversions
- Transformations of other classes of objects
- Canonical objects, minimization

Thank you for your attention!

Büchi, J. R. (1962).

On a decision method in restricted second-order arithmetic.

In *Int. Congr. for Logic Methodology and Philosophy of Science*, pages 1–11. Standford University Press, Stanford.