Tree Transformations by means of visibly pushdown transducers

Jean-Marc Talbot
Joint Work L. Dartois [1], E. Filiot [1],
P.-A. Reynier [2]

[1]Université Libre de Bruxelles

[2]Université d'Aix-Marseille

CAALM - Jan 2019

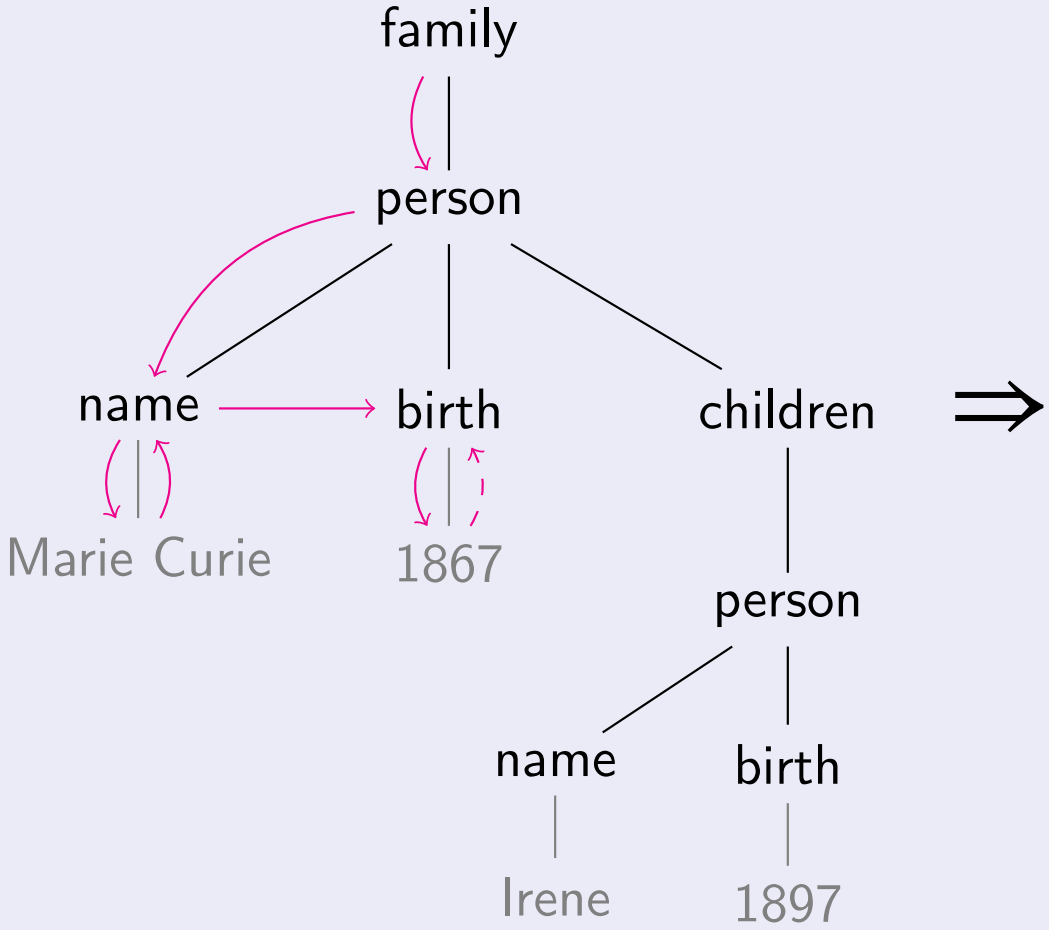
Outline

Outline

Well-nested words for trees : an example

Encoding

Linearizations of (unranked) trees $\subseteq$ Well-nested words



$\Rightarrow$

$< family >$
$< person >$
$< name >$ *MarieCurie* $< \backslash name >$
$< birth >$ 1867 $< \backslash birth >$
$< children >$
$< name >$ *Irene* $< \backslash name >$
$< birth >$ 1897 $< \backslash birth >$
$< \backslash children >$
$< \backslash person >$
$< \backslash family >$

Well-nested words for trees

Definition (Structured Alphabet)

A *structured alphabet* Σ is a set $\Sigma = \Sigma_c \uplus \Sigma_r$ (*call* and *return* symbols resp.).

Definition (Well-nested Words)

A word is *well-nested* if it is generated by the following grammar

$$W \rightarrow \epsilon \mid WW \mid c_i W r_j \qquad c_i \in \Sigma_c, r_j \in \Sigma_r$$

Σ^*_{wn} is the set of all well-nested words.

Well-nested words for trees

Definition (Structured Alphabet)

A *structured alphabet* Σ is a set $\Sigma = \Sigma_c \uplus \Sigma_r$ (*call* and *return* symbols resp.).

Definition (Well-nested Words)

A word is *well-nested* if it is generated by the following grammar

$$W \rightarrow \epsilon \mid WW \mid c_i W r_j \qquad c_i \in \Sigma_c, r_j \in \Sigma_r$$

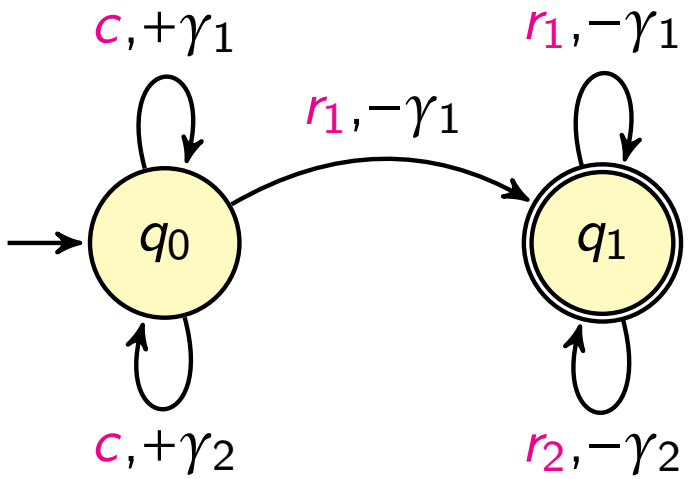
Σ^*_{wn} is the set of all well-nested words.

Induces a matching relation

Visibly Pushdown Automata (VPAs) [Alur,Madhusudan,04]

VPAs = Pushdown Automata

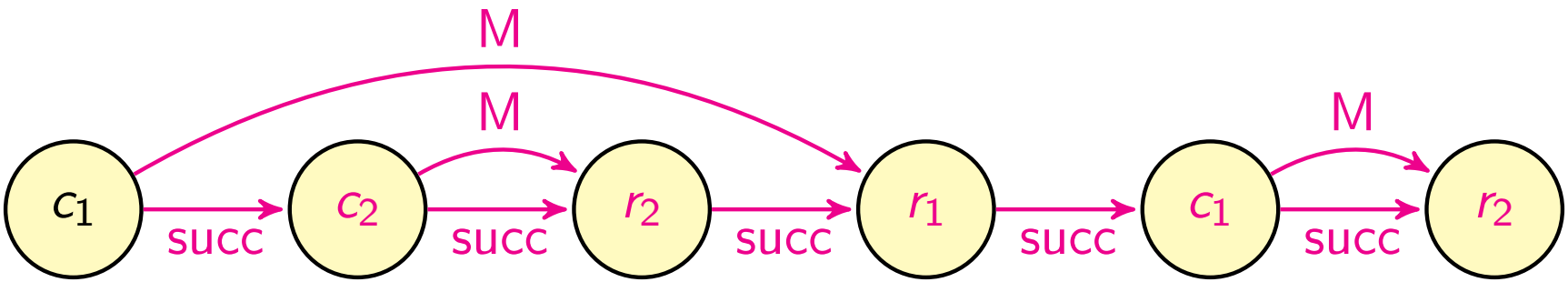
- running on on a *structured* alphabet $\Sigma = \Sigma_c \uplus \Sigma_r$:
- whose behavior is driven by the input
  - push **one** stack symbol on **call** symbols Σ_c
  - pop **one** stack symbol on **return** symbols Σ_r
- accept on empty stack and final state (accepted $\subseteq \Sigma^*_{\text{wn}}$) (in this talk)



$$L(A) = \{c^n \cdot r_1 \cdot (r_1 + r_2)^{n-1} \mid n > 0\}.$$

MSO for nested words

Logical Structure of Nested Words :



$\mathrm{MSO}_{\mathrm{nw}}$ formulas :

$$lab_a(x) \mid succ(x,y) \mid M(x,y) \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$$

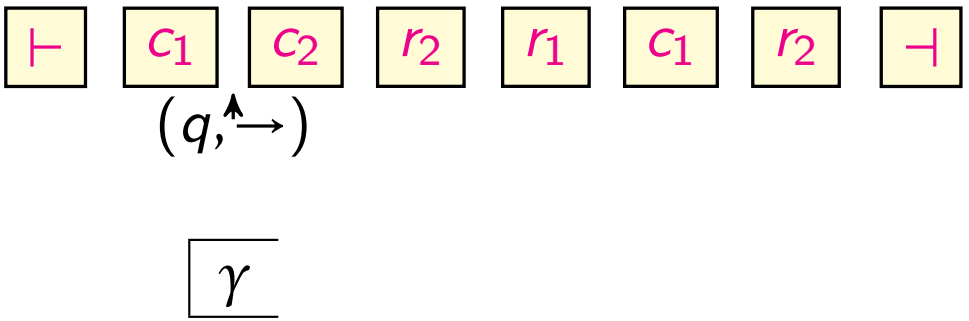
Theorem [Alur,Madhusudan,04]

$\mathrm{MSO}_{\mathrm{nw}}$ definable languages $\equiv$ VPA definable languages.

Two-way VPA (2VPA) [Madhusudan,Viswanathan,09]

Main features :

- directed states $Q \times \{\leftarrow, \rightarrow\}$ ($\rightarrow$: read the symbol on the right, ...)
- Behaviour wrt the stack:
  - Forward ($\rightarrow$): just as VPA
  - Backward ($\leftarrow$): dually, **pop** on call and **push** on return

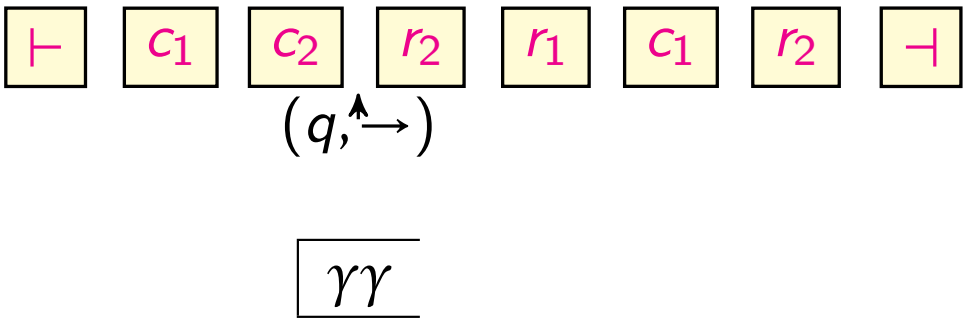


- $(q, \rightarrow) \xrightarrow{c_2, +\gamma} (q, \rightarrow)$
  $(q, \rightarrow) \xrightarrow{r_2, -\gamma} (q', \leftarrow)$
  $(q', \leftarrow) \xrightarrow{r_2, +\gamma''} (q'', \rightarrow)$
  $(q'', \rightarrow) \xrightarrow{r_2, -\gamma''} (q', \rightarrow)$

Two-way VPA (2VPA) [Madhusudan,Viswanathan,09]

Main features :

- directed states $Q \times \{\leftarrow, \rightarrow\}$ ($\rightarrow$: read the symbol on the right, ...)
- Behaviour wrt the stack:
  - Forward ($\rightarrow$): just as VPA
  - Backward ($\leftarrow$): dually, **pop** on call and **push** on return

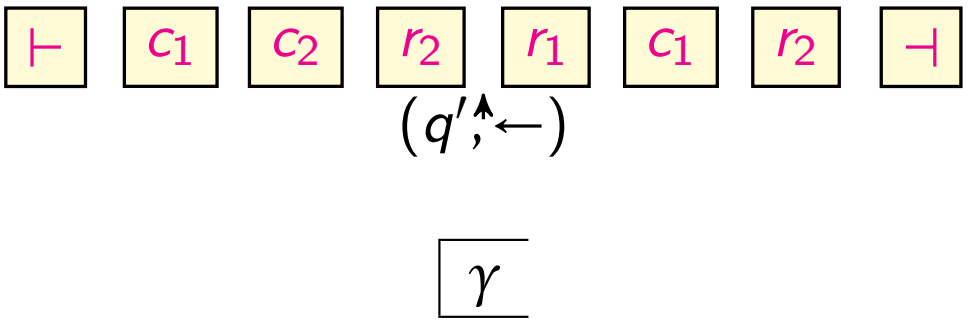


$$(q, \rightarrow) \xrightarrow{c_2, +\gamma} (q, \rightarrow)$$
$$(q, \rightarrow) \xrightarrow{r_2, -\gamma} (q', \leftarrow)$$
$$(q', \leftarrow) \xrightarrow{r_2, +\gamma''} (q'', \rightarrow)$$
$$(q'', \rightarrow) \xrightarrow{r_2, -\gamma''} (q', \rightarrow)$$

Two-way VPA (2VPA) [Madhusudan,Viswanathan,09]

Main features :

- directed states $Q \times \{\leftarrow, \rightarrow\}$ ($\rightarrow$: read the symbol on the right, ...)
- Behaviour wrt the stack:
  - Forward ($\rightarrow$): just as VPA
  - Backward ($\leftarrow$): dually, **pop** on call and **push** on return

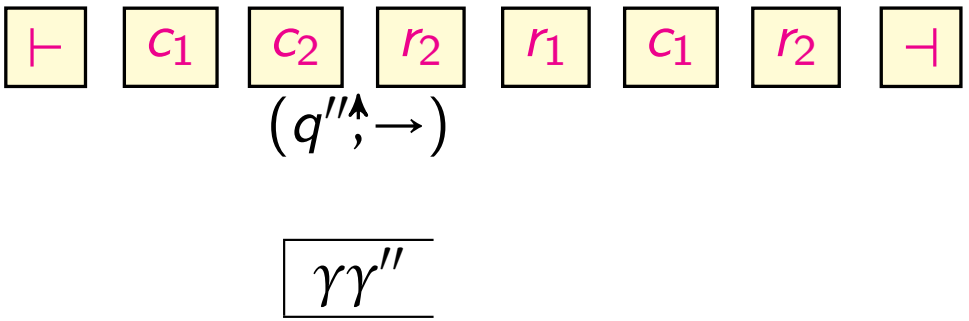


$$(q, \rightarrow) \xrightarrow{c_2, +\gamma} (q, \rightarrow)$$
$$(q, \rightarrow) \xrightarrow{r_2, -\gamma} (q', \leftarrow)$$
$$(q', \leftarrow) \xrightarrow{r_2, +\gamma''} (q'', \rightarrow)$$
$$(q'', \rightarrow) \xrightarrow{r_2, -\gamma''} (q', \rightarrow)$$

Two-way VPA (2VPA) [Madhusudan,Viswanathan,09]

Main features :

- directed states $Q \times \{\leftarrow, \rightarrow\}$ ($\rightarrow$: read the symbol on the right, ...)
- Behaviour wrt the stack:
  - Forward ($\rightarrow$): just as VPA
  - Backward ($\leftarrow$): dually, **pop** on call and **push** on return

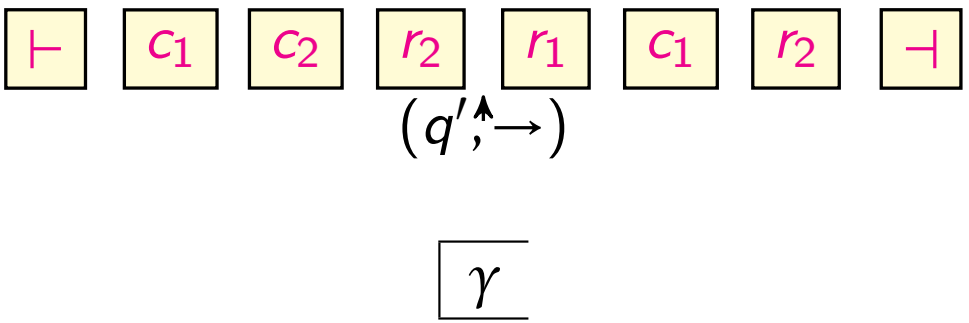


$$(q, \rightarrow) \xrightarrow{c_2, +\gamma} (q, \rightarrow)$$
$$(q, \rightarrow) \xrightarrow{r_2, -\gamma} (q', \leftarrow)$$
$$(q', \leftarrow) \xrightarrow{r_2, +\gamma''} (q'', \rightarrow)$$
$$(q'', \rightarrow) \xrightarrow{r_2, -\gamma''} (q', \rightarrow)$$

Two-way VPA (2VPA) [Madhusudan,Viswanathan,09]

Main features :

- directed states $Q \times \{\leftarrow, \rightarrow\}$ ($\rightarrow$: read the symbol on the right, ...)
- Behaviour wrt the stack:
  - Forward ($\rightarrow$): just as VPA
  - Backward ($\leftarrow$): dually, **pop** on call and **push** on return



$$(q, \rightarrow) \xrightarrow{c_2, +\gamma} (q, \rightarrow)$$
$$(q, \rightarrow) \xrightarrow{r_2, -\gamma} (q', \leftarrow)$$
$$(q', \leftarrow) \xrightarrow{r_2, +\gamma''} (q'', \rightarrow)$$
$$(q'', \rightarrow) \xrightarrow{r_2, -\gamma''} (q', \rightarrow)$$

Theorem

2VPA are as expressive as VPA.

Tree Transformations

Transformation : $\mathscr{T}$, a function from Σ^*_{wn} to Γ^*

Tree Transformations

Transformation : $\mathscr{T}$, a function from Σ^*_{wn} to Γ^*

- a transducer $T = (A, O)$ extends automata A with an output mechanism O mapping transitions of A to words from Γ^*.

Tree Transformations

Transformation : $\mathscr{T}$, a function from Σ^*_{wn} to Γ^*

- a transducer $T = (A, O)$ extends automata A with an output mechanism O mapping transitions of A to words from Γ^*.
- MSO transformations (à la Courcelle) : a set of MSO formulas interpreted on a **fixed** number of copies of the input structure defining the predicates of some output structure.

Tree Transformations

Transformation : $\mathscr{T}$, a function from Σ^*_{wn} to Γ^*

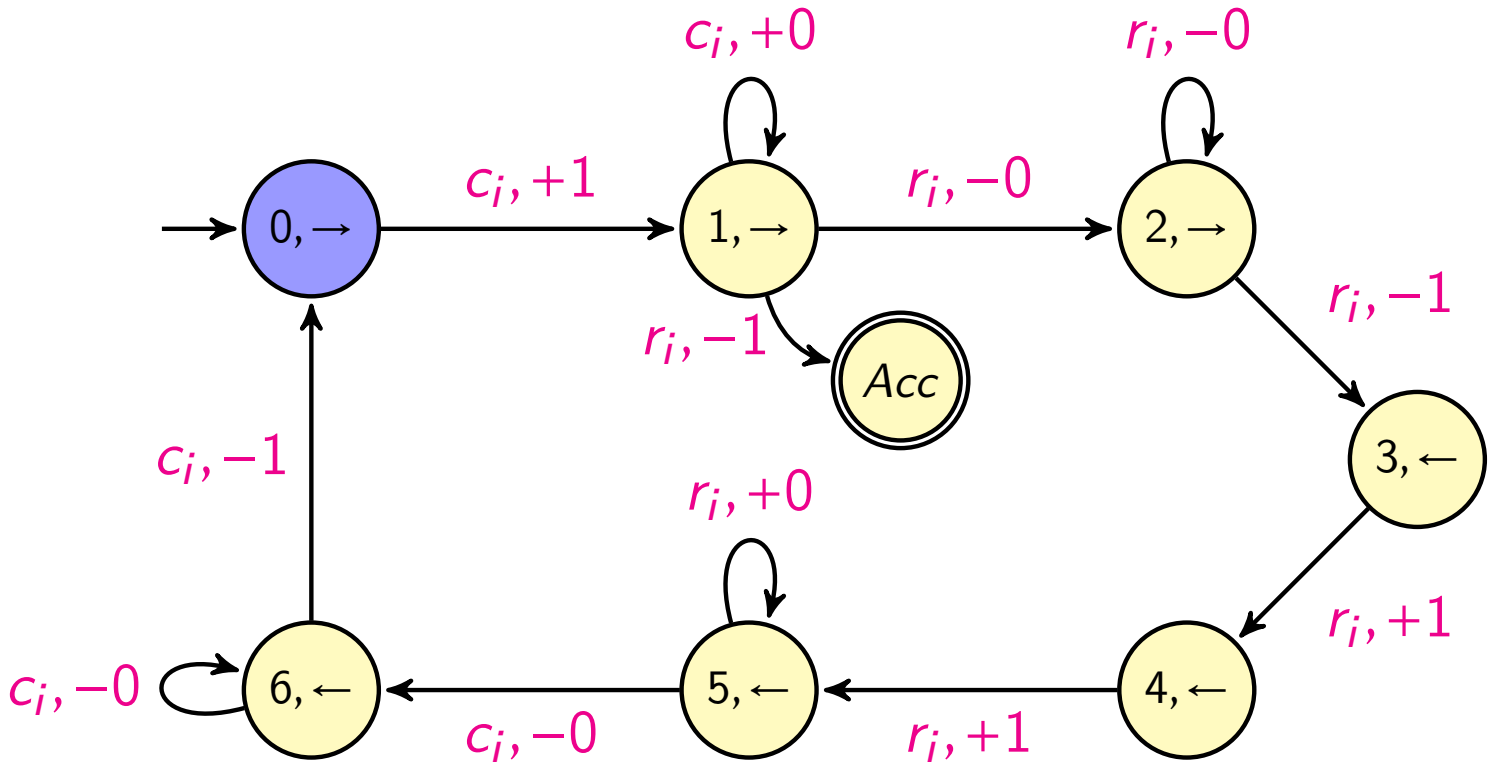
- a transducer $T = (A, O)$ extends automata A with an output mechanism O mapping transitions of A to words from Γ^*.
- MSO transformations (à la Courcelle) : a set of MSO formulas interpreted on a **fixed** number of copies of the input structure defining the predicates of some output structure.

Interesting properties :

- Expressiveness
- Type-checking problem : $[\![T]\!](L_I) \subseteq L_O$
- Closure by composition (when possible)

Outline

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

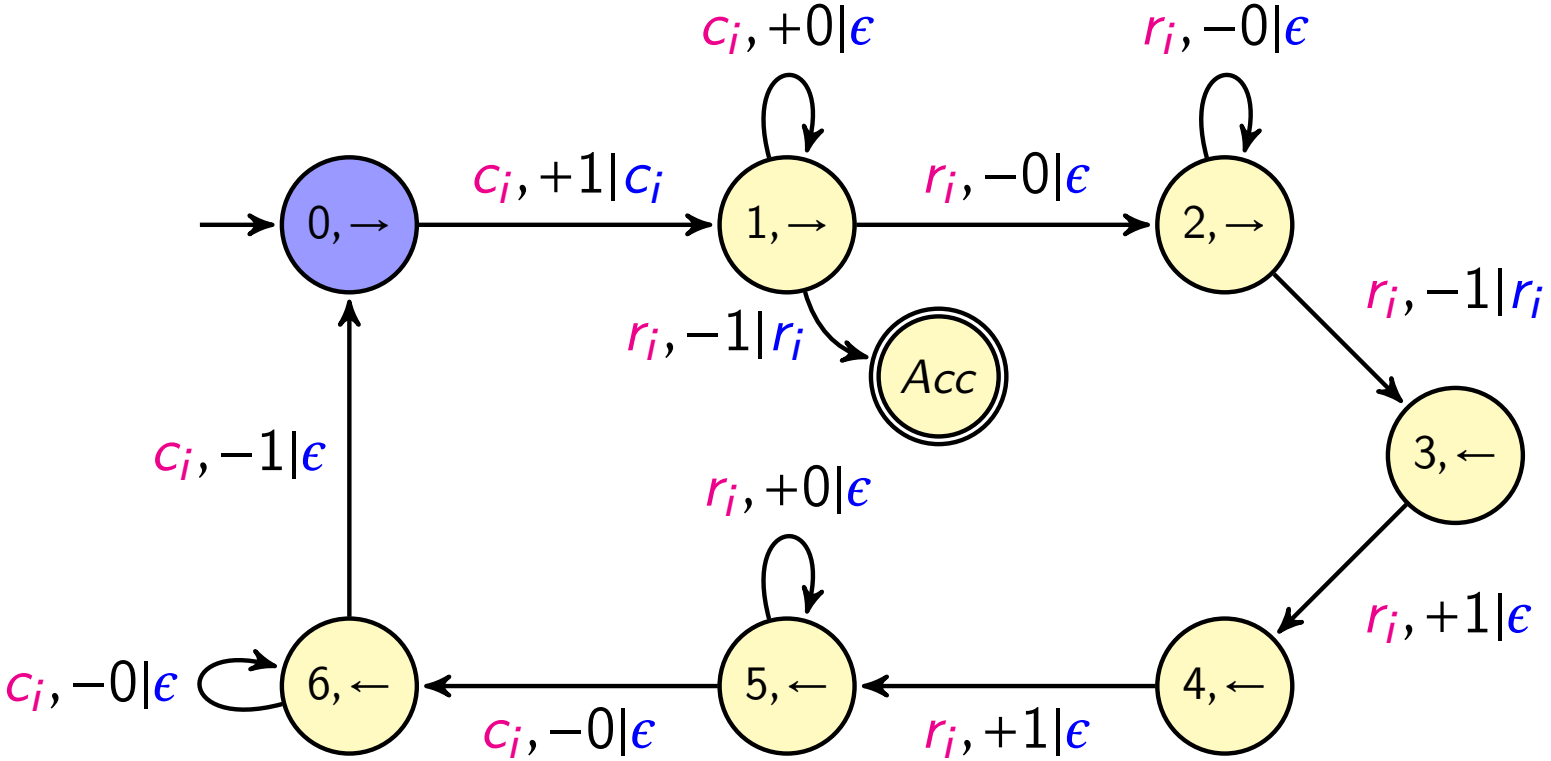
$(0, \rightarrow)$

$\perp$

Output Tape:

stack

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

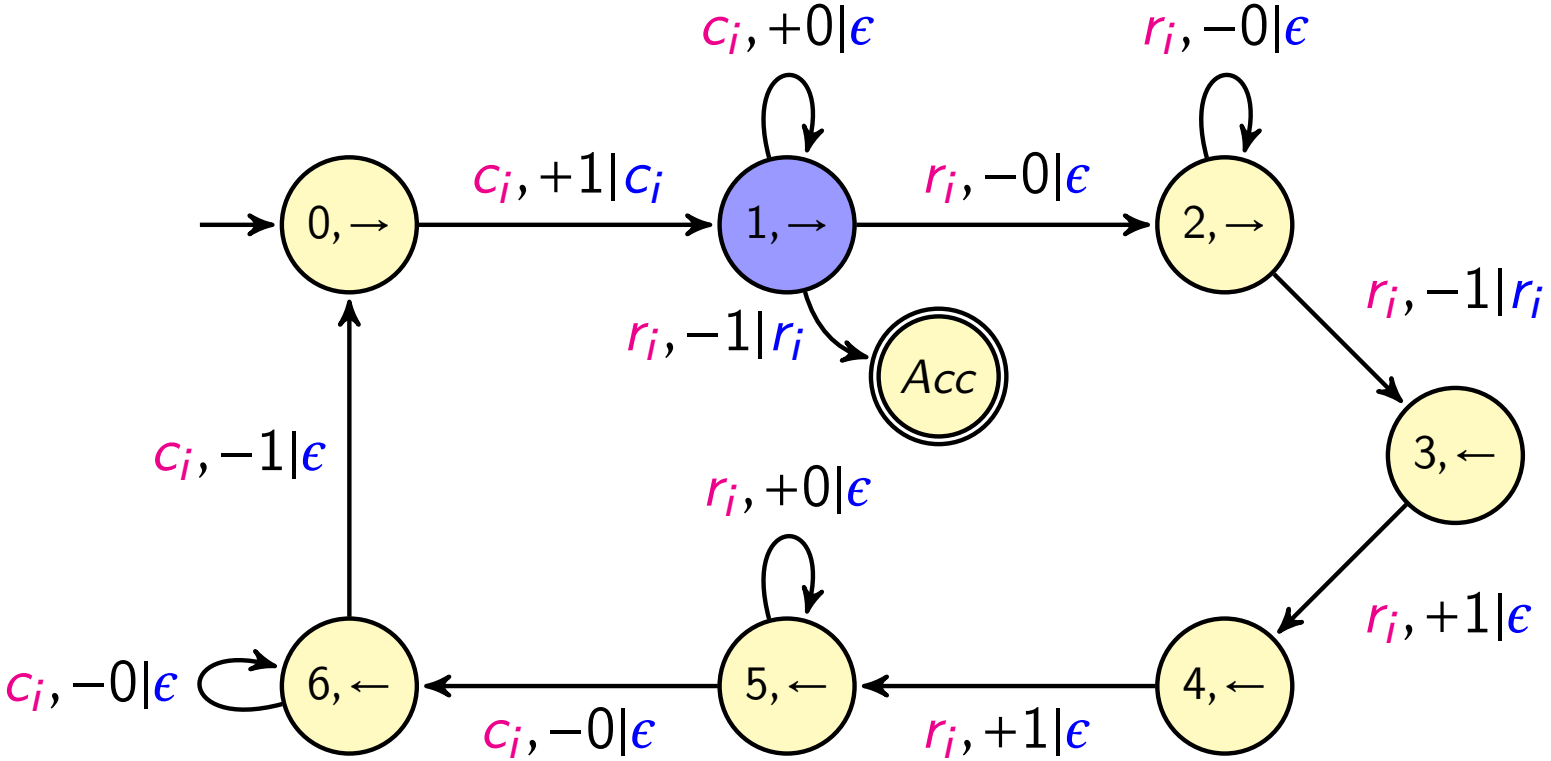
$(0, \rightarrow)$

$\perp$

Output Tape:

stack

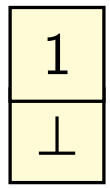
2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

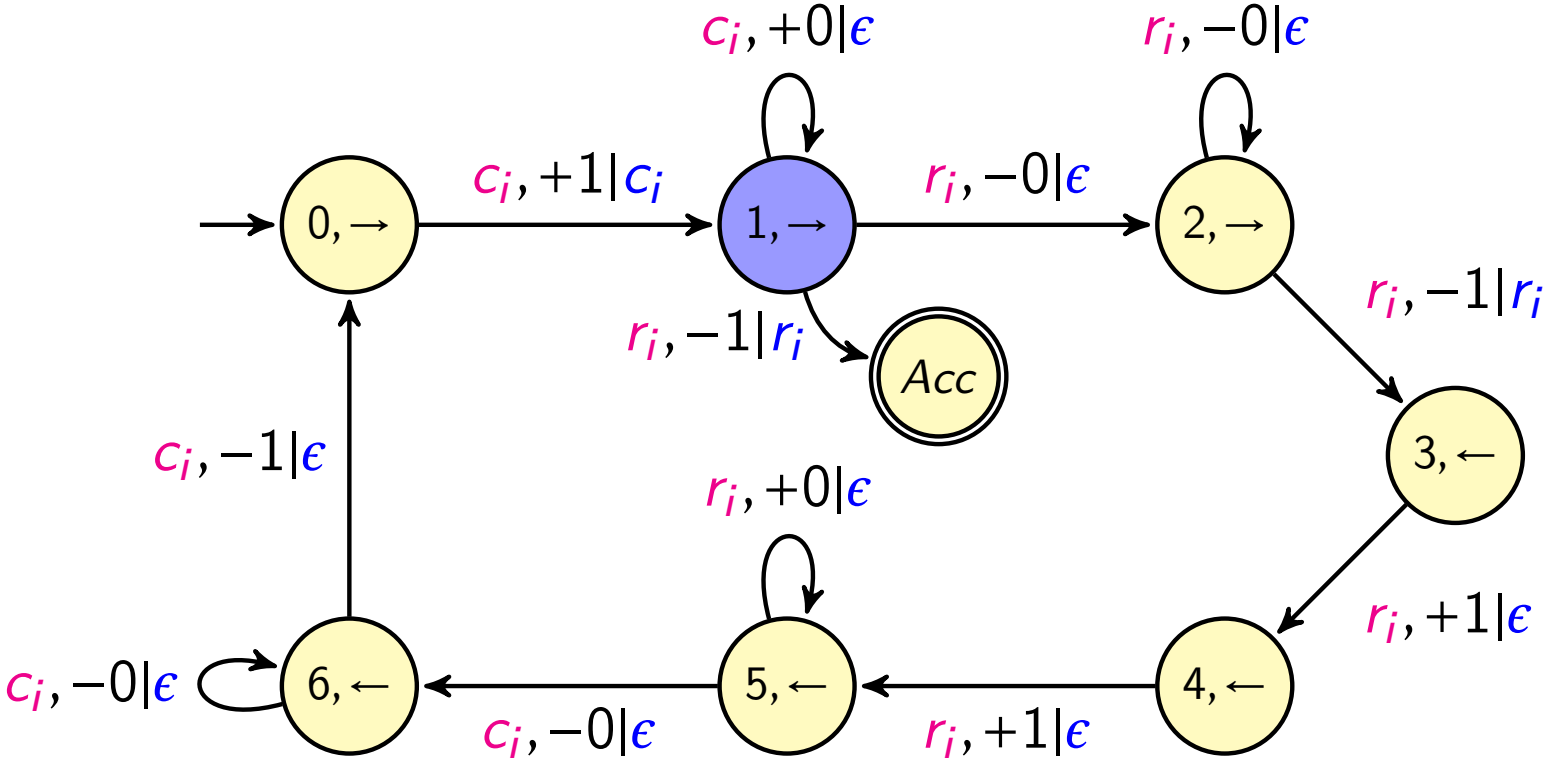
$(1, \rightarrow)$

Output Tape: c_1



stack

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

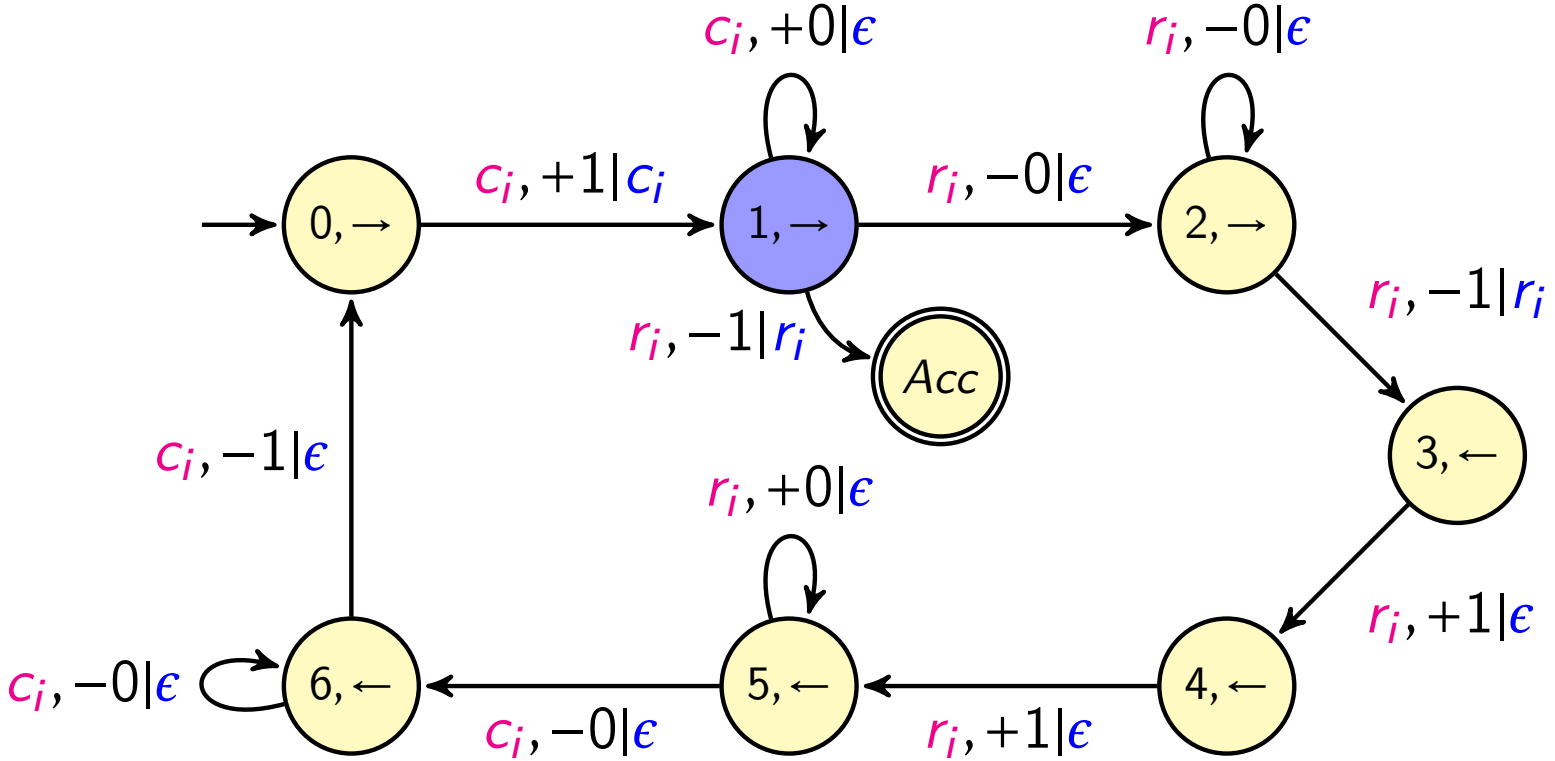
$(1, \rightarrow)$

Output Tape: c_1

0
1
$\perp$

stack

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

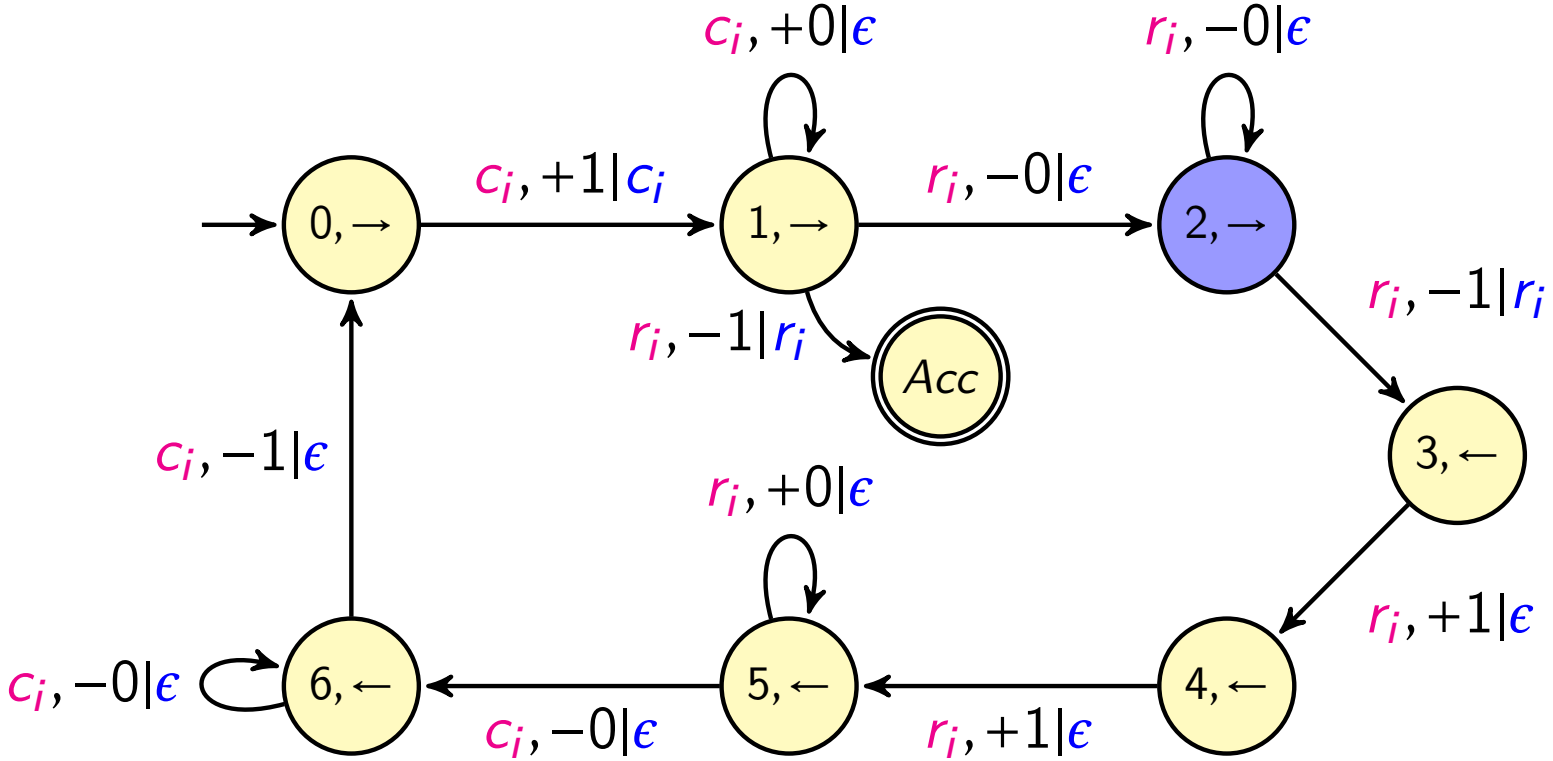
$(1, \rightarrow)$

Output Tape: c_1

0
0
1
$\perp$

stack

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

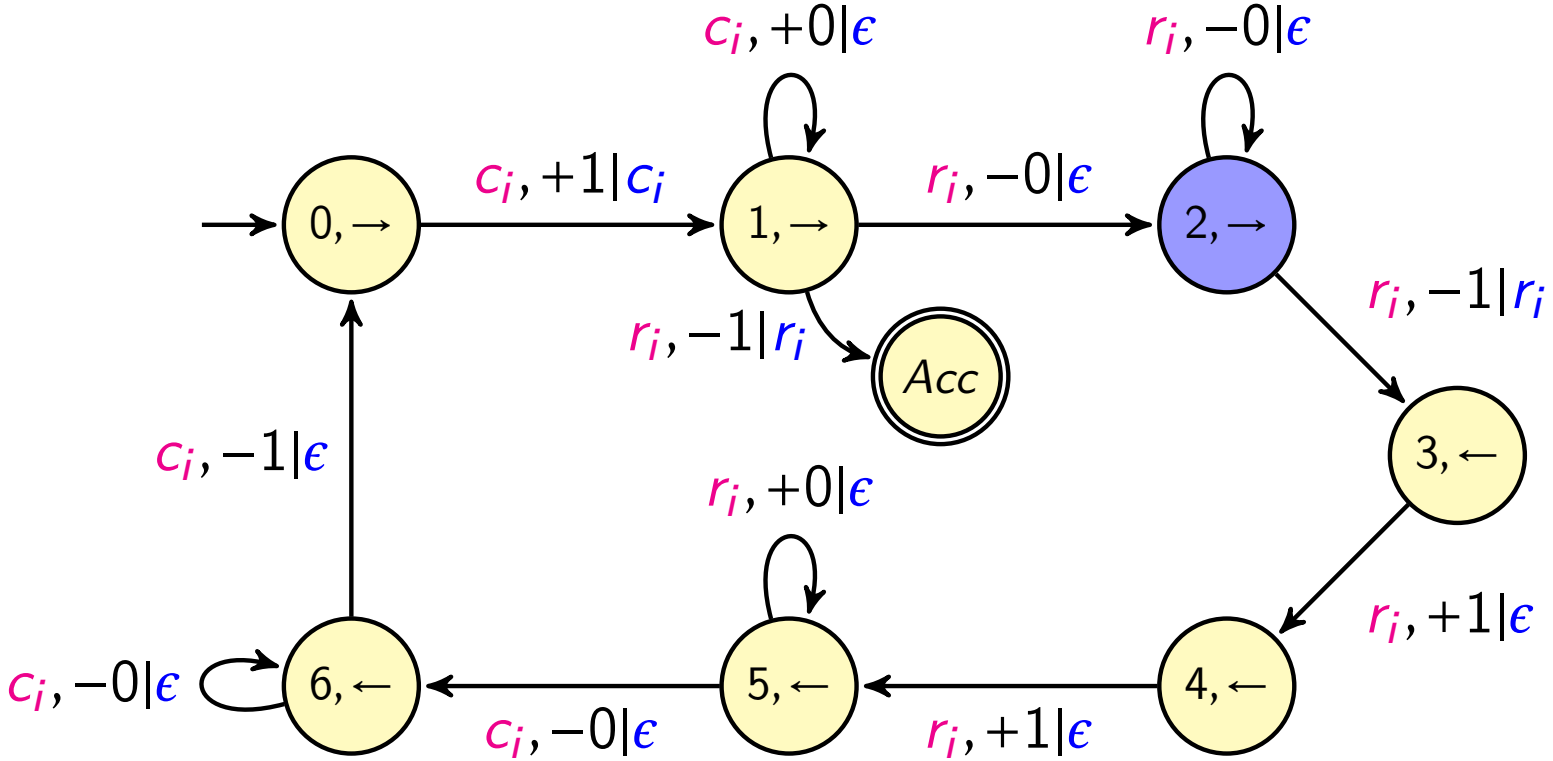
$(2, \rightarrow)$

0
1
$\perp$

stack

Output Tape: c_1

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

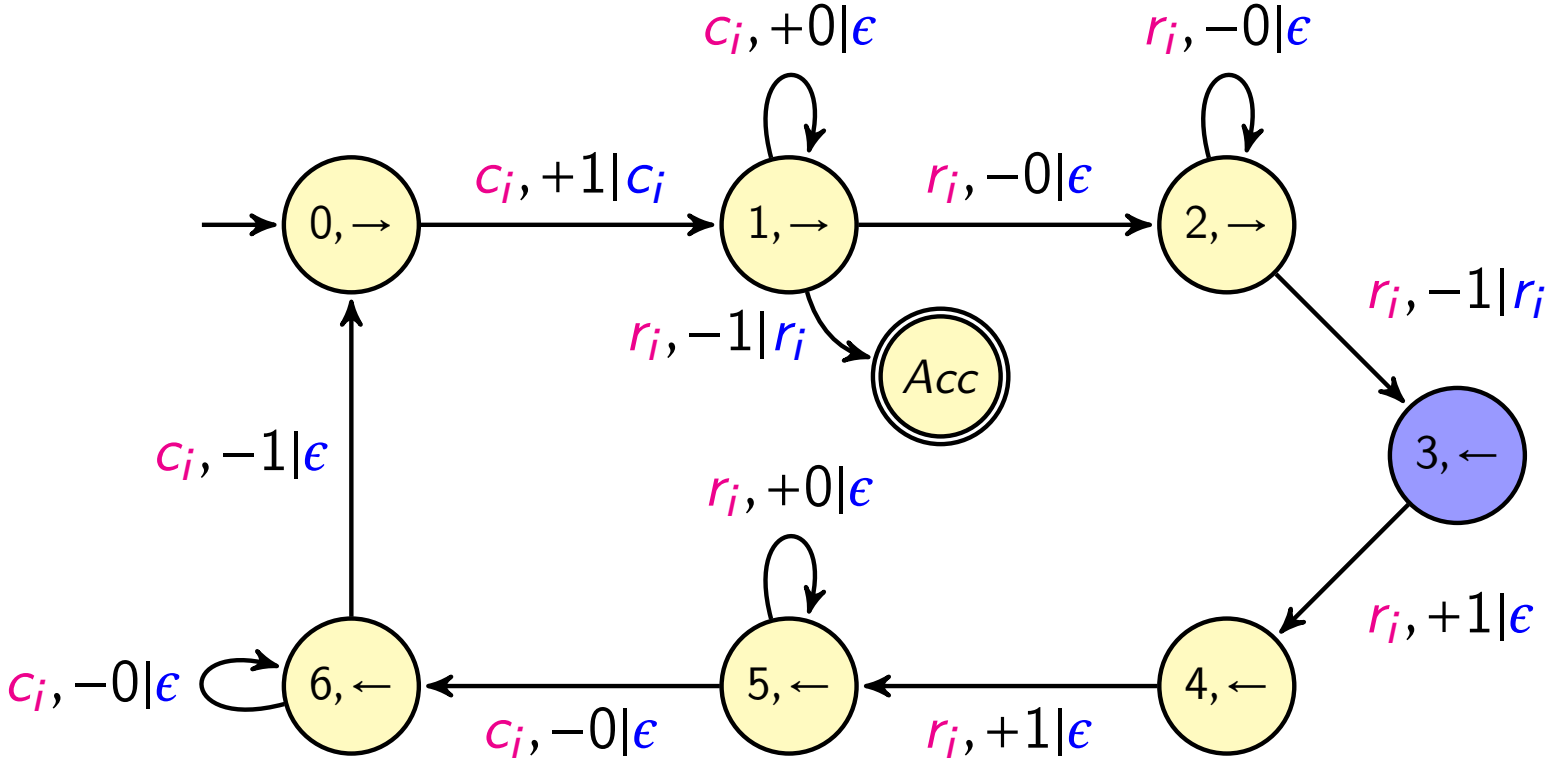
$(2, \rightarrow)$

1
$\perp$

stack

Output Tape: c_1

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

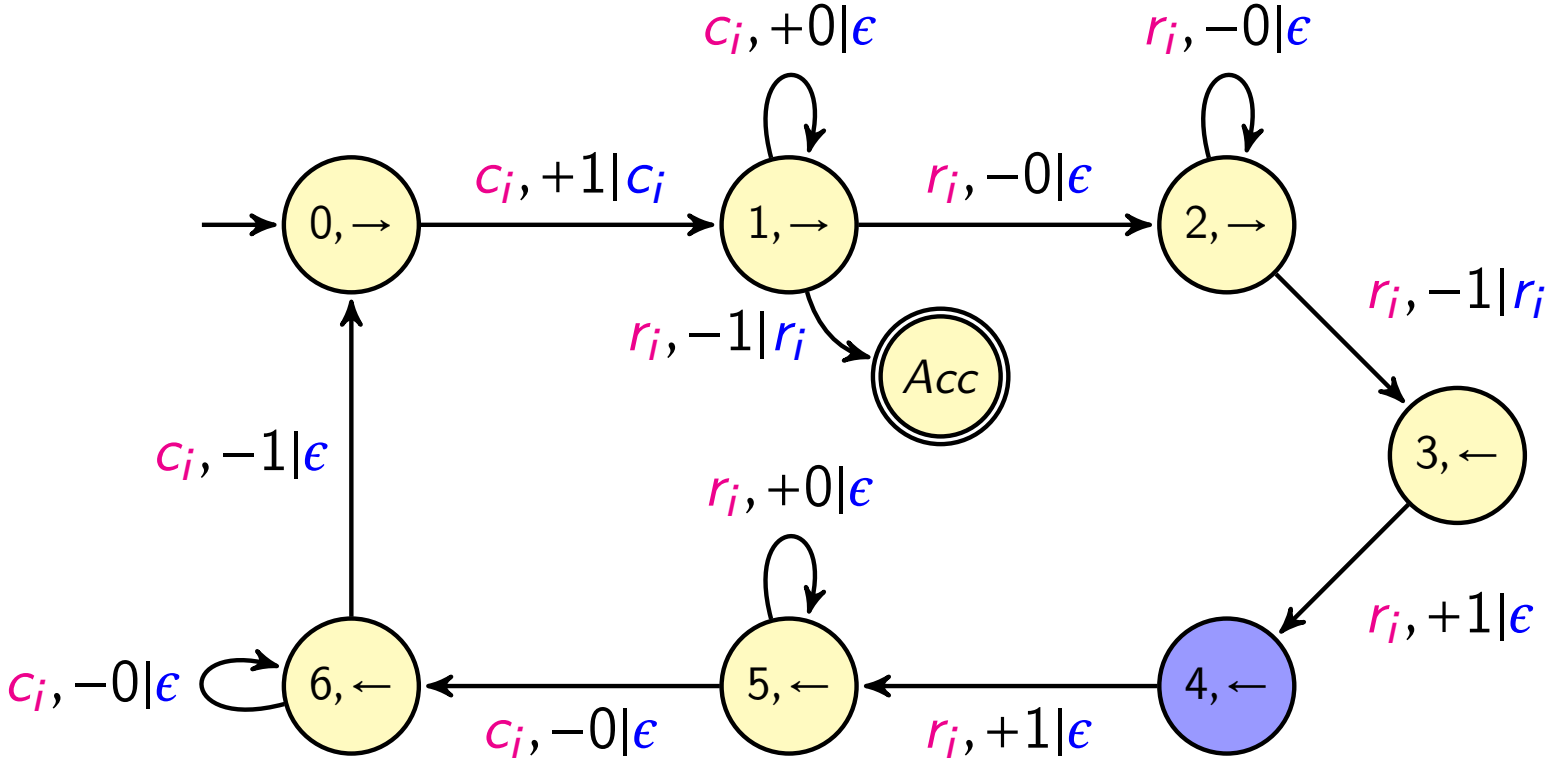
$(3, \leftarrow)$

$\perp$

stack

Output Tape: c_1 r_1

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

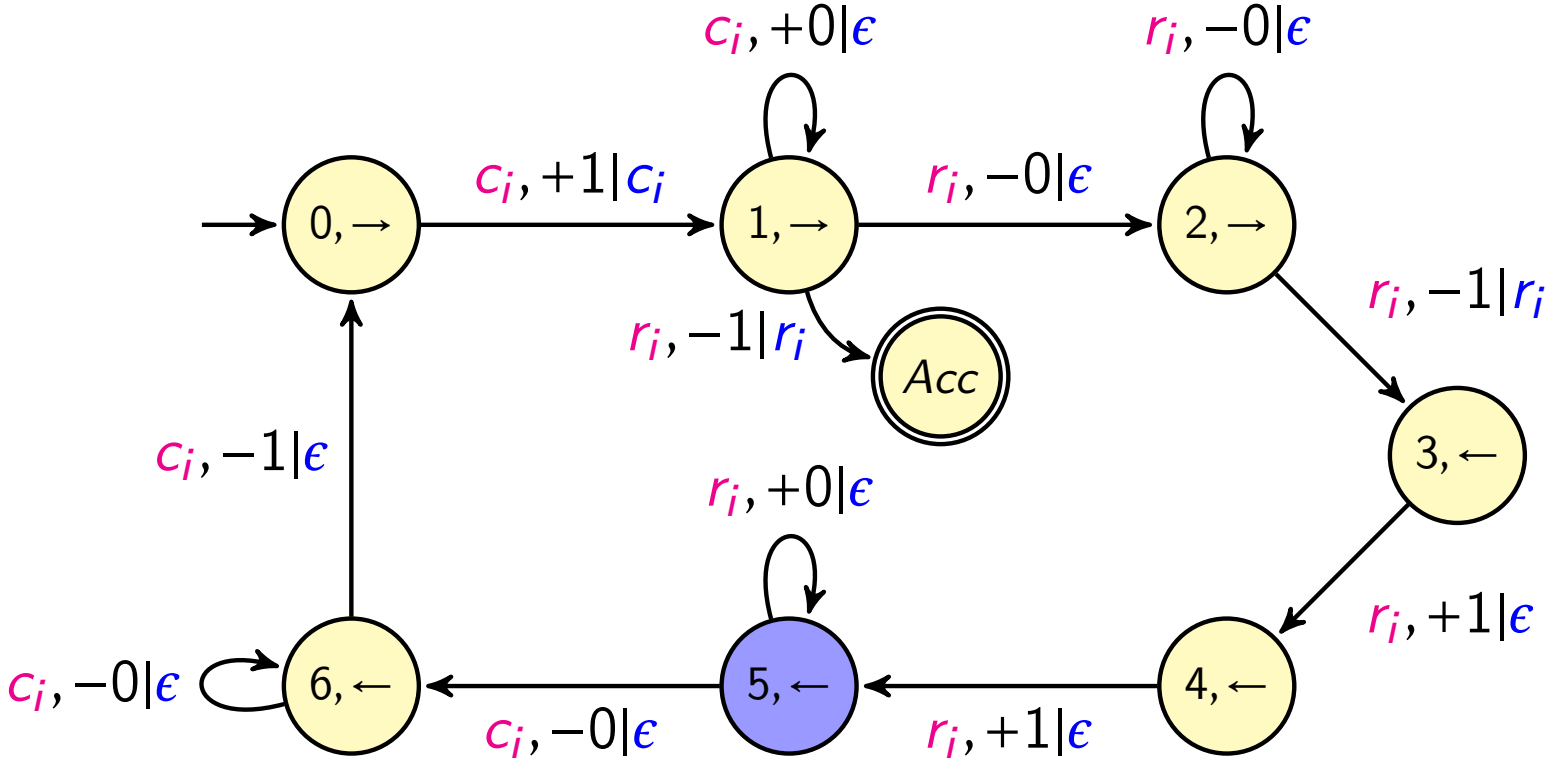
$(4, \leftarrow)$

1
$\perp$

stack

Output Tape: c_1 r_1

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

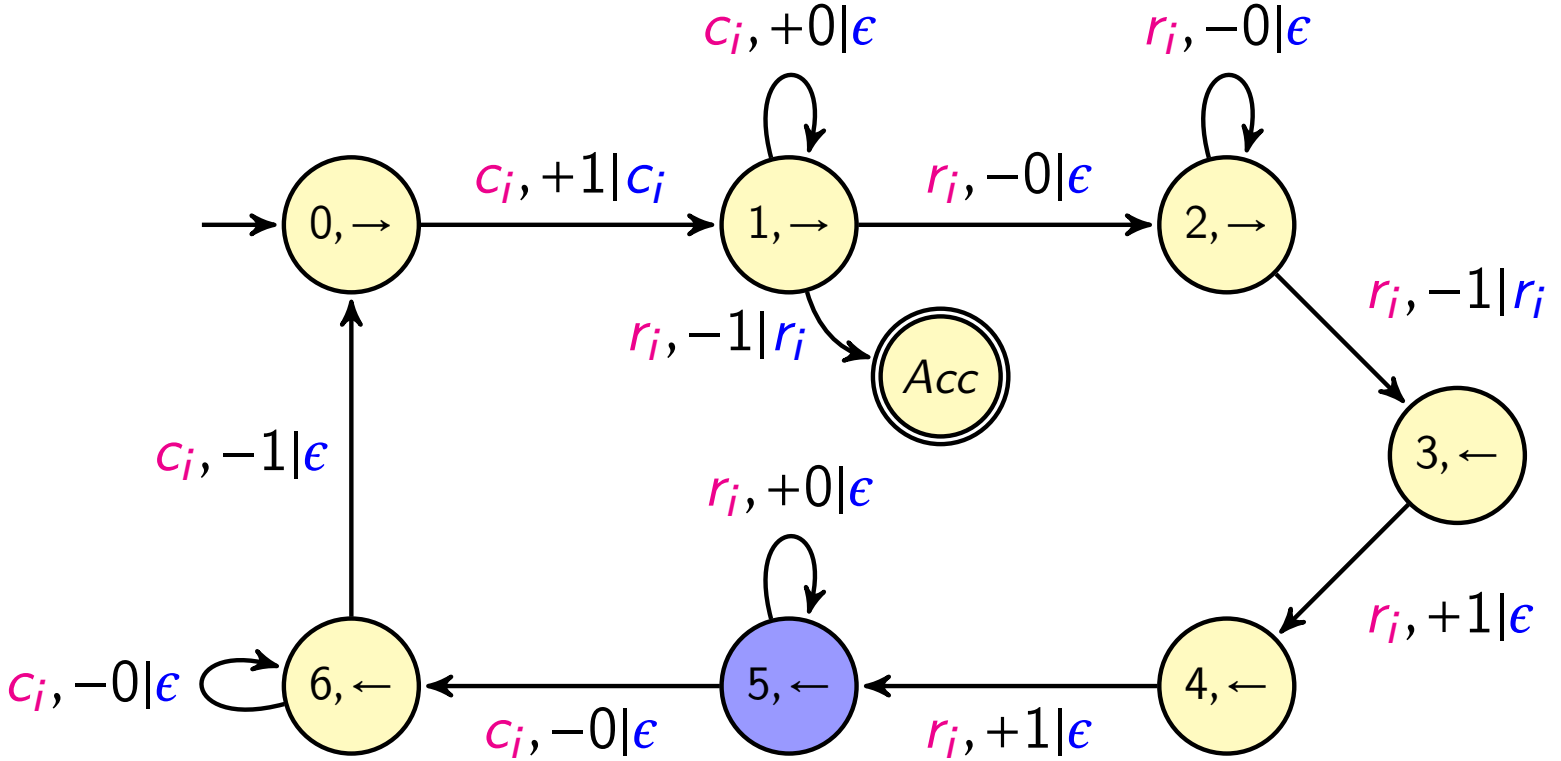
$(5, \leftarrow)$

Output Tape: c_1 r_1

1
1
$\perp$

stack

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

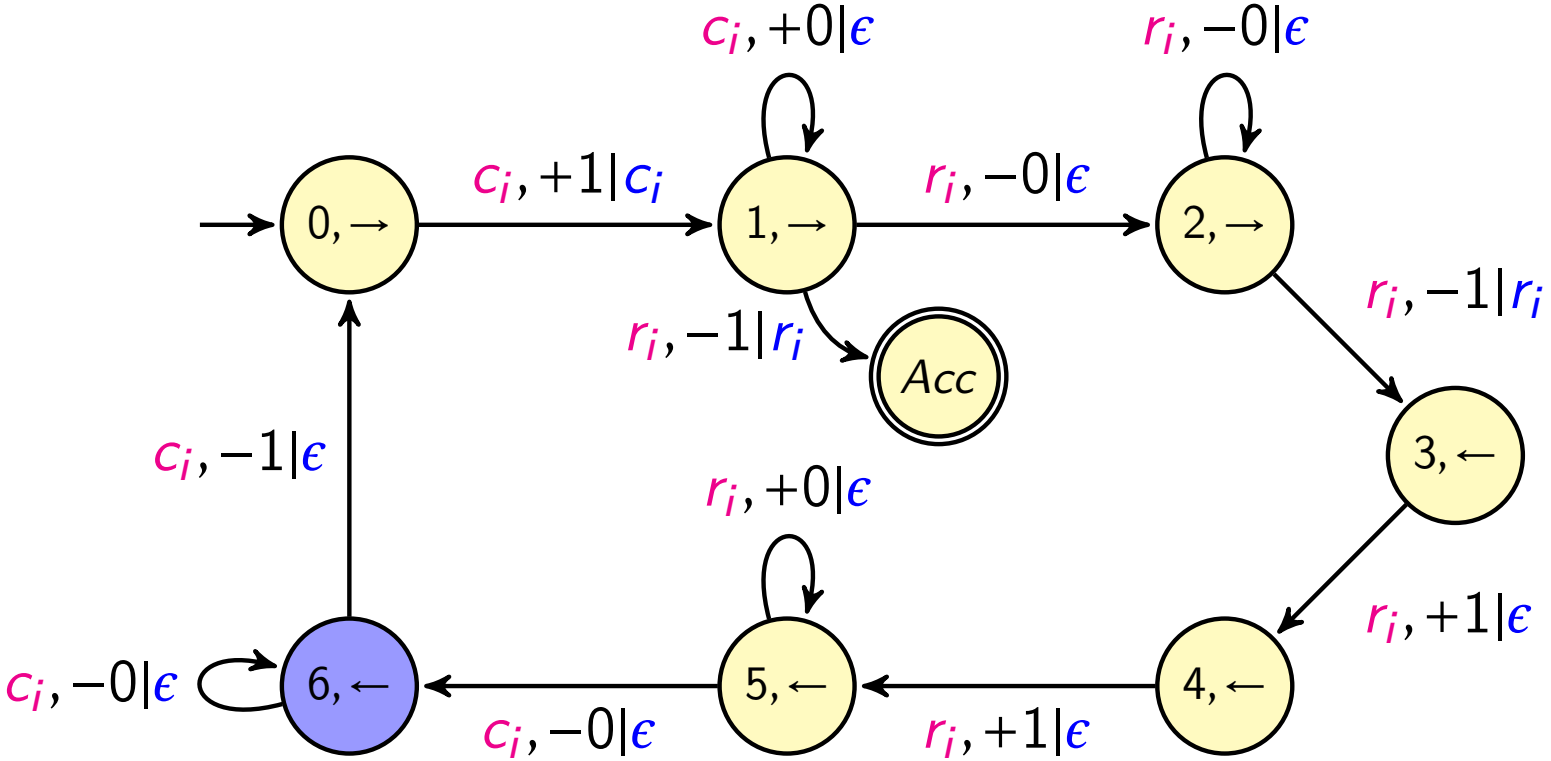
$(5, \leftarrow)$

Output Tape: c_1 r_1

0
1
1
$\perp$

stack

2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

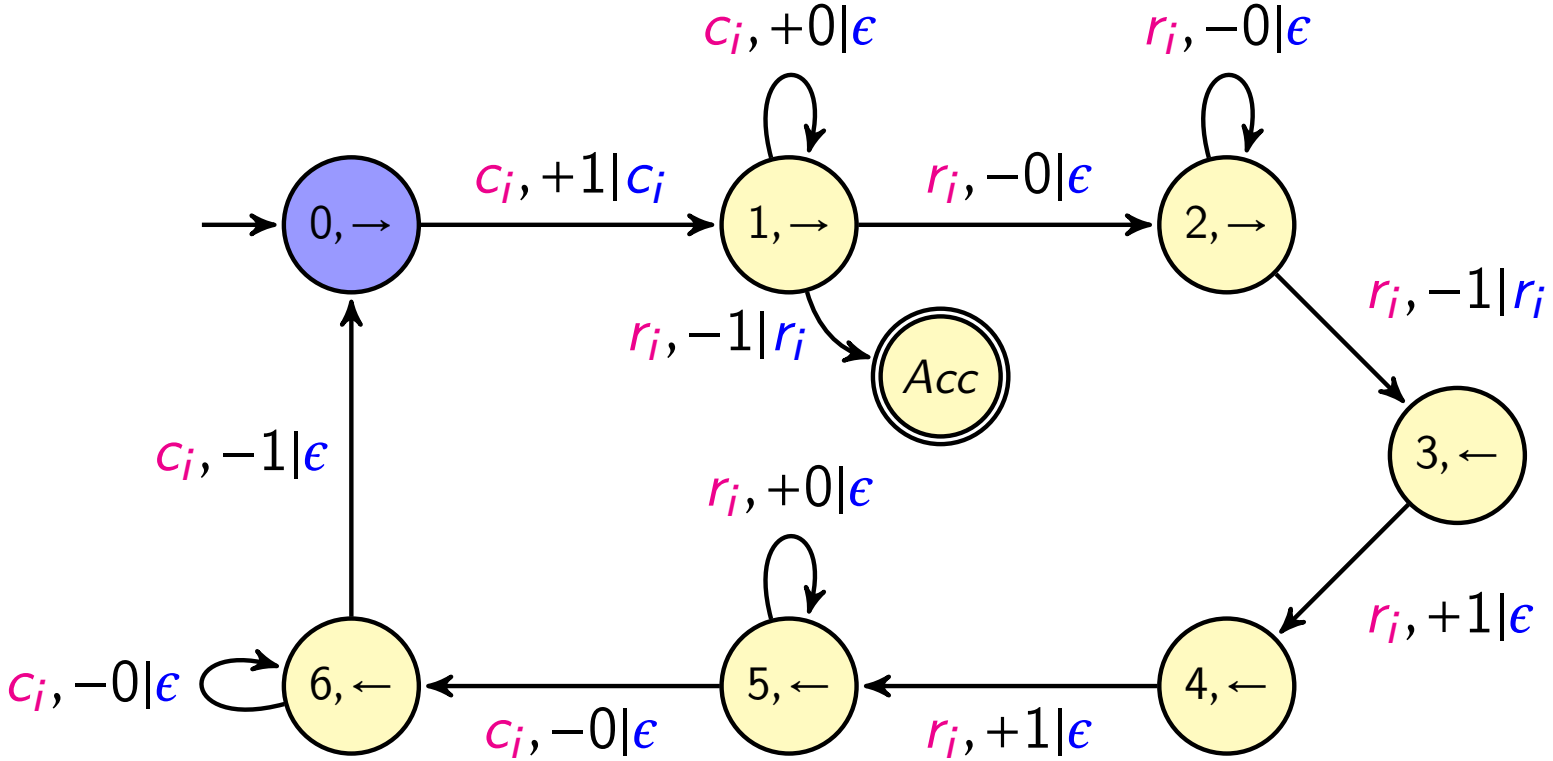
$(6, \leftarrow)$

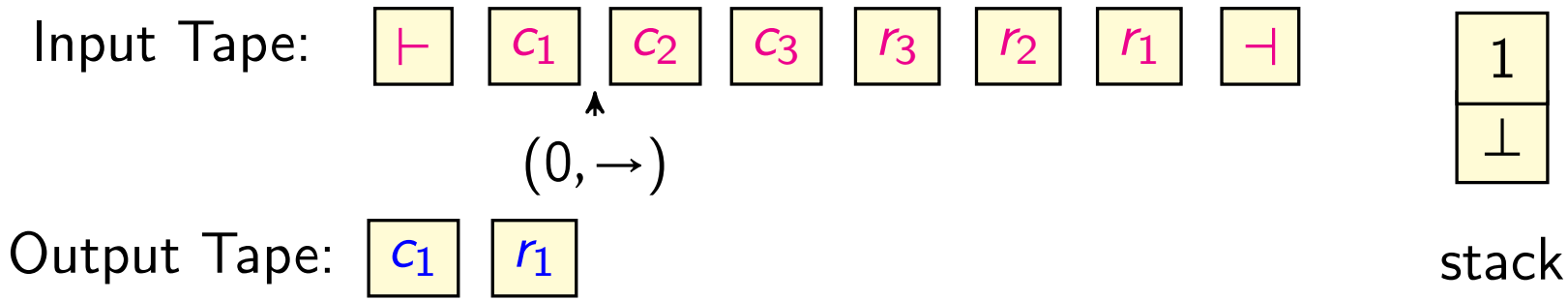
Output Tape: c_1 r_1

1
1
$\perp$

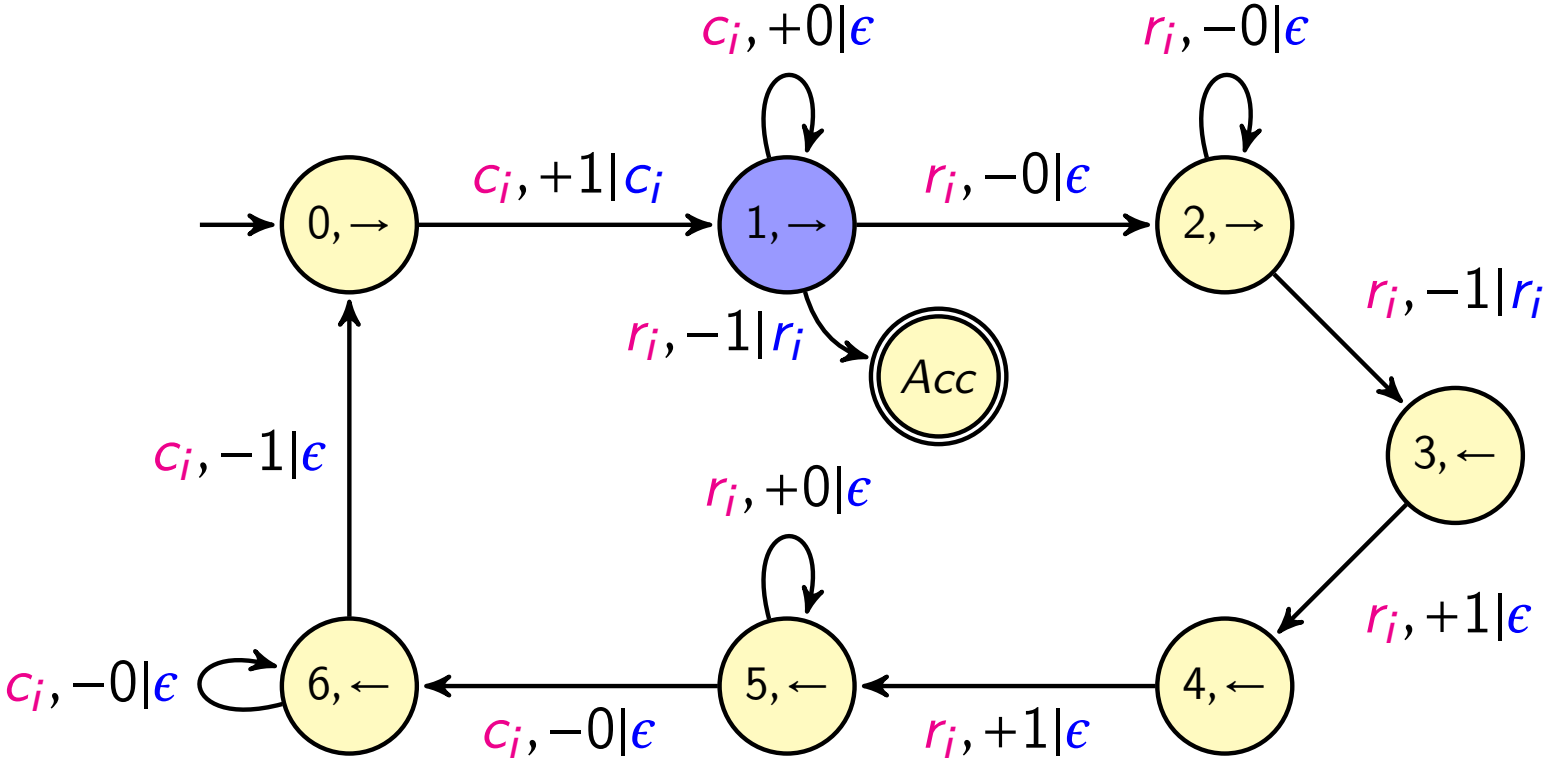
stack

2-way Visibly Pushdown Transducers (2VPT)





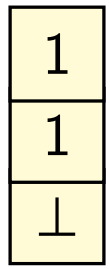
2-way Visibly Pushdown Transducers (2VPT)



Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

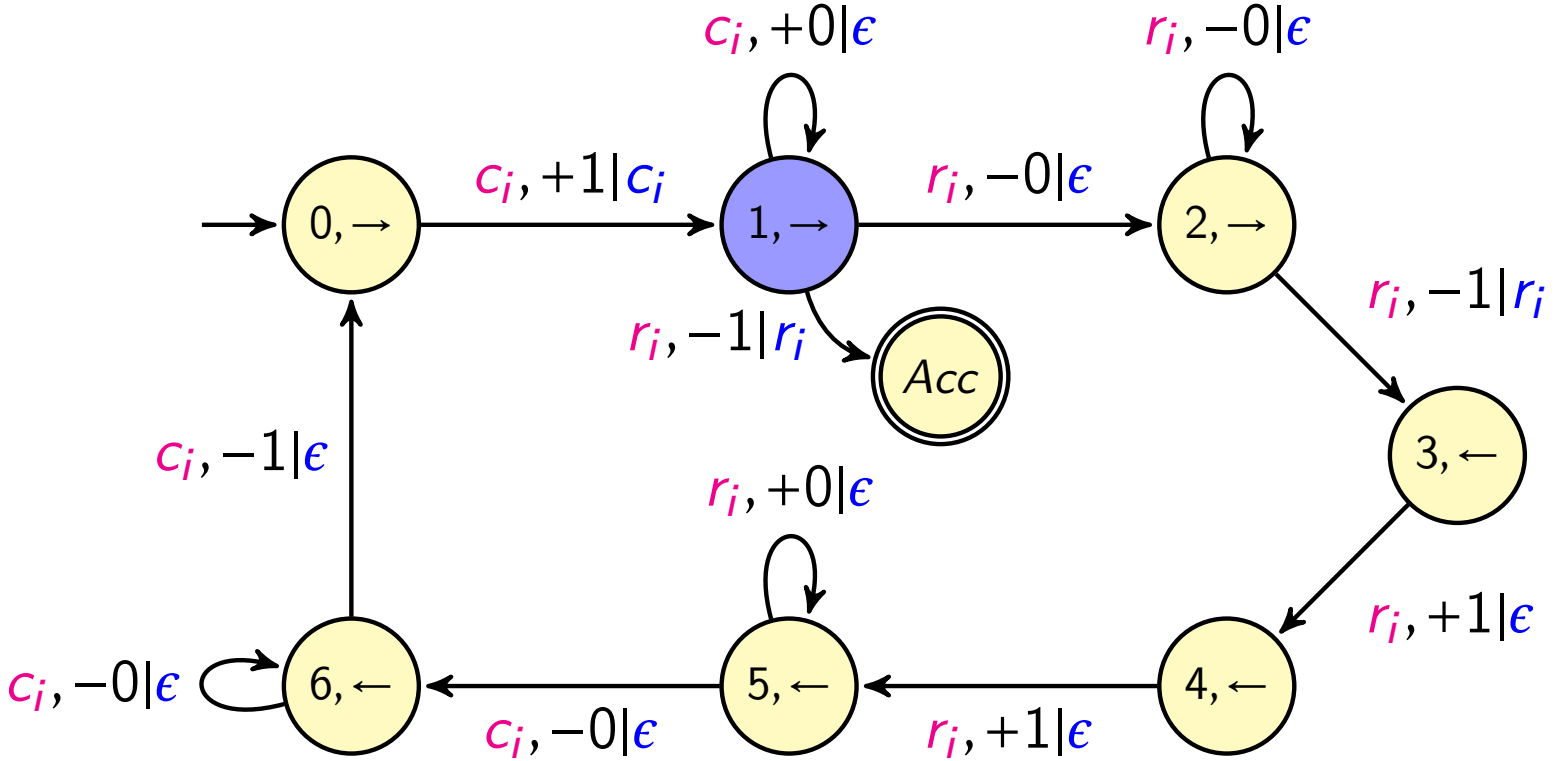
$(1, \rightarrow)$

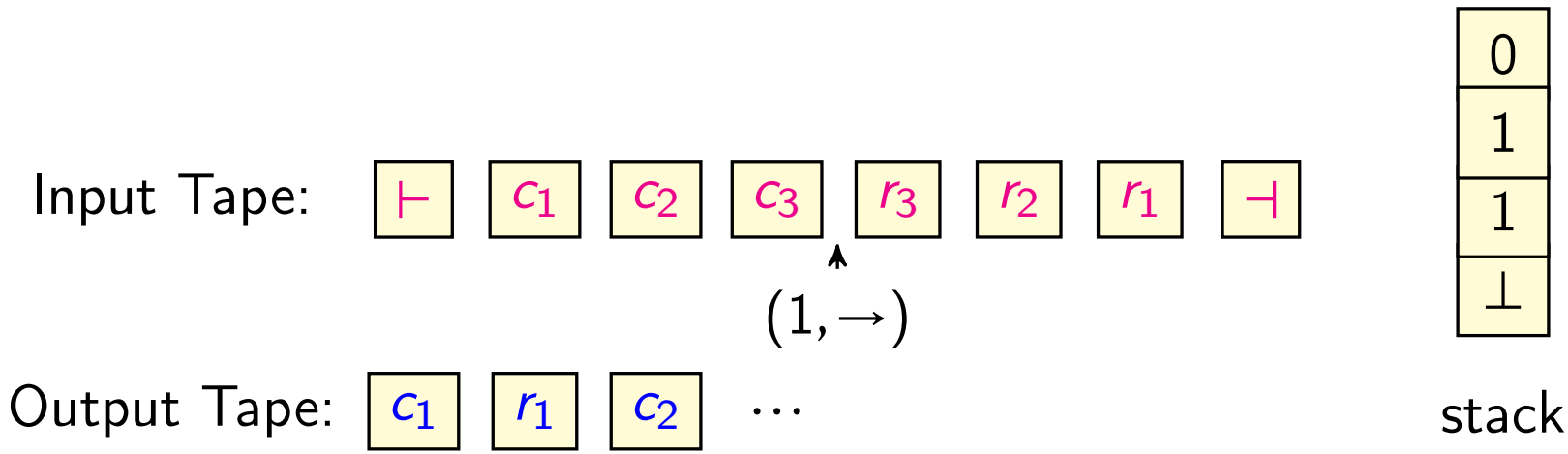
Output Tape: c_1 r_1 c_2



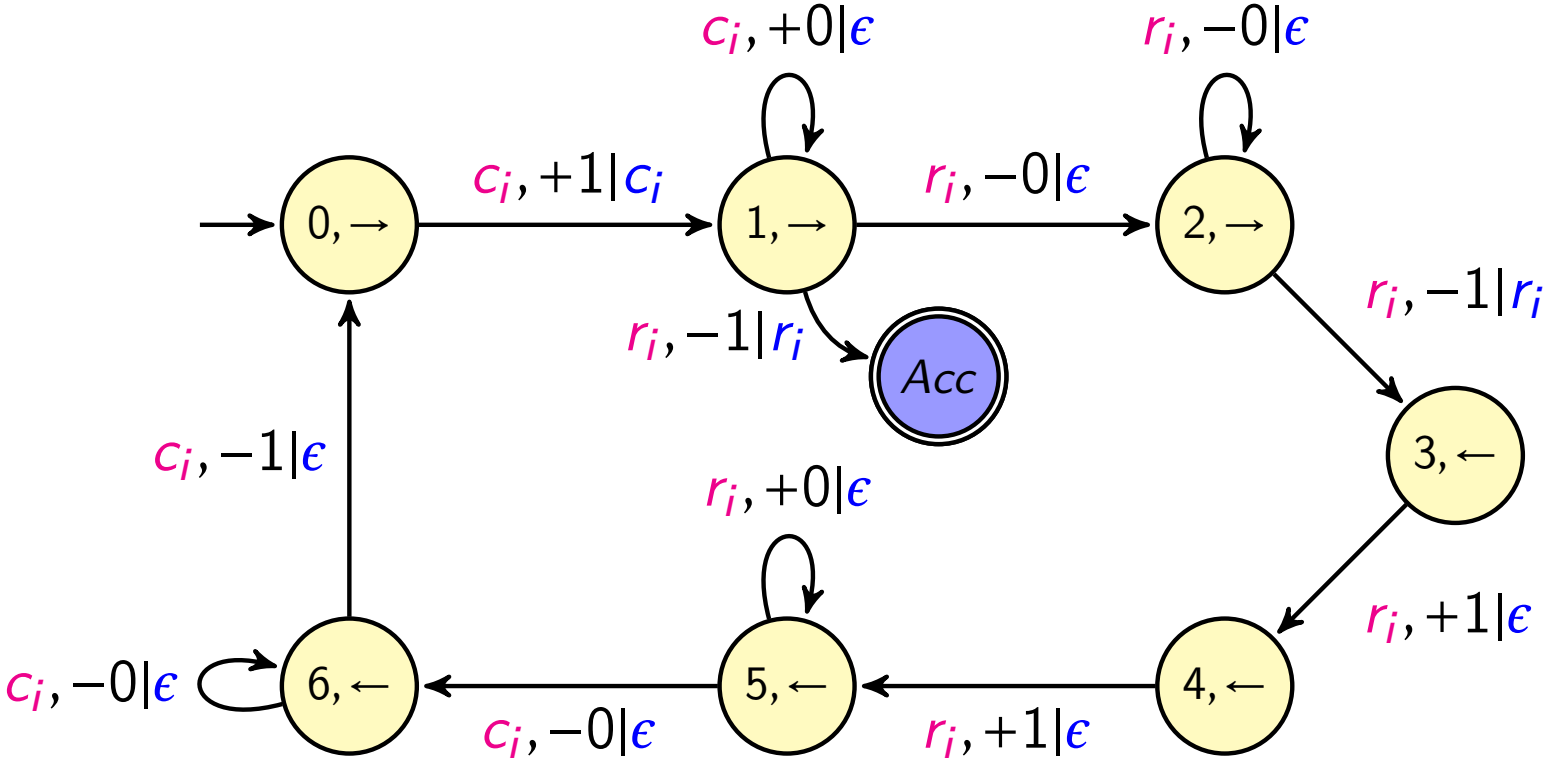
stack

2-way Visibly Pushdown Transducers (2VPT)





2-way Visibly Pushdown Transducers (2VPT)



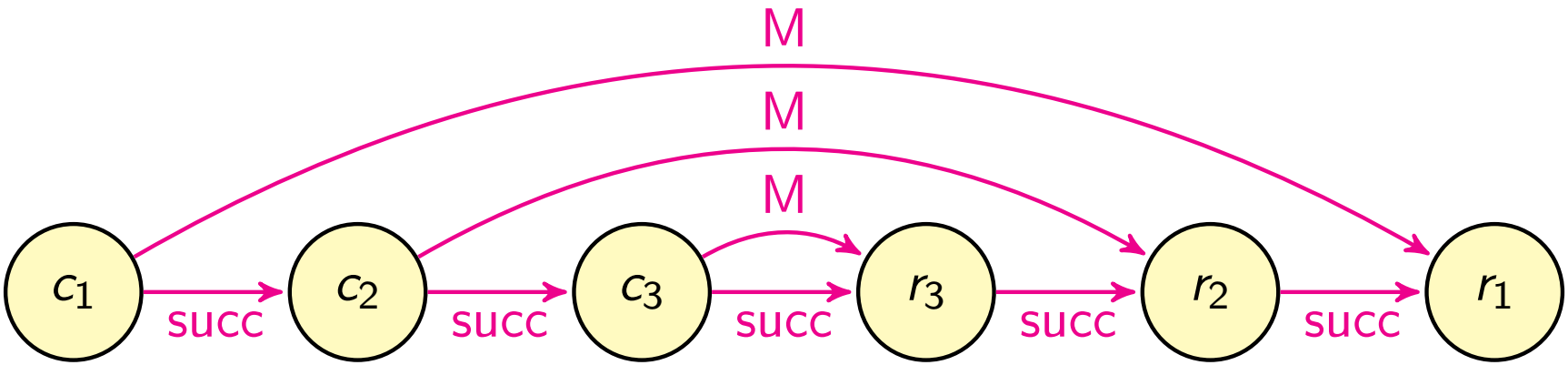
Input Tape: $\vdash$ c_1 c_2 c_3 r_3 r_2 r_1 $\dashv$

Output Tape: c_1 r_1 c_2 r_2 c_3 r_3

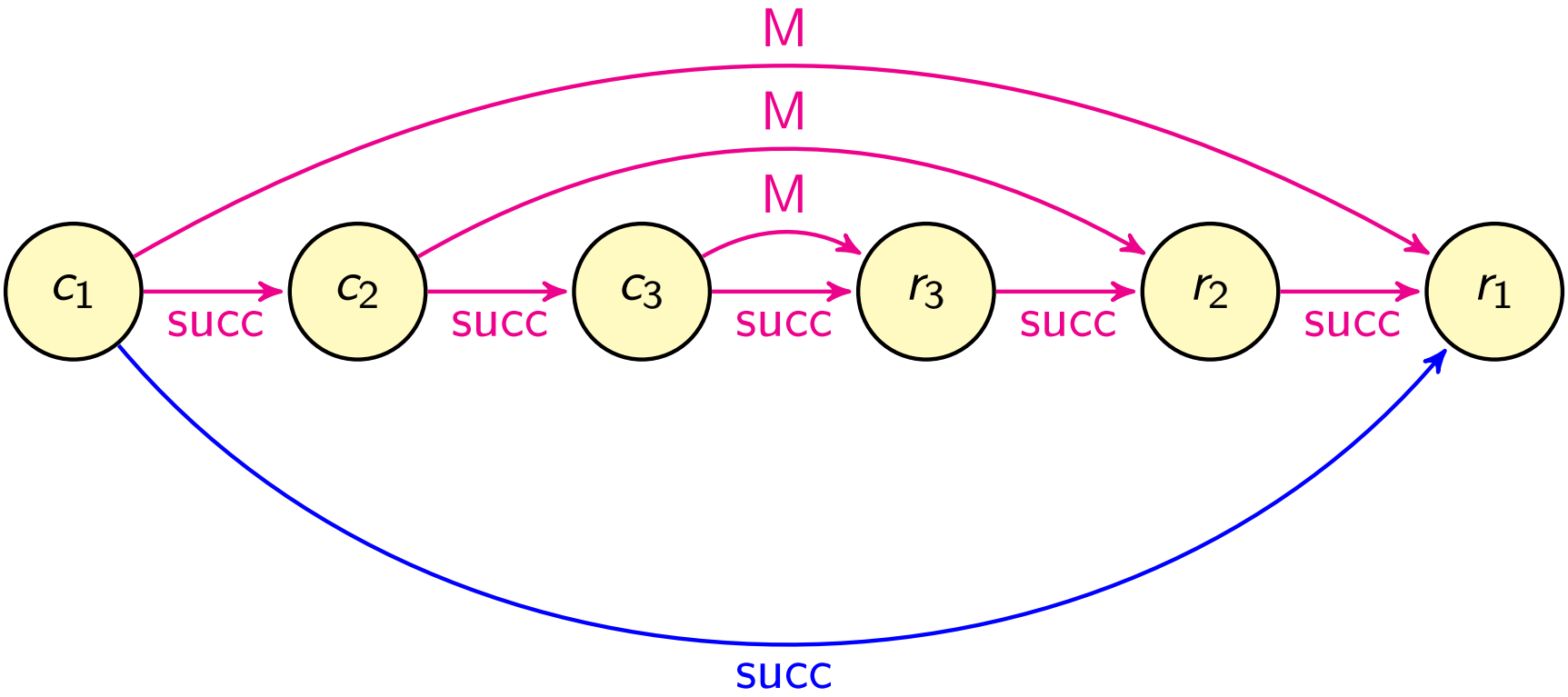
$\perp$

stack

MSO[nw2w] transduction

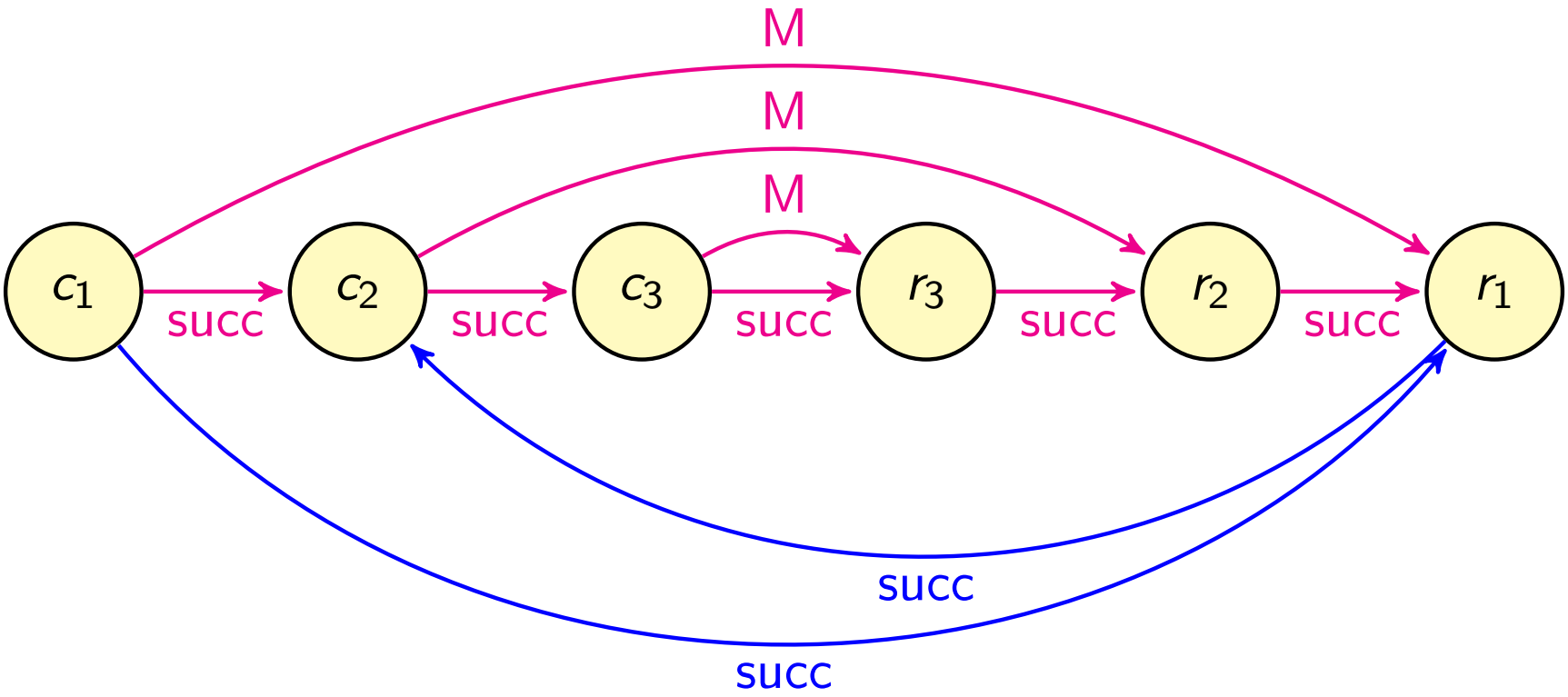


MSO[nw2w] transduction



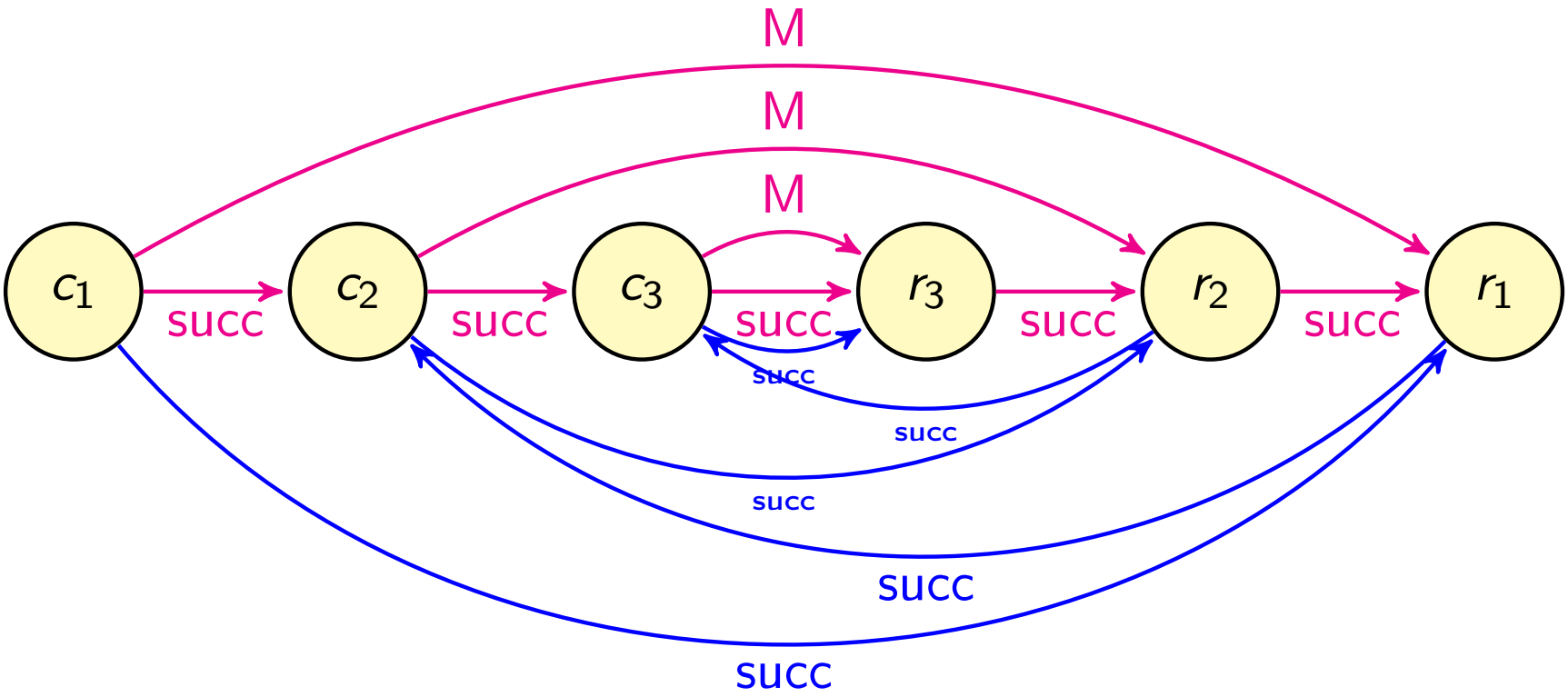
$$\phi_{lab_a}(x) \equiv lab_a(x)$$
$$\phi_{succ}(x,y) \equiv \bigvee_{i,j}(lab_{c_i}(x) \wedge M(x,y)) \vee (lab_{r_j}(x) \wedge \exists z\ M(z,x) \wedge succ(z,y))$$

MSO[nw2w] transduction



$$\phi_{lab_a}(x) \equiv lab_a(x)$$
$$\phi_{succ}(x,y) \equiv \bigvee_{i,j}(lab_{c_i}(x) \wedge M(x,y)) \vee (lab_{r_j}(x) \wedge \exists z\ M(z,x) \wedge succ(z,y))$$

MSO[nw2w] transduction



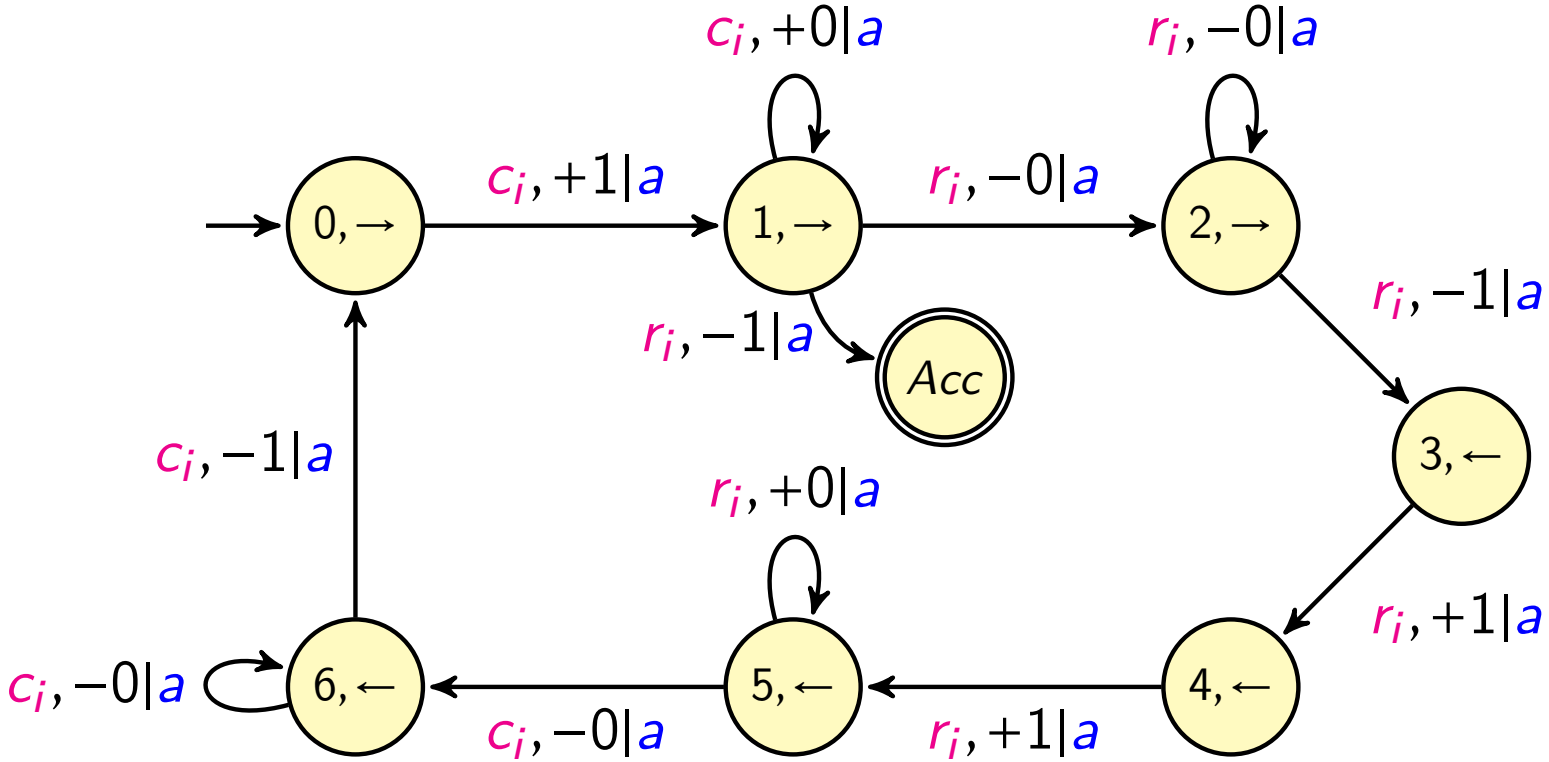
$$\phi_{lab_a}(x) \equiv lab_a(x)$$
$$\phi_{succ}(x,y) \equiv \bigvee_{i,j}(lab_{c_i}(x) \wedge M(x,y)) \vee (lab_{r_j}(x) \wedge \exists z\ M(z,x) \wedge succ(z,y))$$

Comparing expressiveness

- MSO[nw2w] transductions are **linear size increase**, as "a fixed number of copies" of the input structure

Comparing expressiveness

- MSO[nw2w] transductions are **linear size increase**, as "a fixed number of copies" of the input structure
- but (deterministic) 2VPT (D2VPT) are not.



$$|T(c^n r^n)| = O(n^2)$$

Single-use Property

Definition (Single-use restriction)

A D2VPT is *single-use* (D2VPT_{su}) if in any accepting run, any producing transition occur at most once at a given position.

Single-use Property

Definition (Single-use restriction)

A D2VPT is *single-use* (D2VPT_{su}) if in any accepting run, any producing transition occur at most once at a given position.

Proposition (Su decision)

Given a D2VPT, *deciding if it satisfies the single-use property is Exptime-complete.*

Some results ([Dartois, Filiot, Reynier, T. 16)

- Expressiveness :
  - D2VPT_{su} and MSO[nw2w] are equally expressive.

Some results ([Dartois, Filiot, Reynier, T. 16)

- Expressiveness :
  - D2VPT_{su} and MSO[nw2w] are equally expressive.
  - functional VPT and order-preserving MSO[nw2w] are equally expressive.

Some results ([Dartois, Filiot, Reynier, T. 16)

- Expressiveness :
  - D2VPT_{su} and MSO[nw2w] are equally expressive.
  - functional VPT and order-preserving MSO[nw2w] are equally expressive.

- Type-checking problem against
  - a regular language is Exptime-complete.
  - a VPA language is undecidable even for deterministic (one-way) VPT (followed from the undecidability of inclusion of CFL into VPA languages)

Some results ([Dartois, Filiot, Reynier, T. 16)

- Expressiveness :
  - D2VPT_{su} and MSO[nw2w] are equally expressive.
  - functional VPT and order-preserving MSO[nw2w] are equally expressive.
- Type-checking problem against
  - a regular language is Exptime-complete.
  - a VPA language is undecidable even for deterministic (one-way) VPT (followed from the undecidability of inclusion of CFL into VPA languages)
- Composition is not possible (even for deterministic (one-way) VPT)

Outline

2 - Tree to Tree Transformations

Tree to Tree Transformations with (2)VPT

Requires :

- output on a structured alphabet
- output range in Σ^*_{wn}

Tree to Tree Transformations with (2)VPT

Requires :

- output on a structured alphabet
- output range in Σ^*_{wn}

Definition [Filiot,Raskin,Reynier,Servais,T. - '10]

A VPT is locally well-nested if

for all stack symbols γ and all transitions $p_1 \xrightarrow{c|w_1,+\gamma} p_2$ and $q_2 \xrightarrow{r|w_2,-\gamma} q_1$,

$$w_1 w_2 \in \Sigma^*_{\text{wn}}$$

Tree to Tree Transformations with (2)VPT

Requires :

- output on a structured alphabet
- output range in Σ^*_{wn}

Definition [Filiot,Raskin,Reynier,Servais,T. - '10]

A VPT is locally well-nested if

for all stack symbols γ and all transitions $p_1 \xrightarrow{c|w_1,+\gamma} p_2$ and $q_2 \xrightarrow{r|w_2,-\gamma} q_1$,

$$w_1 w_2 \in \Sigma^*_{\text{wn}}$$

- syntactic restriction
- locally well-nested VPTs produce well-nested words
- closed under composition
- typechecking against VPA is ExpTime-complete

Well-Nested VPT

What about a semantical class $\{T \in \mathrm{VPT} \mid \forall L,\ [\![T]\!](L) \subseteq \Sigma^*_{\mathrm{wn}}\}$?

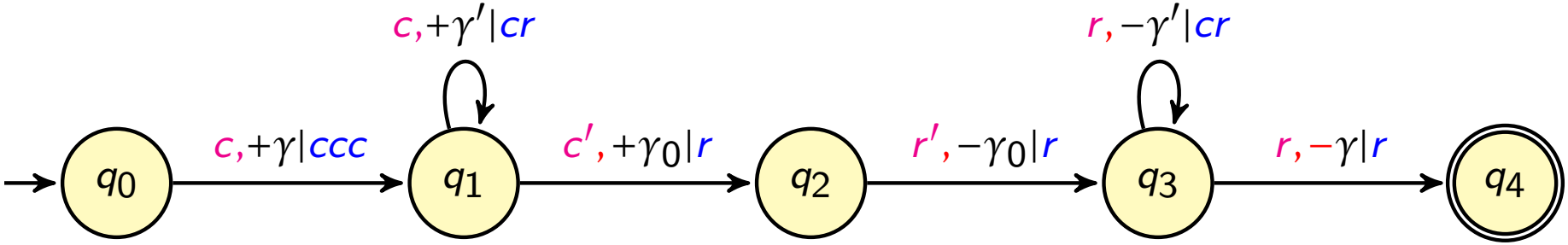
Well-Nested VPT

What about a semantical class $\{T \in \mathrm{VPT} \mid \forall L,\ [\![T]\!](L) \subseteq \Sigma^*_{\mathrm{wn}}\}$?

This is the class of **well-nested** VPT

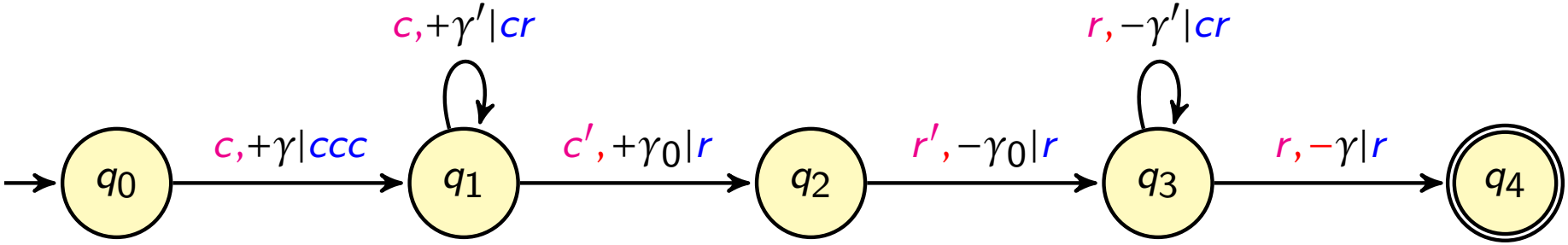
- can this class be decided ?
- is this class closed by composition ? (and thus, has type-checking against VPA is decidable)
- how does it relate to locally well-nested VPT?

Locally Well-Nested vs Well-Nested VPT

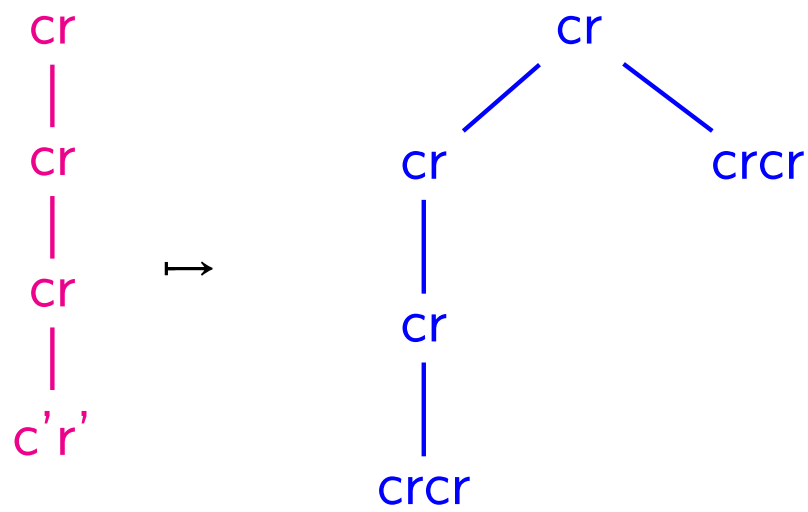


$$[\![T]\!] = \{(cc^k c' r' r^k r, ccc(cr)^k rr(cr)^k r) \mid k \in \mathbb{N}\}$$

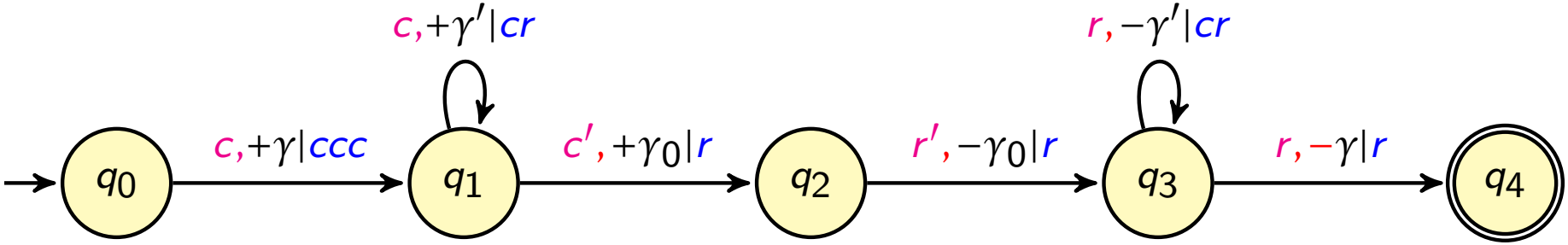
Locally Well-Nested vs Well-Nested VPT



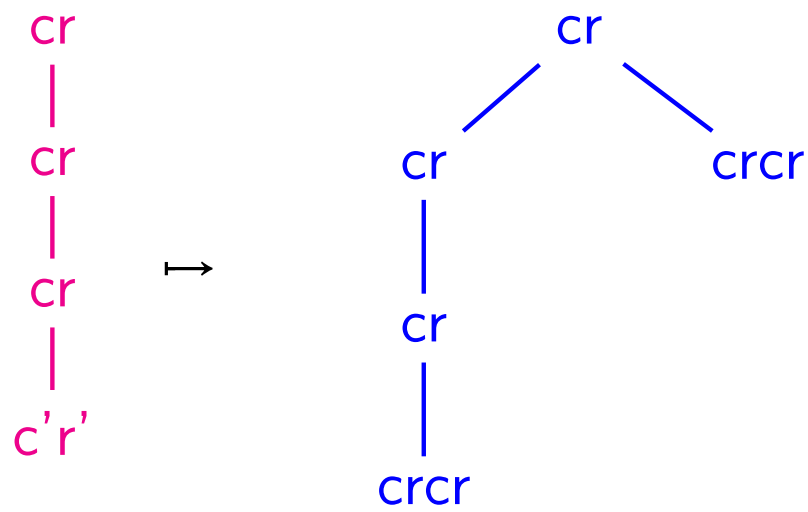
$$[\![T]\!] = \{(cc^k c' r' r^k r, ccc(cr)^k rr(cr)^k r) \mid k \in \mathbb{N}\}$$



Locally Well-Nested vs Well-Nested VPT



$$[\![T]\!] = \{(cc^k c' r' r^k r, ccc(cr)^k rr(cr)^k r) \mid k \in \mathbb{N}\}$$



locally well-nested VPT $\subsetneq$ well-nested VPT

Properties of Well-Nested VPT

Proposition (Reynier T. '14)

The class of well-nested VPT*:*

- *can be decided in* PTime
- *is effectively closed under composition*
- *has a type-checking problem against* VPA *in 2-*ExpTime*.*

Properties of Well-Nested VPT

Proposition (Reynier T. '14)

The class of well-nested VPT*:*

- *can be decided in* PTime
- *is effectively closed under composition*
- *has a type-checking problem against* VPA *in 2-*ExpTime*.*

Proposition

Well-nested fVPT $\equiv \mathrm{fVPT} \cap (\Sigma_{\mathrm{wn}}^* \to \Sigma_{\mathrm{wn}}^*)$

Properties of Well-Nested VPT

Proposition (Reynier T. '14)

The class of well-nested VPT*:*

- *can be decided in* PTime
- *is effectively closed under composition*
- *has a type-checking problem against* VPA *in 2-*ExpTime*.*

Proposition

Well-nested fVPT $\equiv \text{fVPT} \cap (\Sigma_{\text{wn}}^* \to \Sigma_{\text{wn}}^*)$
$\equiv$ *order-preserving* $\text{MSO}[\text{nw2w}] \cap (\Sigma_{\text{wn}}^* \to \Sigma_{\text{wn}}^*)$

Properties of Well-Nested VPT

Proposition (Reynier T. '14)

The class of well-nested VPT*:*

- *can be decided in* PTime
- *is effectively closed under composition*
- *has a type-checking problem against* VPA *in* 2-ExpTime.

Proposition

Well-nested fVPT $\equiv \text{fVPT} \cap (\Sigma^*_{\text{wn}} \to \Sigma^*_{\text{wn}})$
$\equiv$ *order-preserving* MSO[nw2w] $\cap (\Sigma^*_{\text{wn}} \to \Sigma^*_{\text{wn}})$
$\equiv$ *order-preserving* MSO[nw2nw]

As, output matching is MSO-definable.

Properties of Well-Nested VPT

Proposition (Reynier T. '14)

The class of well-nested VPT*:*

- *can be decided in* PTime
- *is effectively closed under composition*
- *has a type-checking problem against* VPA *in* 2-ExpTime.

Proposition

Well-nested fVPT $\equiv \text{fVPT} \cap (\Sigma_{\text{wn}}^* \to \Sigma_{\text{wn}}^*)$
$\equiv$ *order-preserving* MSO[nw2w] $\cap (\Sigma_{\text{wn}}^* \to \Sigma_{\text{wn}}^*)$
$\equiv$ *order-preserving* MSO[nw2nw]

As, output matching is MSO-definable.

Can we decide local transformations amongst global ones ?

D2VPT for tree to tree transformations

Tree restriction

$\mathsf{D2VPT}_{\mathsf{su}}$	$\equiv$	MSO[nw2w]

D2VPT for tree to tree transformations

Tree restriction

$\mathsf{D2VPT}_{\mathsf{su}} \cap(\Sigma^*_{\mathsf{wn}} \to \Sigma^*_{\mathsf{wn}}) \quad \equiv \quad \mathsf{MSO[nw2w]} \cap(\Sigma^*_{\mathsf{wn}} \to \Sigma^*_{\mathsf{wn}})$

D2VPT for tree to tree transformations

Tree restriction

$\mathsf{D2VPT}_{\mathsf{su}} \cap (\Sigma^*_{\mathsf{wn}} \to \Sigma^*_{\mathsf{wn}}) \quad \equiv \quad \mathsf{MSO[nw2w]} \cap (\Sigma^*_{\mathsf{wn}} \to \Sigma^*_{\mathsf{wn}})$

Some questions :

- is $\mathsf{D2VPT}_{\mathsf{su}} \subseteq (\Sigma^*_{\mathsf{wn}} \to \Sigma^*_{\mathsf{wn}})$ is decidable (when structured output alphabet) ?
- Instead of D2VPT, what about functional 2VPT? Is it a decidable class ?

D2VPT for tree to tree transformations

Tree restriction

$\text{D2VPT}_{\text{su}} \cap (\Sigma^*_{\text{wn}} \to \Sigma^*_{\text{wn}}) \quad \equiv \quad \text{MSO[nw2w]} \cap (\Sigma^*_{\text{wn}} \to \Sigma^*_{\text{wn}})$

Some questions :

- is $\text{D2VPT}_{\text{su}} \subseteq (\Sigma^*_{\text{wn}} \to \Sigma^*_{\text{wn}})$ is decidable (when structured output alphabet) ?
- Instead of D2VPT, what about functional 2VPT? Is it a decidable class ?

Several questions but a single tool :

Two-way Visibly Pushdown Parikh Automata (2VPPA)

Outline

4 - A new Tool : (Two-way) Visibly Pushdown Parikh Automata

Parikh Automata

Definition (Klaedtke, Rueß, 03)

A Parikh automaton $P = (A, dim, \lambda, S)$ where $A = (\Sigma, Q, I, F, \Delta)$ is an NFA, dim is a natural number, $\lambda : \Delta \mapsto \mathbb{N}^{dim}$ and S is a semi-linear subset of $\mathbb{N}^{dim}$.

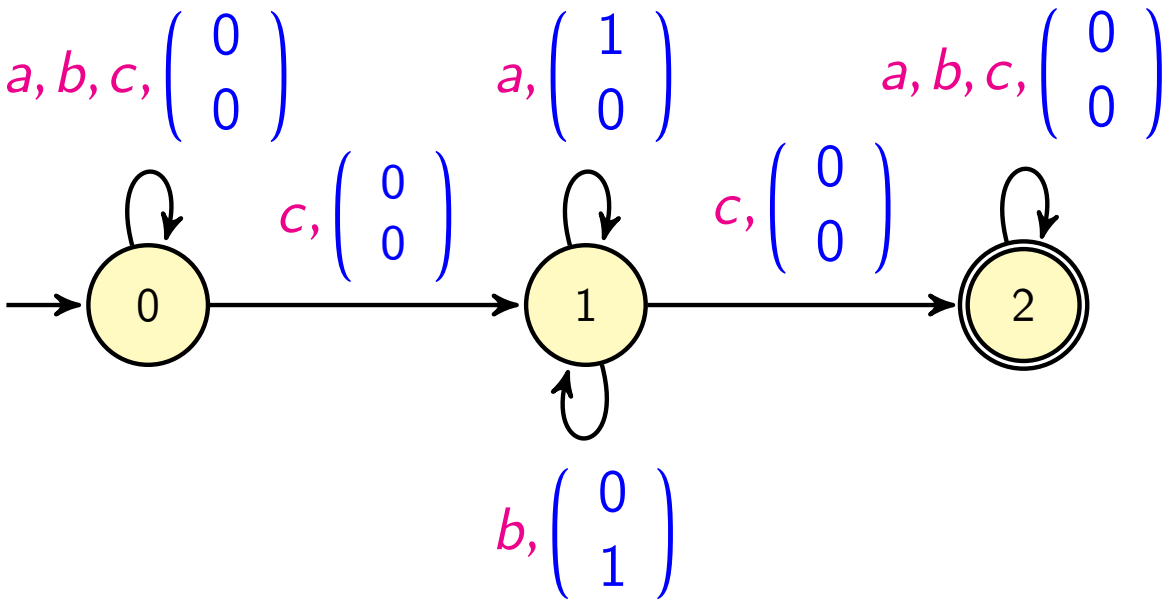
Here, S as a Presburger formula with dim free variables $x_1, \ldots, x_{dim}$

Parikh Automata

Definition (Klaedtke, Rueß, 03)

A Parikh automaton $P = (A, dim, \lambda, S)$ where $A = (\Sigma, Q, I, F, \Delta)$ is an NFA, dim is a natural number, $\lambda : \Delta \mapsto \mathbb{N}^{dim}$ and S is a semi-linear subset of $\mathbb{N}^{dim}$.

Here, S as a Presburger formula with dim free variables $x_1, \ldots, x_{dim}$



$$\varphi_S(x_1, x_2) = (x_1 = 2 * x_2 + 1)$$

Words containing a factor cwc with $w \in (a+b)^*$ and $|w|_a = 2|w|_b + 1$

Parikh Automata

Definition (Klaedtke, Rueß, 03)

A Parikh automaton $P = (A, dim, \lambda, S)$ where $A = (\Sigma, Q, I, F, \Delta)$ is an NFA, dim is a natural number, $\lambda : \Delta \mapsto \mathbb{N}^{dim}$ and S is a semi-linear subset of $\mathbb{N}^{dim}$.

Here, S as a Presburger formula with dim free variables $x_1, \ldots, x_{dim}$

Equi-expressive to (non-deterministic) reversal-bounded counter machines [Ibarra,78] (weaker in the deterministic case [Cadilhac, Finkel, McKenzie,11]).

In Parikh automata, counter values (and thus, updates) do not influence the control state evolution.

Two-way Visibly Pushdown Parikh Automata

Definition

A two-way Visibly Pushdown Parikh automaton $P = (A, dim, \lambda, S)$ where $A = (\Sigma, Q, I, F, \Gamma, \Delta)$ is an 2VPA, dim is a natural number, $\lambda : \Delta \mapsto \mathbb{N}^{dim}$ and S is a semi-linear subset of $\mathbb{N}^{dim}$.

Two-way Visibly Pushdown Parikh Automata

Definition

A two-way Visibly Pushdown Parikh automaton $P = (A, dim, \lambda, S)$ where $A = (\Sigma, Q, I, F, \Gamma, \Delta)$ is an 2VPA, dim is a natural number, $\lambda : \Delta \mapsto \mathbb{N}^{dim}$ and S is a semi-linear subset of $\mathbb{N}^{dim}$.

Reducing (D)2VPT problems to emptiness of (D)2VPPA

Well-nestedness of (D)2VPT

Globally as many calls as returns. Locally always more calls then returns.

$dim = 4$

$$q_1 \xrightarrow{\alpha,\pm\gamma|w} q_2 \;:\; \begin{cases} q_1 \xrightarrow{\alpha,\pm\gamma,(|w|_{\Sigma_c},|w|_{\Sigma_r},|w|_{\Sigma_c},|w|_{\Sigma_r})} q_2 \\ q_1 \xrightarrow{\alpha,\pm\gamma,(|w|_{\Sigma_c},|w|_{\Sigma_r},|w|_{\Sigma_c},|w|_{\Sigma_r})} q_2' \\ q_1' \xrightarrow{\alpha,\pm\gamma,(0,0,|w|_{\Sigma_c},|w|_{\Sigma_r})} q_2' \end{cases}$$

$$\varphi = x_3 \neq x_4 \vee x_1 < x_2$$

Not well-nested $\Leftrightarrow$ Not empty

Functionality of 2VPT

On the same input word, there exists an output position where two different letters are output in two different computations (essentially).

$dim = 2$

- Compute a position in the output

$$q_1 \xrightarrow{\alpha,\pm\gamma|w} q_2 \quad \Rightarrow q_1 \xrightarrow{\alpha,\pm\gamma,(|w|,0)} q_2$$

- Guess the (first half of the) mismatched

$$q_1 \xrightarrow{\alpha,\pm\gamma|w} q_2 \quad \Rightarrow q_1 \xrightarrow{\alpha,\pm\gamma,(|w_1|,0)} q_2^a \qquad w = w_1 a w_2$$

- Rewind the input word and start reading again

$$q_1 \xrightarrow{\alpha,\pm\gamma|w} q_2 \quad \Rightarrow q_1^a \xrightarrow{\alpha,\pm\gamma,(0,|w|)} q_2^a$$

- Guess the (second half of the) mismatched

$$q_1 \xrightarrow{\alpha,\pm\gamma|w} q_2 \quad \Rightarrow q_1^a \xrightarrow{\alpha,\pm\gamma,(0,|w_1|)} q_F \qquad w = w_1 b w_2 \ , \ a \neq b$$

$$\varphi = \ x_1 = x_2$$

Not functional $\Leftrightarrow$ Not empty

Some Existing Results about the emptiness problem

Emptiness for :

- deterministic two-way counter machines with 2 counters and k (fixed) counter-reversals is undecidable

Some Existing Results about the emptiness problem

Emptiness for :

- deterministic two-way counter machines with 2 counters and k (fixed) counter-reversals is undecidable
- (non-deterministic) two-way reversal-bounded counter machines with finite crossing is decidable

Some Existing Results about the emptiness problem

Emptiness for :

- deterministic two-way counter machines with 2 counters and k (fixed) counter-reversals is undecidable
- (non-deterministic) two-way reversal-bounded counter machines with finite crossing is decidable
- (non-deterministic) two-way pushdown automata with finite crossing is undecidable

Some Bad News

Theorem

Emptiness for deterministic two-way visibly pushdown Parikh automata is undecidable.

Some Bad News

Theorem

Emptiness for deterministic two-way visibly pushdown Parikh automata is undecidable.

Proof Idea :

Reduction of solvability of Diophantine Eq. : $P = Q$ with $P, Q \in \mathbb{N}[\mathscr{X}]$, eg $2xy + zz = 4x + 2xz + 6$

The automaton part encodes only of monomials; sums and equality test encoded in Presburger accepting formula, a dimension for each monomial : $2x_1 + x_2 = 4x_3 + 2x_4 + 6$.

Input nested words represent valuations of variables

$$\begin{array}{ll} \text{Encode}(x * y * 1, [x \mapsto 3, y \mapsto 2]) = & x_c c^3 \text{Encode}(y * 1, [x \mapsto 3, y \mapsto 2])\, r^3 x_r \\ \text{Encode}(1, [x \mapsto 3, y \mapsto 2]) = & \mathbb{1} \end{array}$$

The automaton checks well-formedness

- visibly pushdown "regular" sequence of monomials
- one dimension for each variable occurence in P and Q : test via the Presburger accepting formula that there are valuated the same way.

and evaluates monomials via counters :

The dimension updated as

$$[\![\text{Encode}(x_c c^3 \text{Encode}(y * 1, [x \mapsto 3, y \mapsto 2])\, r^3 x_r)]\!] = 3 * [\![\text{Encode}(y * 1, [x \mapsto 3, y \mapsto 2])]\!]$$

Some (Modest) Good News

Theorem

The non-emptiness problem for (one-way) VPPA *and* PPA *is* NP*-complete.*

Some (Modest) Good News

Theorem

The non-emptiness problem for (one-way) VPPA *and* PPA *is* NP*-complete.*

Upperbound : let V be the values used for update in the PPA A

New alphabet $\Sigma' = \Sigma \cup (u_v^i)$ (update counter i by adding value v ($v \in V$)).

Some (Modest) Good News

Theorem

The non-emptiness problem for (one-way) VPPA *and* PPA *is* NP*-complete.*

Upperbound : let V be the values used for update in the PPA A

New alphabet $\Sigma' = \Sigma \cup (u_v^i)$ (update counter i by adding value v ($v \in V$)).

- Step 1 : Define a NPA B accepting words of the form $\alpha_1 u_0^1 u_1^2 \alpha_2 u_3^1 u_2^2 \alpha_3 u_0^1 u_0^2 \alpha_4 u_0^1 u_1^2 \alpha_5 \ldots$ such that
  - $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\ldots$ has a run in A from an initial to a final configuration
  - each $u_{v_1}^1 u_{v_2}^2$ corresponds to the update performed by the run at that position : $(v_1, v_2) \Rightarrow u_{v_1}^1 u_{v_2}^2$
  - B is of polynomial size in A

Some (Modest) Good News

Theorem

The non-emptiness problem for (one-way) VPPA *and* PPA *is* NP*-complete.*

Upperbound : let V be the values used for update in the PPA A

New alphabet $\Sigma' = \Sigma \cup (u_v^i)$ (update counter i by adding value v ($v \in V$)).

- Step 1 : Define a NPA B accepting words of the form $\alpha_1 u_0^1 u_1^2 \alpha_2 u_3^1 u_2^2 \alpha_3 u_0^1 u_0^2 \alpha_4 u_0^1 u_1^2 \alpha_5 \ldots$ such that
  - $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\ldots$ has a run in A from an initial to a final configuration
  - each $u_{v_1}^1 u_{v_2}^2$ corresponds to the update performed by the run at that position : $(v_1, v_2) \Rightarrow u_{v_1}^1 u_{v_2}^2$
  - B is of polynomial size in A
- Step 2 : There is an existential Presburger formula $\psi((y_\sigma)_{\sigma\in\Sigma'})$ of size polynomial in B defining the Parikh image of $L(B)$. [Verma, et al. 05]

Some (Modest) Good News

Theorem

The non-emptiness problem for (one-way) VPPA *and* PPA *is* NP*-complete.*

Upperbound : let V be the values used for update in the PPA A

New alphabet $\Sigma' = \Sigma \cup (u_v^i)$ (update counter i by adding value v $(v \in V)$).

- Step 1 : Define a NPA B accepting words of the form $\alpha_1 u_0^1 u_1^2 \alpha_2 u_3^1 u_2^2 \alpha_3 u_0^1 u_0^2 \alpha_4 u_0^1 u_1^2 \alpha_5 \ldots$ such that
  - $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\ldots$ has a run in A from an initial to a final configuration
  - each $u_{v_1}^1 u_{v_2}^2$ corresponds to the update performed by the run at that position : $(v_1, v_2) \Rightarrow u_{v_1}^1 u_{v_2}^2$
  - B is of polynomial size in A
- Step 2 : There is an existential Presburger formula $\psi((y_\sigma)_{\sigma\in\Sigma'})$ of size polynomial in B defining the Parikh image of $L(B)$. [Verma, et al. 05]
- Step 3 : In NP [Scarpellini84], satisfiability test of

$$\psi((y_\sigma)_{\sigma\in\Sigma'}) \wedge \phi(x_1,\ldots,x_n) \wedge \bigwedge_{1\le i\le dim} x_i = \sum_{v\in V} v.y_{u_v^i}$$

Some (Modest) Good News

Lower bound : (hardness does not rely on the acceptance set, eg Presburger satisfiability).

Some (Modest) Good News

Lower bound : (hardness does not rely on the acceptance set, eg Presburger satisfiability).

Reduction of the 2-partition problem :

- I a finite set of natural numbers represented in binary.
- Does there exists $J \subseteq I$ such that

$$\sum_{e \in J} e = \sum_{e \notin J} e$$

Some (Modest) Good News

Lower bound : (hardness does not rely on the acceptance set, eg Presburger satisfiability).

Reduction of the 2-partition problem :

- I a finite set of natural numbers represented in binary.
- Does there exists $J \subseteq I$ such that

$$\sum_{e \in J} e = \sum_{e \notin J} e$$

One dimension for J and one for $I \setminus J$.

Input words : $T_1 v_1 L_2 v_2 T_3 v_3 T_4 v_4 L_5 v_5 \ldots$

- v_k : representation of the unary encoding of the kth natural.
- T_i : take the ith natural (in J) - L_j : leave the jth natural

v_k is encoded as $u_n u_{n-1} \ldots u_1 u_0$ where

- $u_j = \epsilon$ if the jth bit of v_k is 0 and w_j otherwise
- recursively, $w_0 = 1$ and $w_l = c_l w_{l-1} r_l c_l w_{l-1} r_l$

and can be recognized by a small VPA.

Some (Modest) Good News

Lower bound : (hardness does not rely on the acceptance set, eg Presburger satisfiability).

Reduction of the 2-partition problem :

- I a finite set of natural numbers represented in binary.
- Does there exists $J \subseteq I$ such that

$$\sum_{e \in J} e = \sum_{e \notin J} e$$

One dimension for J and one for $I \setminus J$.

Input words : $T_1 v_1 L_2 v_2 T_3 v_3 T_4 v_4 L_5 v_5 \ldots$

- v_k : representation of the unary encoding of the kth natural.
- T_i : take the ith natural (in J) - L_j : leave the jth natural

v_k is encoded as $u_n u_{n-1} \ldots u_1 u_0$ where

- $u_j = \epsilon$ if the jth bit of v_k is 0 and w_j otherwise
- recursively, $w_0 = 1$ and $w_l = c_l w_{l-1} r_l c_l w_{l-1} r_l$

and can be recognized by a small VPA.

Thus, holds even if the automata are deterministic, with a fixed dimension 2, tuples of values in $\{0,1\}^2$ and with a fixed Presburger formula $(x_1, x_2) = x_1 = x_2$

Emptiness for single-use 2VPPA

Single-use 2VPPA : a position is visited only once per counter modifying transition.

Emptiness for single-use 2VPPA

Single-use 2VPPA : a position is visited only once per counter modifying transition.

Proposition

For any single-use 2VPPA, *there exists an equivalent* VPPA *of at most exponential size.*

Emptiness for single-use 2VPPA

Single-use 2VPPA : a position is visited only once per counter modifying transition.

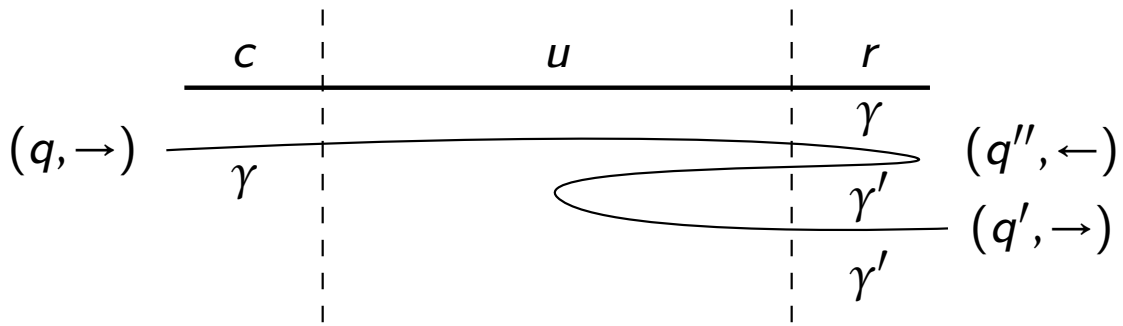
Proposition

For any single-use 2VPPA, *there exists an equivalent* VPPA *of at most exponential size.*

Proof Idea

- Extends [Dartois, Filiot, Reynier, T. 16] following Sherpherson's ideas (for FSA) on traversals T.

$((q,d)(q',d')) \in T(u)$ iff some run enters u by (q,d) and leaves it by (q',d')



Emptiness for single-use 2VPPA

Single-use 2VPPA : a position is visited only once per counter modifying transition.

Proposition

For any single-use 2VPPA, *there exists an equivalent* VPPA *of at most exponential size.*

Proof Idea

- Extends [Dartois, Filiot, Reynier, T. 16] following Sherpherson's ideas (for FSA) on traversals T.

$$((q,d)(q',d')) \in T(u) \text{ iff some run enters } u \text{ by } (q,d) \text{ and leaves it by } (q',d')$$

- Productive and Non-productive traversals form a finite algebra.
- Consider possible decompositions of traversals $((q,\rightarrow)(q',\leftarrow))$ on $c_1 r w_2$ into productive traversals (finitely many as single-use) reachable from each other by non-productive traversals.
- Using commutativity of addition over $\mathbb{N}^{dim}$, extract a VPA and a mapping λ

Emptiness for single-use 2VPPA

Single-use 2VPPA : a position is visited only once per counter modifying transition.

Proposition

For any single-use 2VPPA, *there exists an equivalent* VPPA *of at most exponential size.*

Corollary

Emptiness for single-use 2VPPA *is in* NEXP*time.*

Theorem

Emptiness for single-use 2VPPA *is in* NEXP*time-complete.*

Implied Results

For well-nestedness and functionality single-use 2VPT yields single-use 2VPPA

Proposition

For single-use 2VPT*, well-nestedness and functionality are in* NEXP*time.*

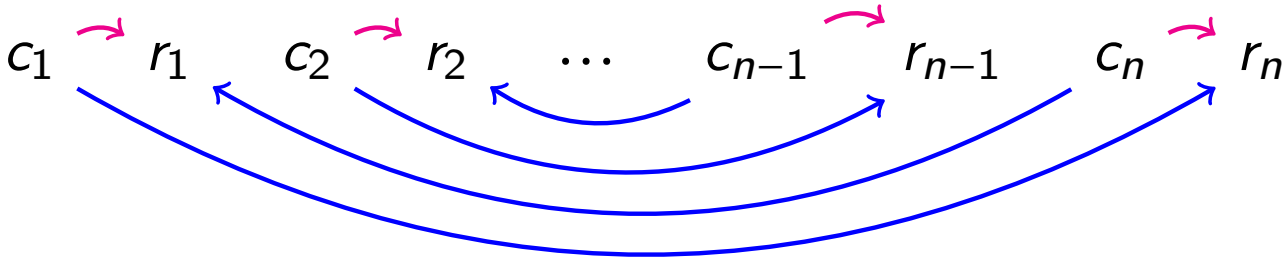
Both known as EXPtime-hard.

Back to tree-to-tree transformations

Consider the D2VPT$_{su}$ defining the transformation with well-nested outputs

$$c_1 r_1 c_2 r_2 \dots c_n r_n \mapsto c_1 c_2 \dots c_n r_1 r_2 \dots r_n$$

c_i and r_{n-i+1} match each other.

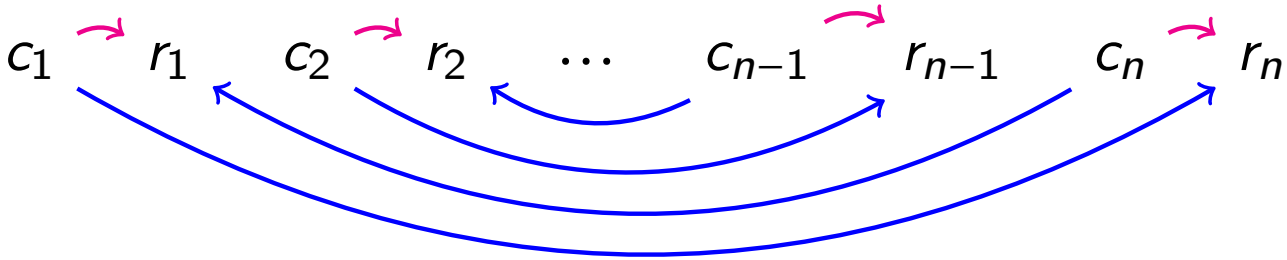


Back to tree-to-tree transformations

Consider the D2VPT$_{su}$ defining the transformation with well-nested outputs

$$c_1 r_1 c_2 r_2 \dots c_n r_n \mapsto c_1 c_2 \dots c_n r_1 r_2 \dots r_n$$

c_i and r_{n-i+1} match each other.



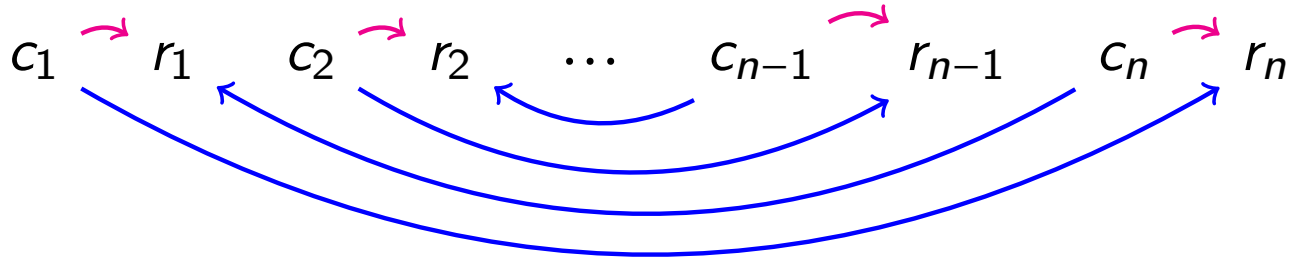
This matching relation is not MSO-definable

Back to tree-to-tree transformations

Consider the $\mathrm{D2VPT}_{\mathrm{su}}$ defining the transformation with well-nested outputs

$$c_1 r_1 c_2 r_2 \dots c_n r_n \mapsto c_1 c_2 \dots c_n r_1 r_2 \dots r_n$$

c_i and r_{n-i+1} match each other.



This matching relation is not MSO-definable

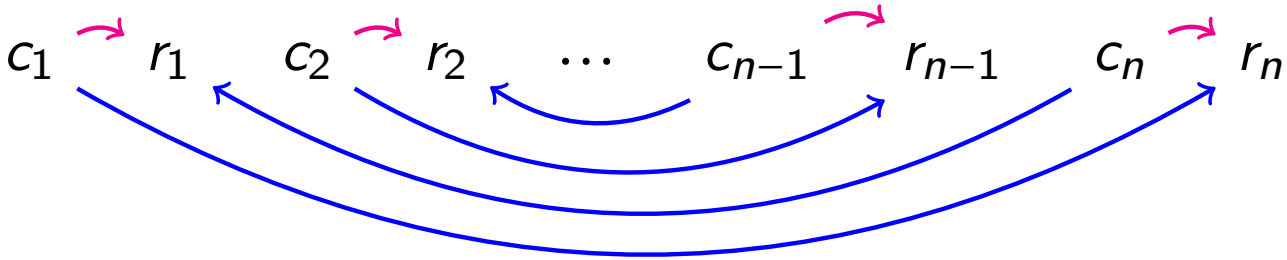
MSO[nw2nw] is strictly included into $\mathrm{MSO[nw2w]} \cap (\Sigma^*_{\mathrm{wn}} \to \Sigma^*_{\mathrm{wn}})$

Back to tree-to-tree transformations

Consider the D2VPT$_{su}$ defining the transformation with well-nested outputs

$$c_1 r_1 c_2 r_2 \dots c_n r_n \mapsto c_1 c_2 \dots c_n r_1 r_2 \dots r_n$$

c_i and r_{n-i+1} match each other.



This matching relation is not MSO-definable

MSO[nw2nw] is strictly included into MSO[nw2w] $\cap (\Sigma^*_{wn} \to \Sigma^*_{wn})$

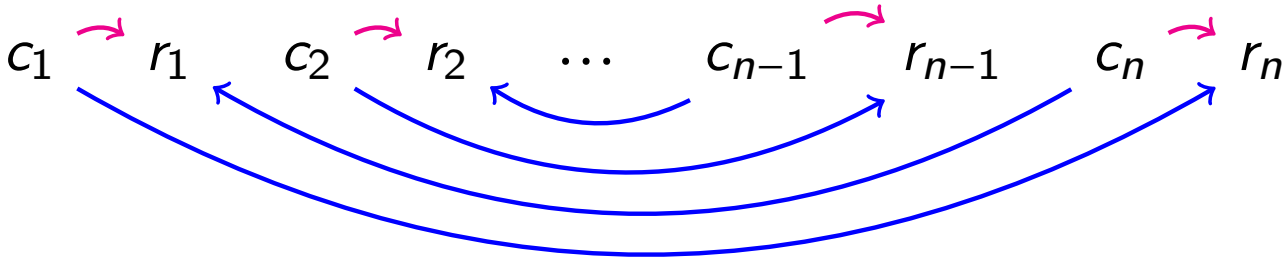
MSO[nw2w] $\cap (\Sigma^*_{wn} \to \Sigma^*_{wn})$ is not closed by composition

Back to tree-to-tree transformations

Consider the D2VPT$_{su}$ defining the transformation with well-nested outputs

$$c_1 r_1 c_2 r_2 \dots c_n r_n \mapsto c_1 c_2 \dots c_n r_1 r_2 \dots r_n$$

c_i and r_{n-i+1} match each other.



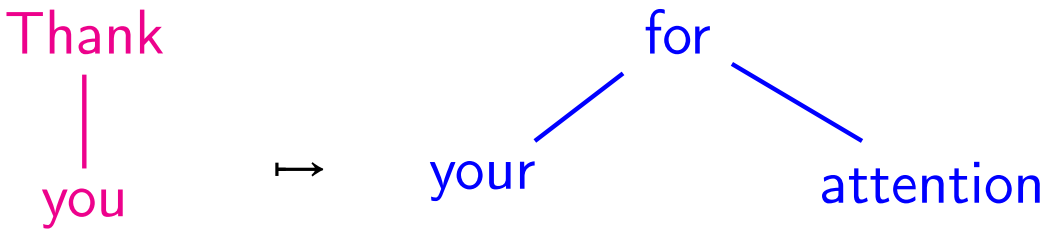
This matching relation is not MSO-definable

MSO[nw2nw] is strictly included into MSO[nw2w] $\cap (\Sigma^*_{wn} \to \Sigma^*_{wn})$

MSO[nw2w] $\cap (\Sigma^*_{wn} \to \Sigma^*_{wn})$ is not closed by composition

A new question :

Decide MSO[nw2nw] amongst MSO[nw2w] $\cap (\Sigma^*_{wn} \to \Sigma^*_{wn})$?


Thank
you
for
your
attention