Realizability of Concurrent Recursive Programs

Benedikt Bollig

LSV, ENS Cachan, CNRS
France

joint work with Manuela-Lidia Grindei and Peter Habermehl

Workshop on Automata, Concurrency, and Timed Systems
CMI, Chennai, January 2009

Concurrent recursive programs

Analysis

- amounts to verifying multi-stack pushdown systems
- abstraction of unrestricted systems
  - overapproximation [BET'03]
  - underapproximation [QR'05]
- restricting the degree of synchronization and parallelism [SV'06]

Synthesis

- language and automata theoretic framework needed
- generalization of asynchronous automata and Mazurkiewicz traces

---

[BET'03] Bouajjani & Esparza & Touili. A generic approach to the static analysis of concurrent programs with procedures. 2003.

[QR'05] Qadeer & Rehof. Context-bounded model checking of concurrent software. 2005.

[SV'06] Sen & Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. 2006.

Concurrent recursive programs

Analysis

- amounts to verifying multi-stack pushdown systems
- abstraction of unrestricted systems
  - overapproximation [BET'03]
  - underapproximation [QR'05]
- restricting the degree of synchronization and parallelism [SV'06]

Synthesis

- language and automata theoretic framework needed
- generalization of asynchronous automata and Mazurkiewicz traces

---

[BET'03] Bouajjani & Esparza & Touili. A generic approach to the static analysis of concurrent programs with procedures. 2003.

[QR'05] Qadeer & Rehof. Context-bounded model checking of concurrent software. 2005.

[SV'06] Sen & Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. 2006.

Outline

- Example of a concurrent recursive program
- Model of concurrent recursive programs:
  Concurrent visibly pushdown automata (CVPA)
- Specifications: Multi-stack visibly pushdown automata (MVPA)
- Synthesis of CVPA from MVPA
- A decidable criterion for realizability of bounded-phase specifications
- An MSO characterization of bounded-phase CVPA

Example of a concurrent recursive program

```
p(int x)
1 wait(turn=0); turn:=1;
2 if x>0 then return f(x,p(x-1));
3         else return x;
```

```
q(int y)
1 wait(turn=1); turn:=0;
2 if y>0 then return g(y,q(y-1));
3         else return y;
```

Example of a concurrent recursive program

```
p(int x)
1 wait(turn=0); turn:=1;
2 if x>0 then return f(x,p(x-1));
3         else return x;
```

```
q(int y)
1 wait(turn=1); turn:=0;
2 if y>0 then return g(y,q(y-1));
3         else return y;
```

$\Sigma_p = \{c_1, r_1, t_1, t_2\}$

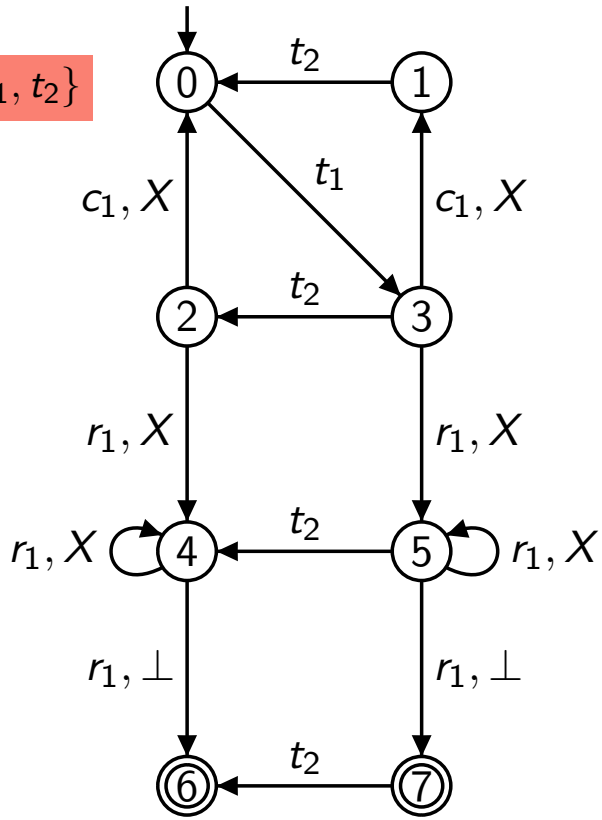
$\Sigma_q = \{c_2, r_2, t_1, t_2\}$

Example of a concurrent recursive program

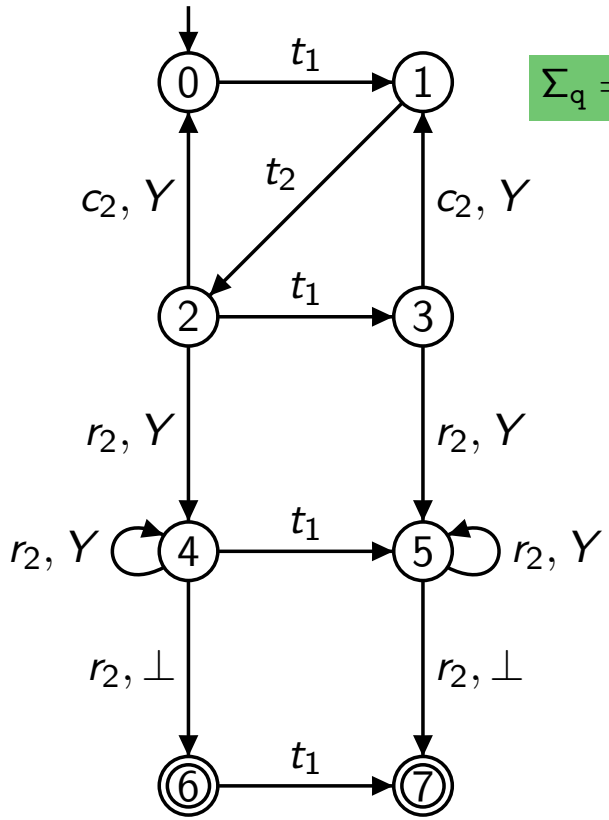
```
p(int x)
1 wait(turn=0); turn:=1;
2 if x>0 then return f(x,p(x-1));
3         else return x;
```

```
q(int y)
1 wait(turn=1); turn:=0;
2 if y>0 then return g(y,q(y-1));
3         else return y;
```

$\Sigma_p = \{c_1, r_1, t_1, t_2\}$



$\Sigma_q = \{c_2, r_2, t_1, t_2\}$

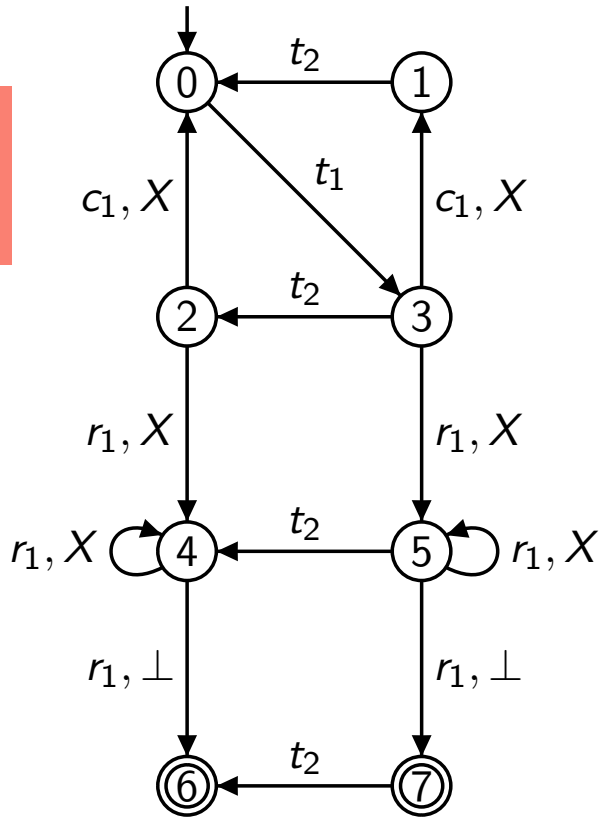


Example of a concurrent recursive program

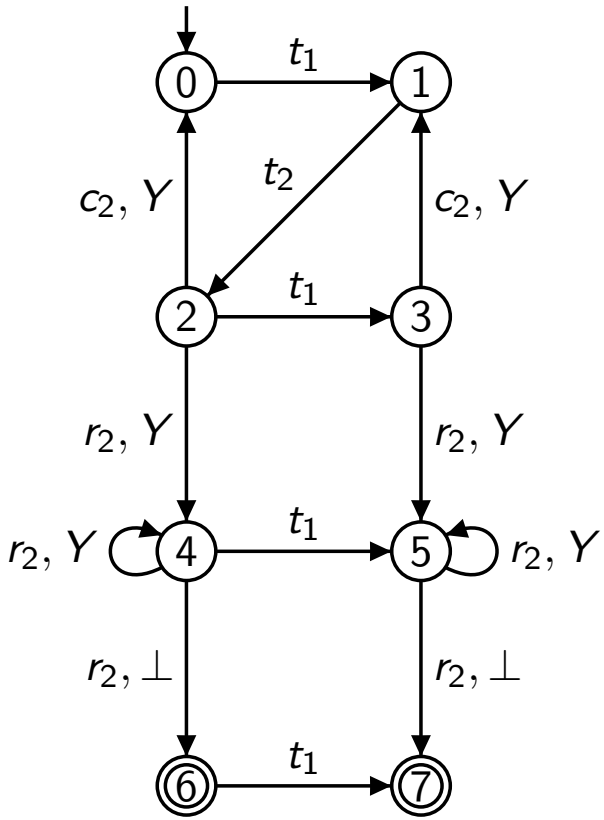
```
p(int x)
1 wait(turn=0); turn:=1;
2 if x>0 then return f(x,p(x-1));
3         else return x;
```

```
q(int y)
1 wait(turn=1); turn:=0;
2 if y>0 then return g(y,q(y-1));
3         else return y;
```

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$



$\texttt{turn} = 0$ $\texttt{turn} = 1$

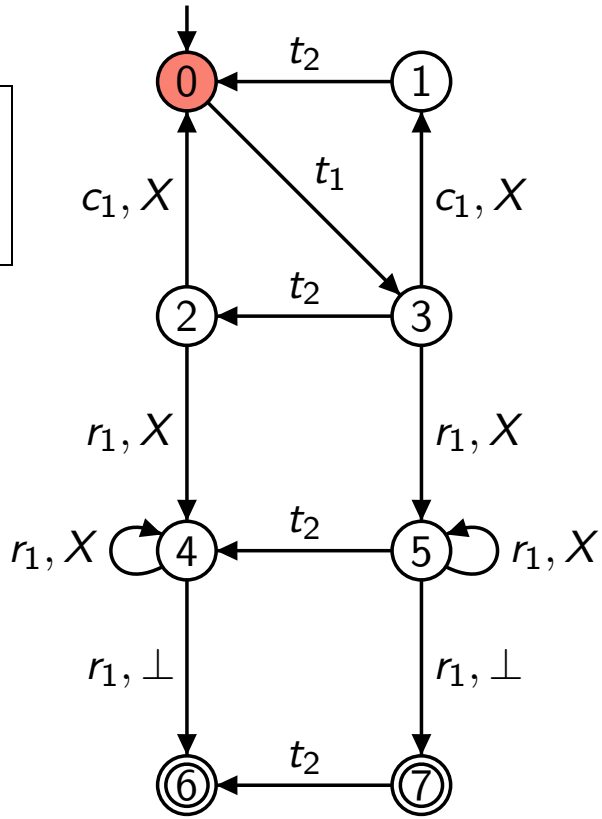


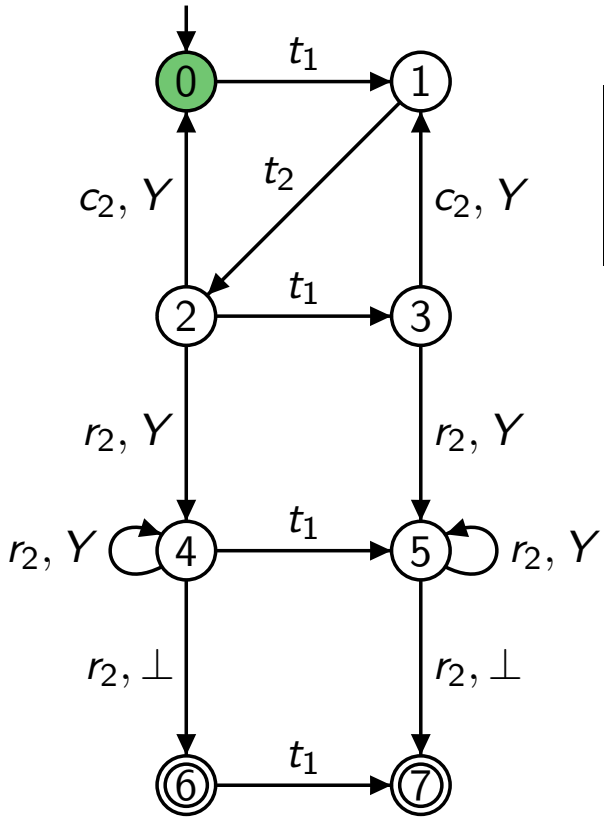
$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

$\texttt{turn} = 0$ $\texttt{turn} = 1$

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

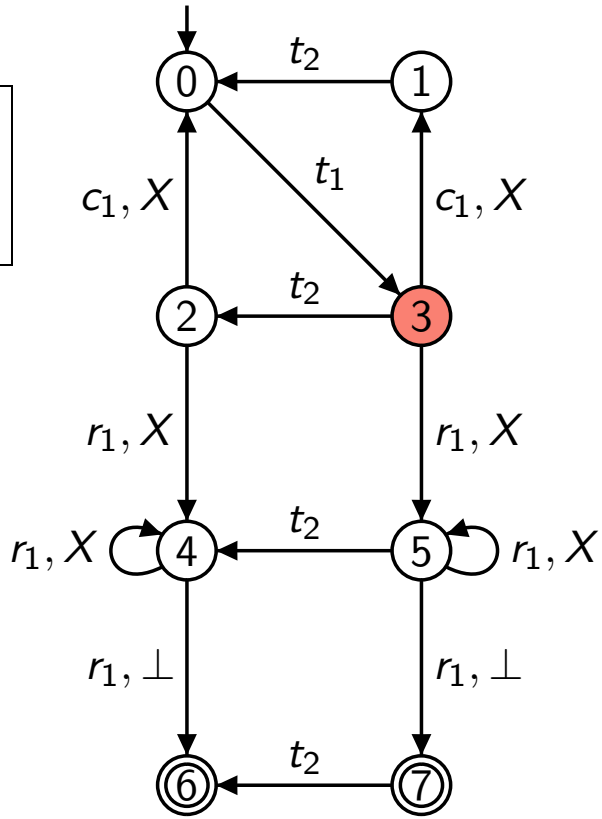
$\bot$

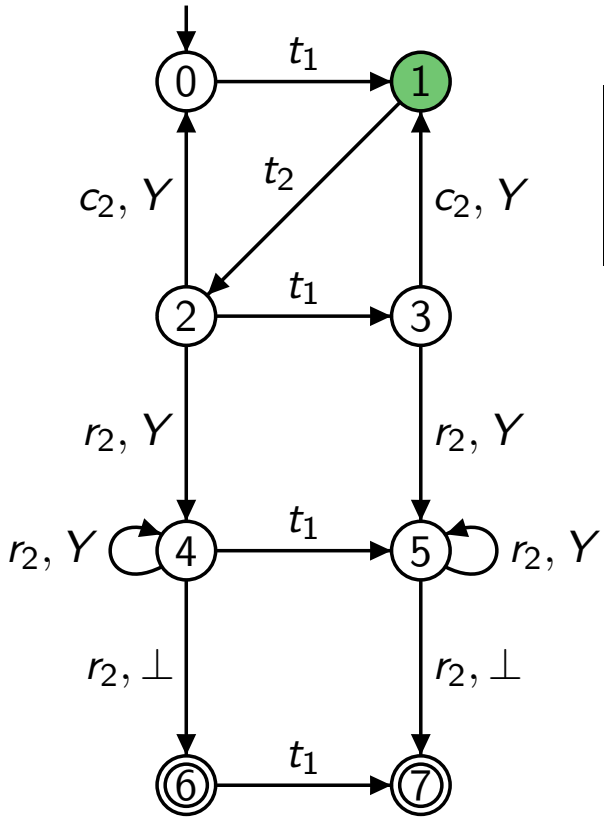
$\bot$

t_1

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

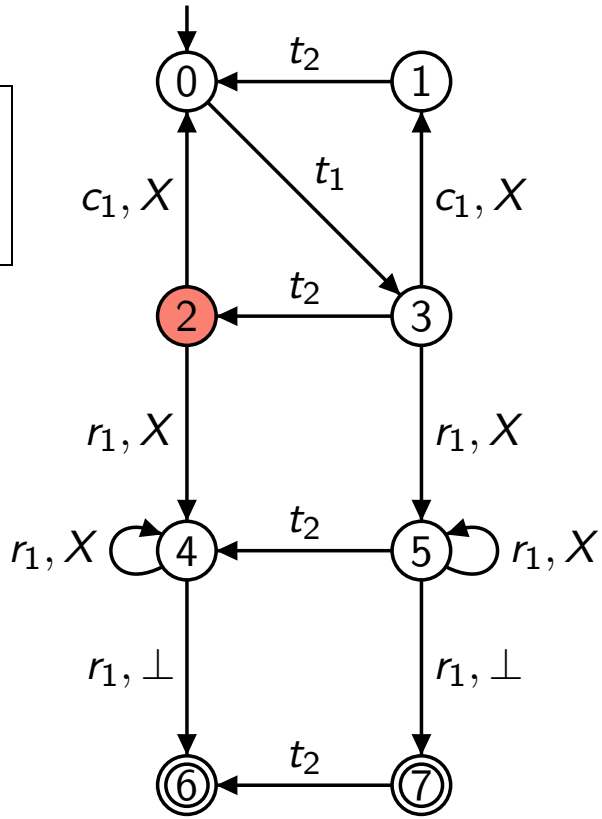
$\bot$

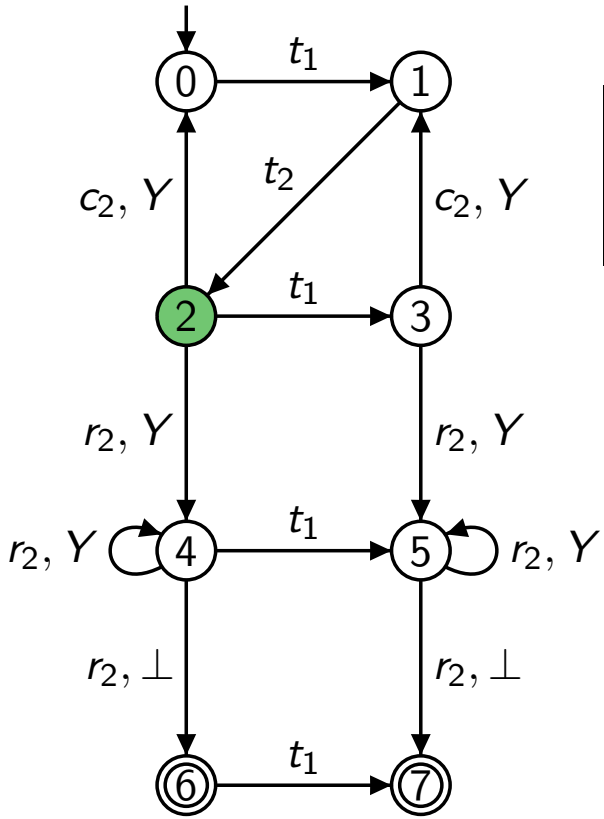
$\bot$

t_1 t_2

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

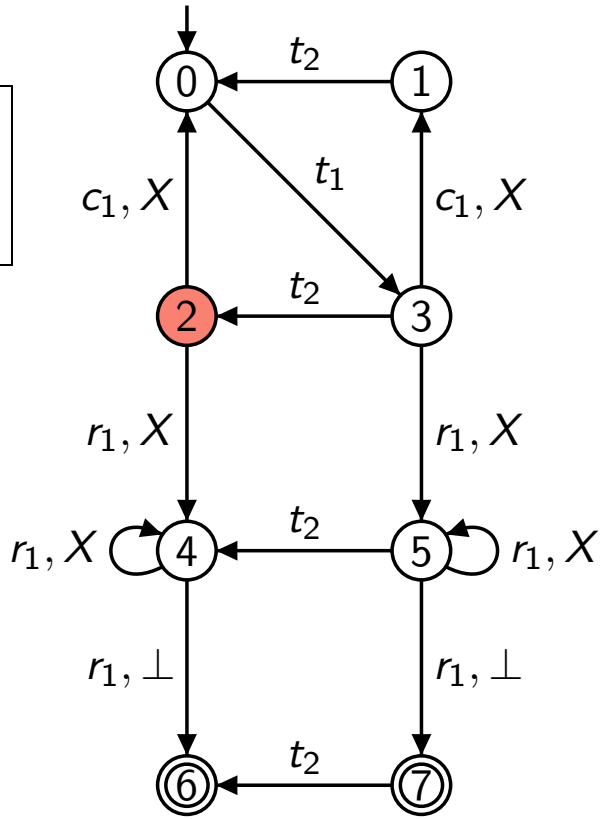
$\bot$

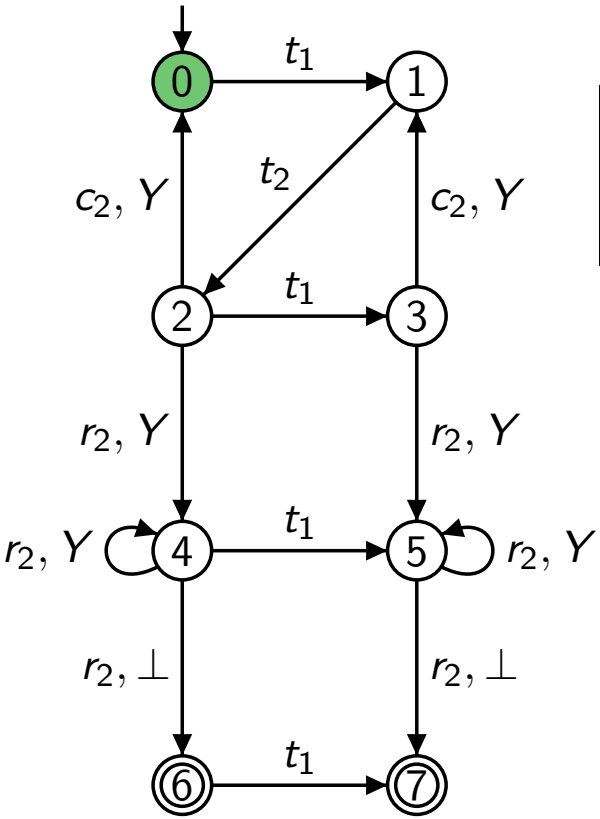
$\bot$

t_1 t_2 c_2

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

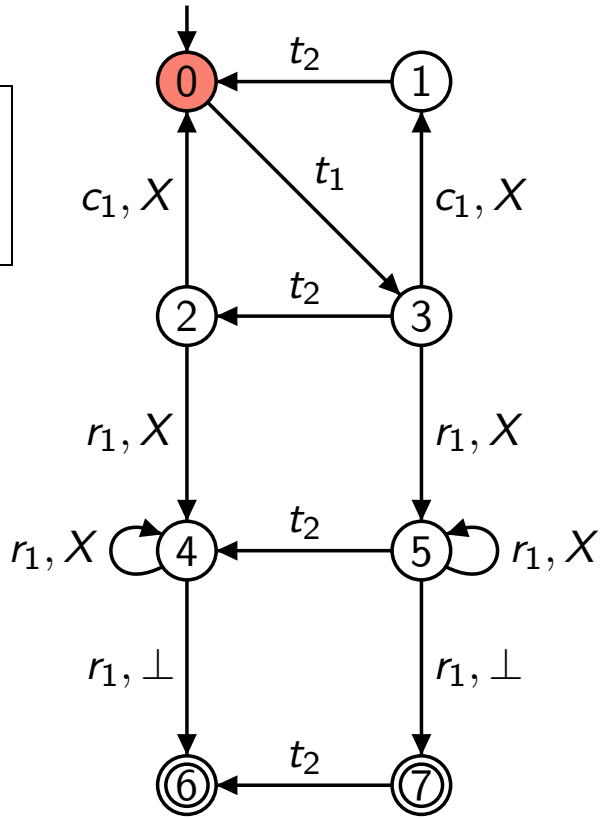
$\bot$

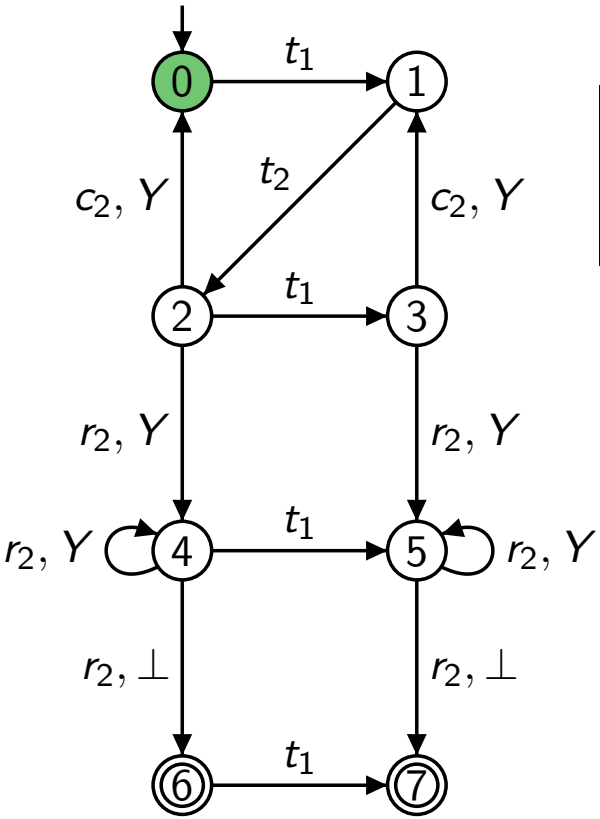
Y
$\bot$

t_1	t_2	c_2	c_1

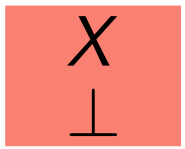
Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$



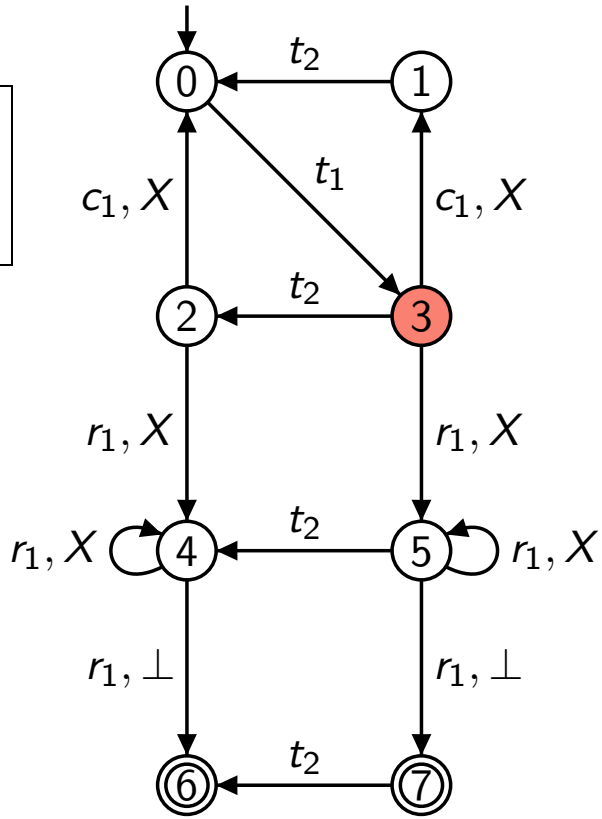


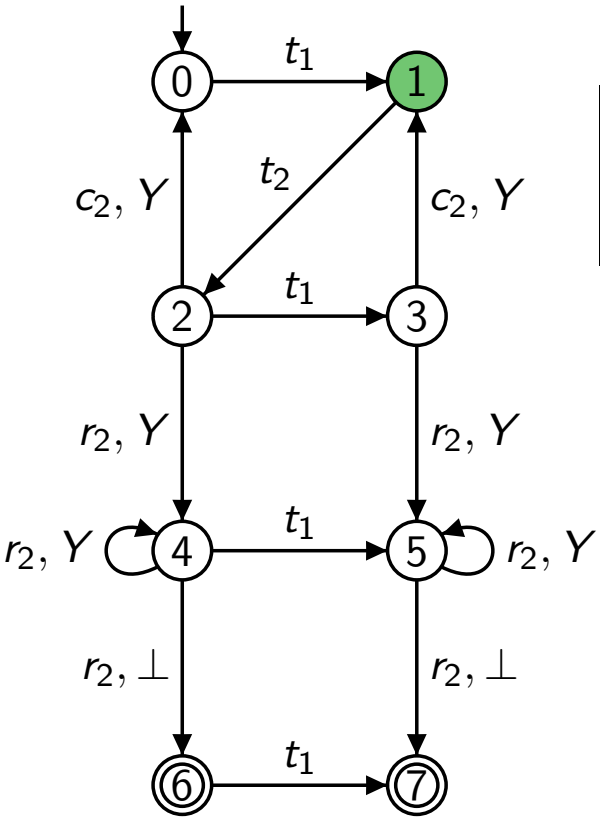
$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$



Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

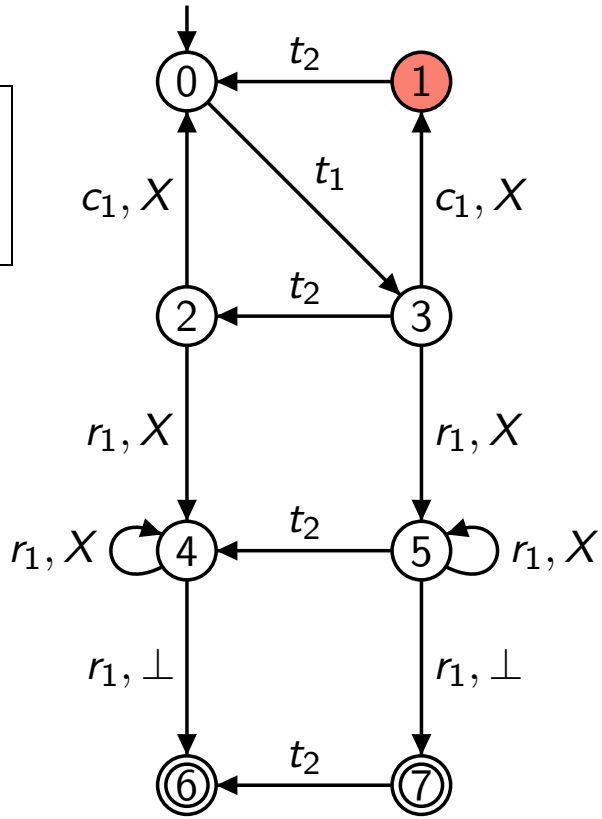
X
$\bot$

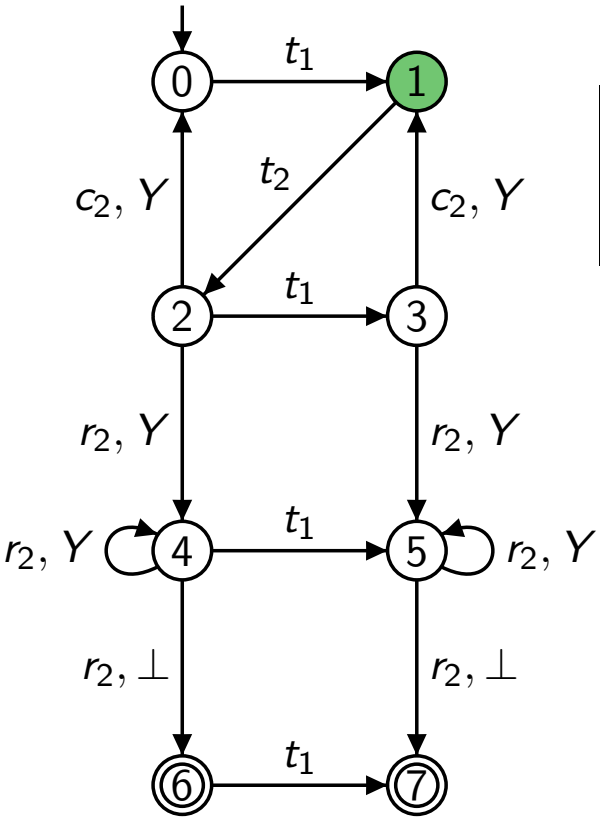
Y
$\bot$

t_1	t_2	c_2	c_1	t_1	c_1

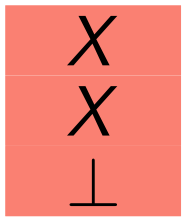
Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$



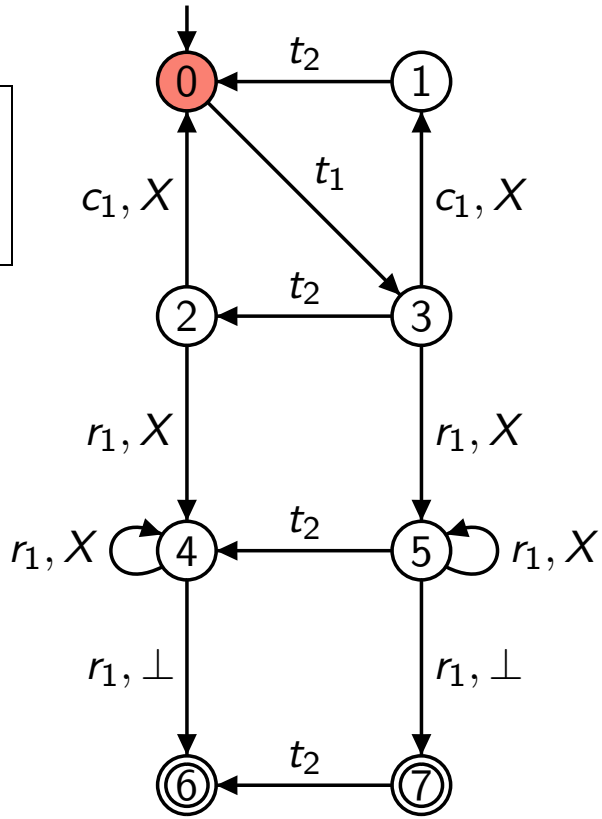


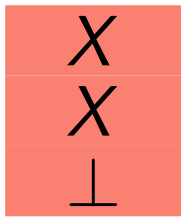
$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

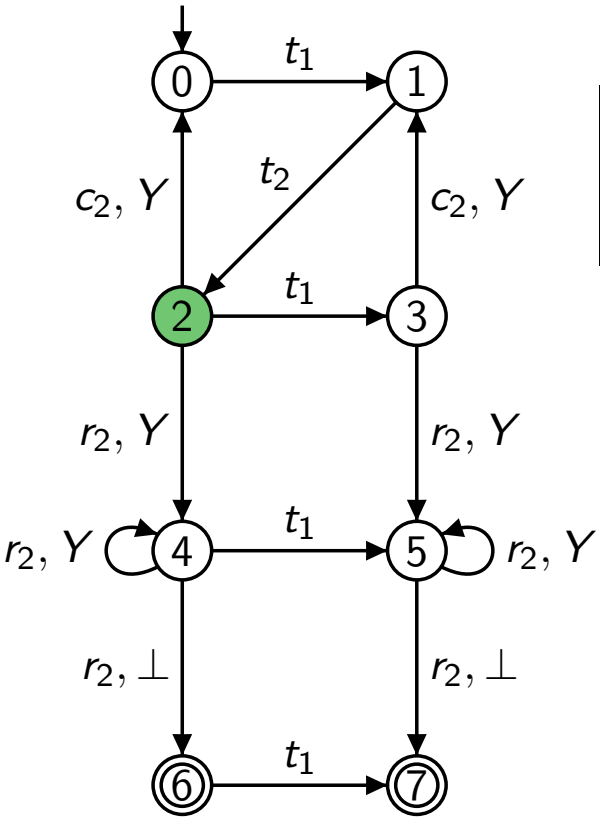


Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$



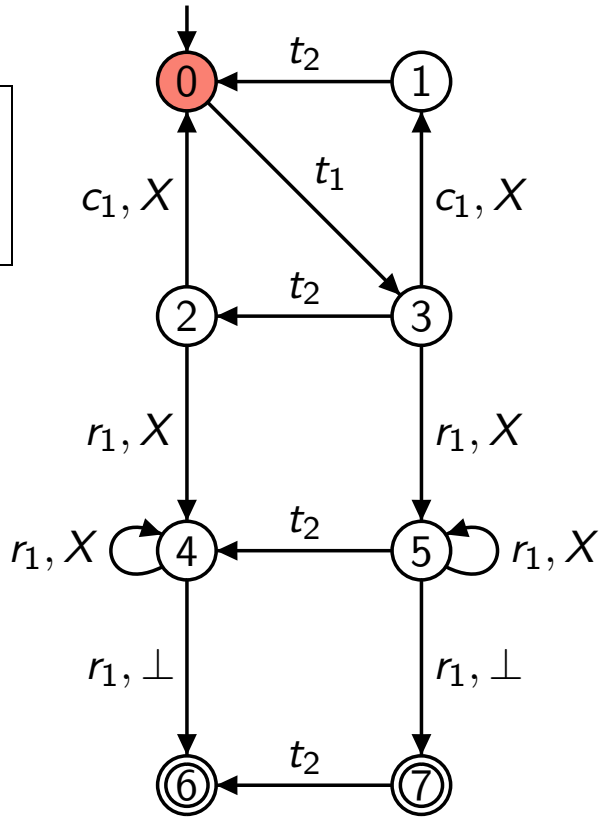


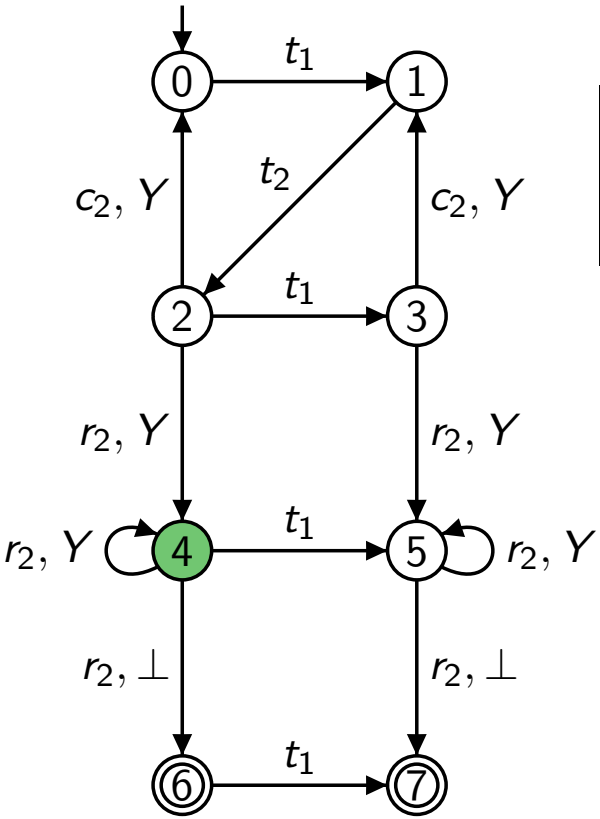


$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

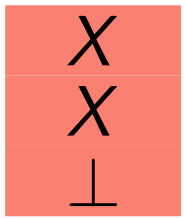
Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$



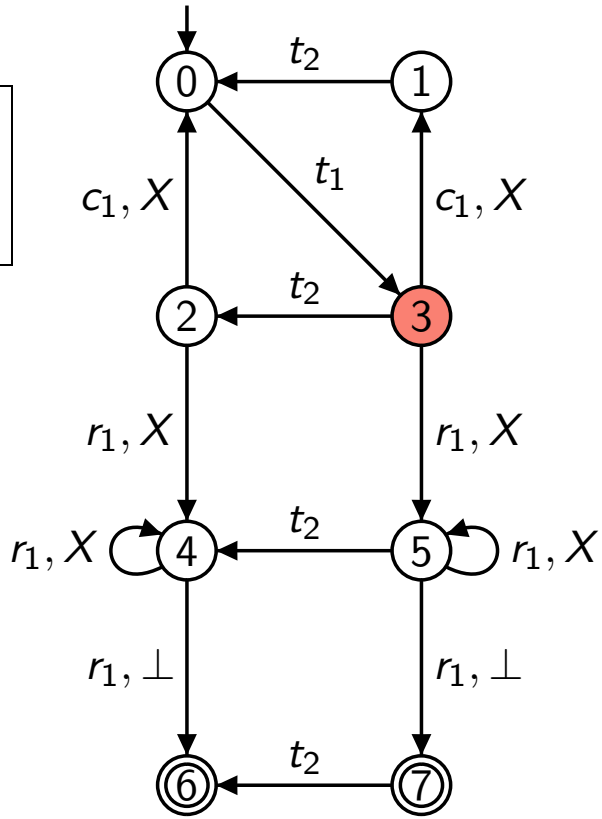


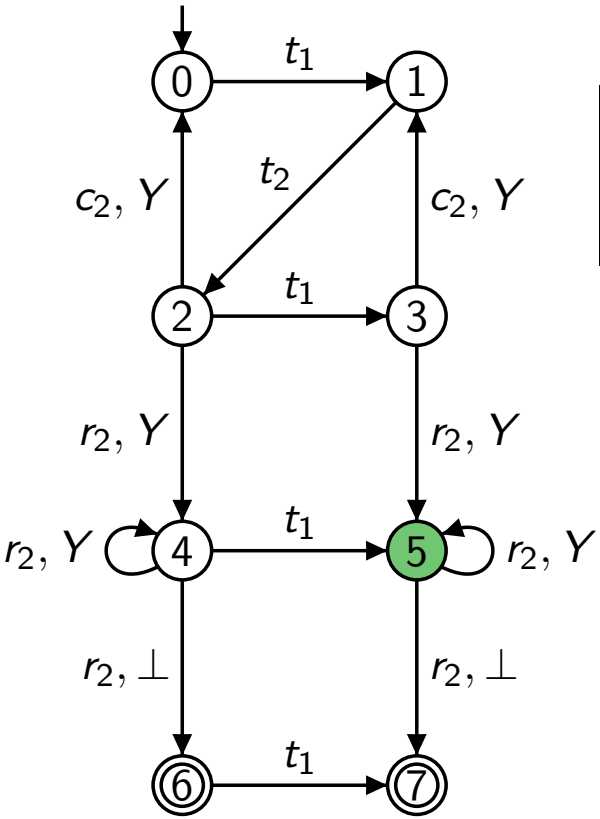
$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$



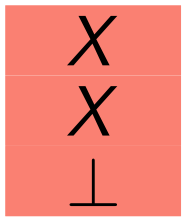
Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$



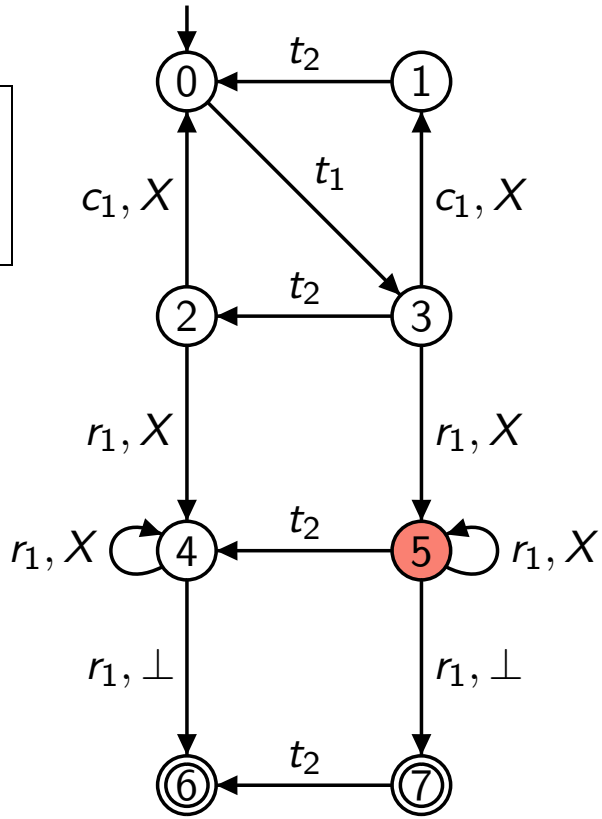


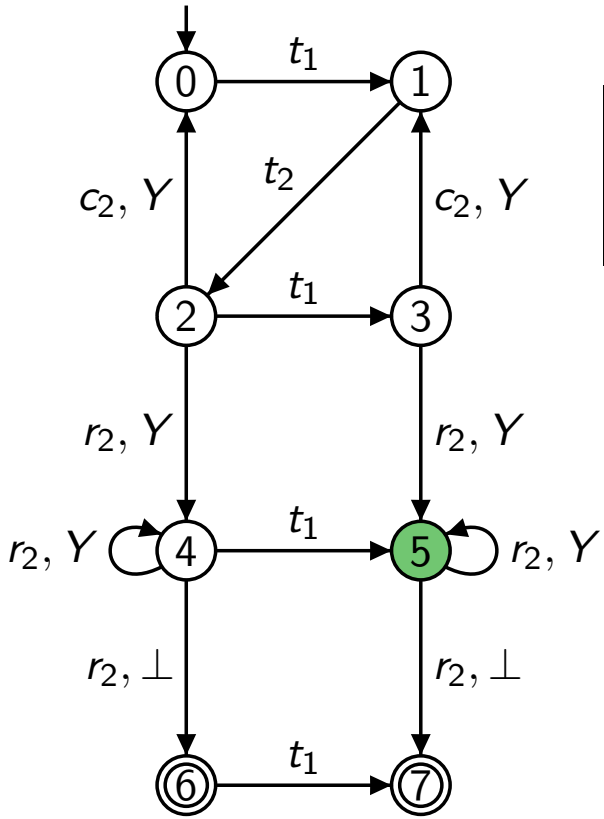
$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$



Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

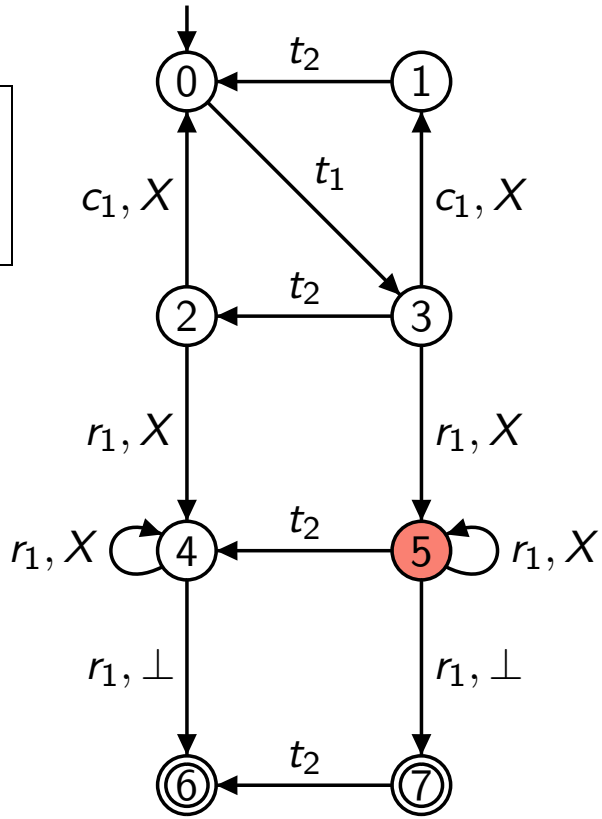
X
$\bot$

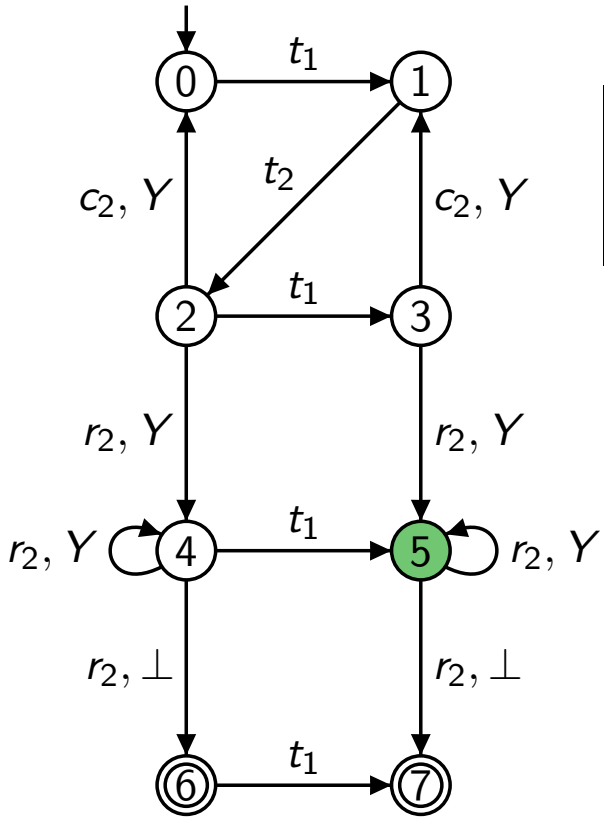
$\bot$

t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

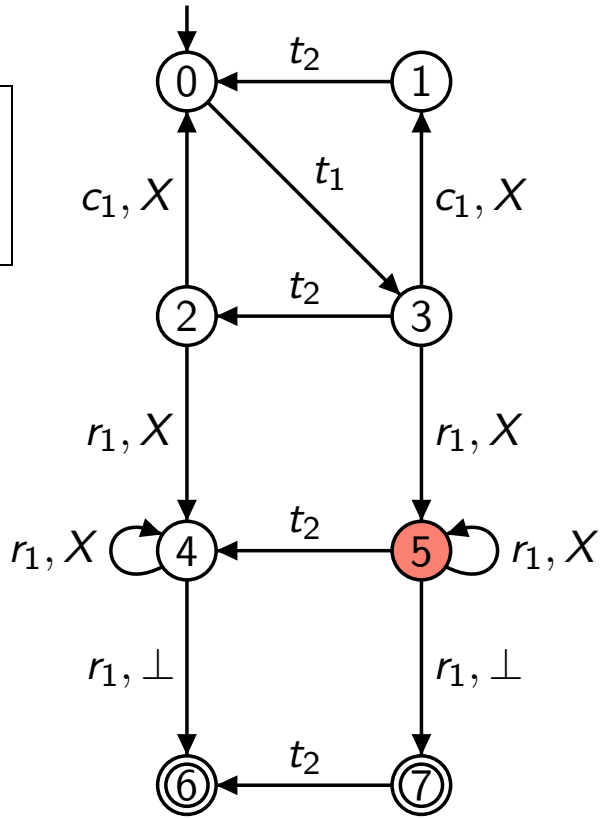
$\bot$

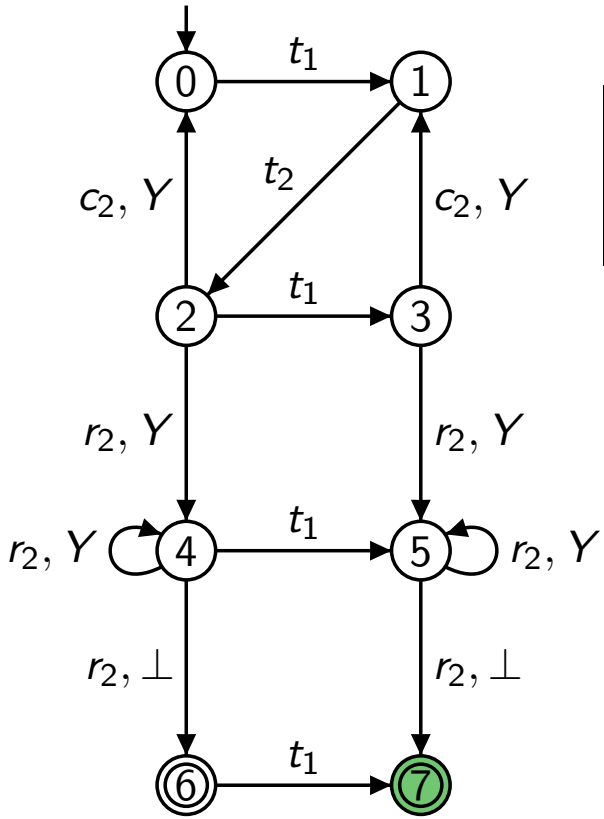
$\bot$

t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

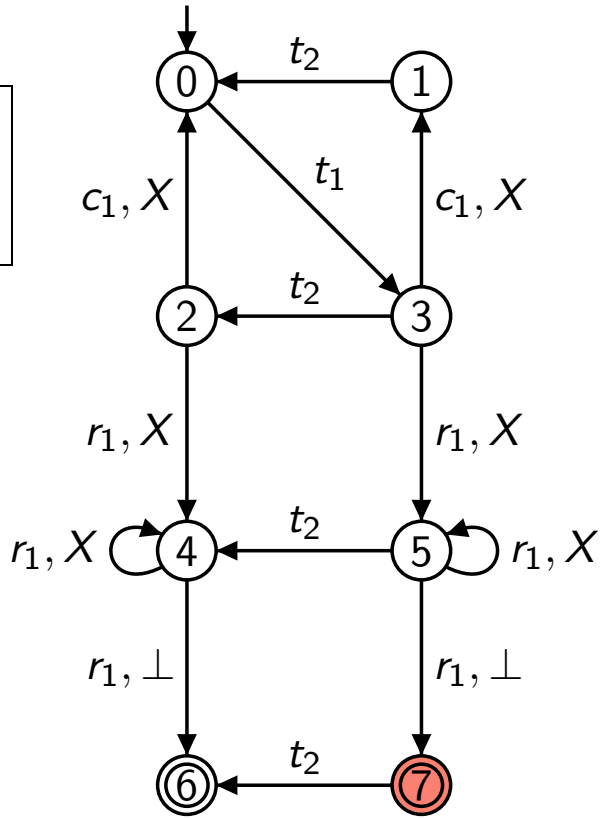
$\bot$

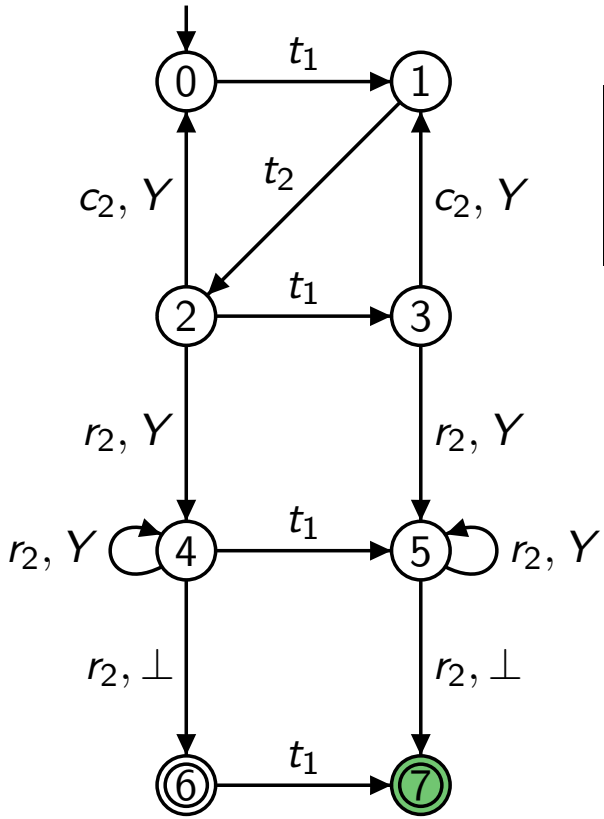
$\bot$

t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

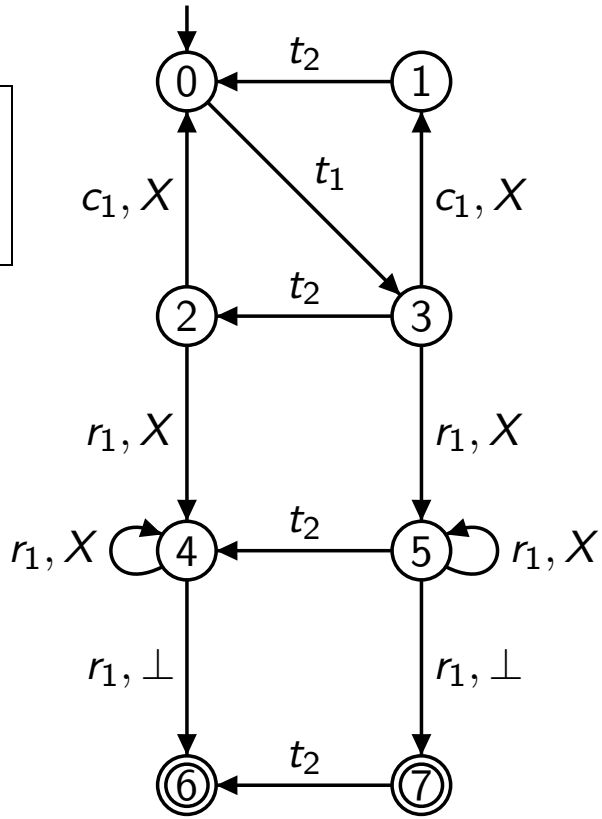
$\bot$

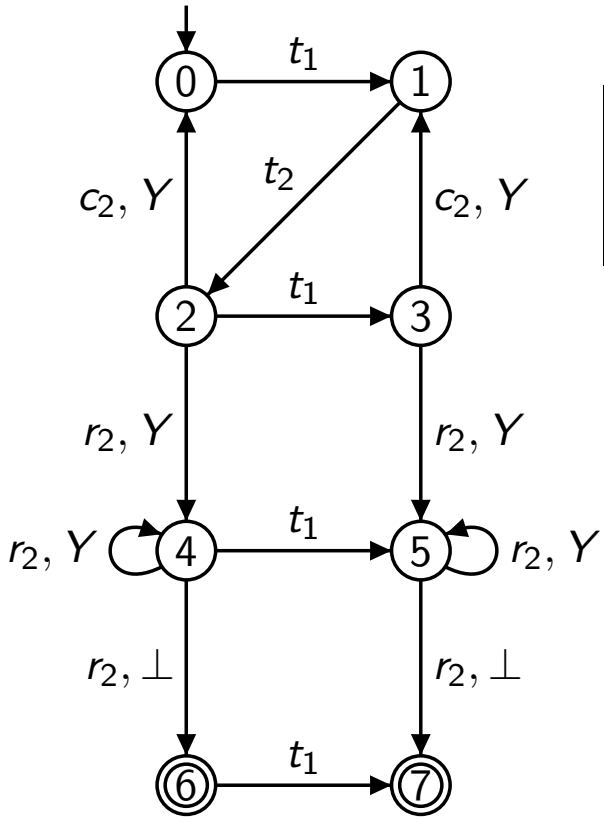
$\bot$

t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

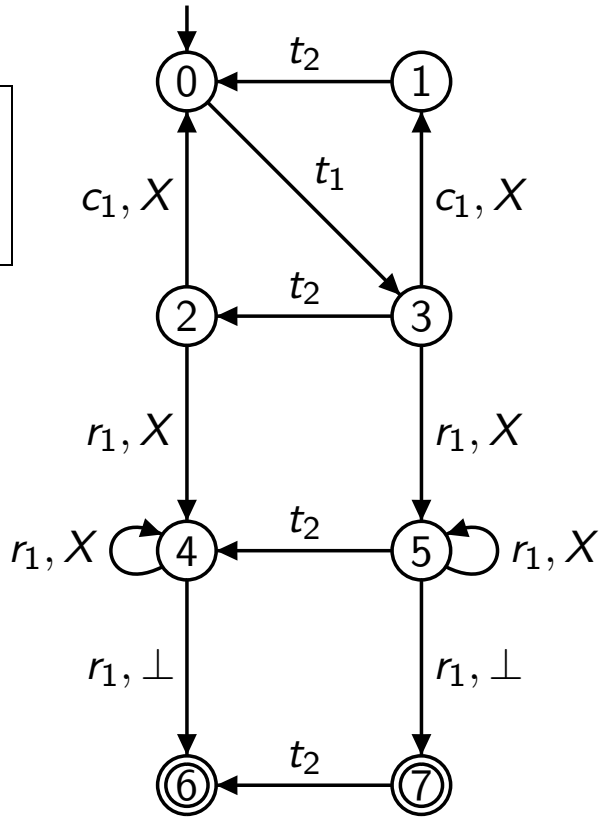
$\bot$

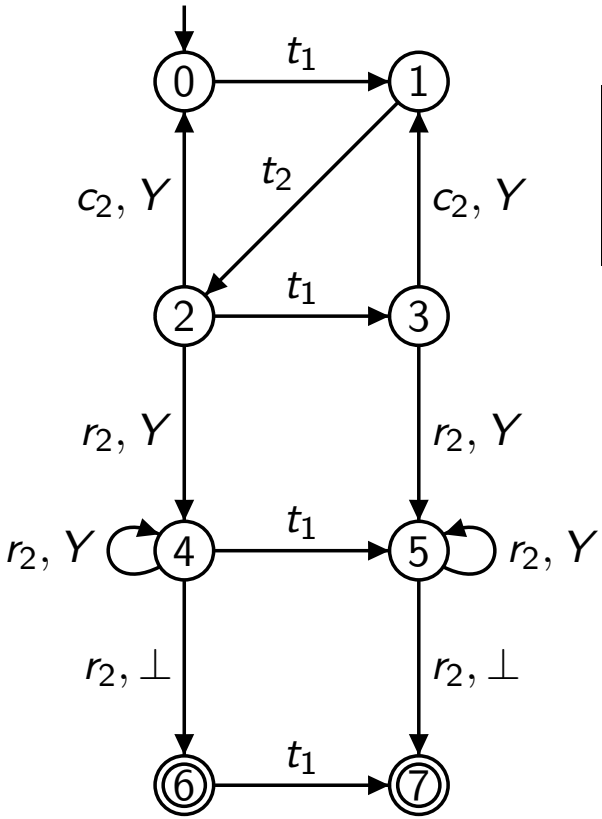
$\bot$

	t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1
$\sim$	t_1	t_2	c_1	c_2	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





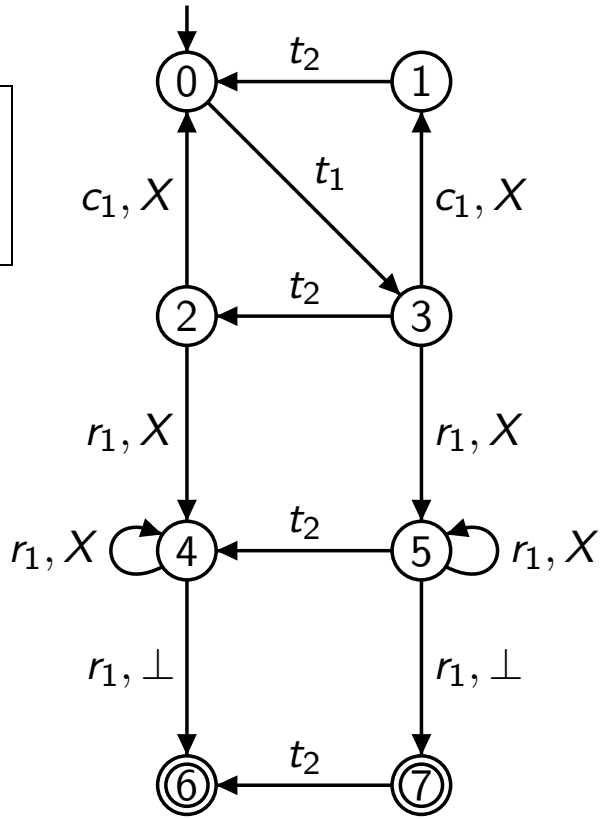
$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

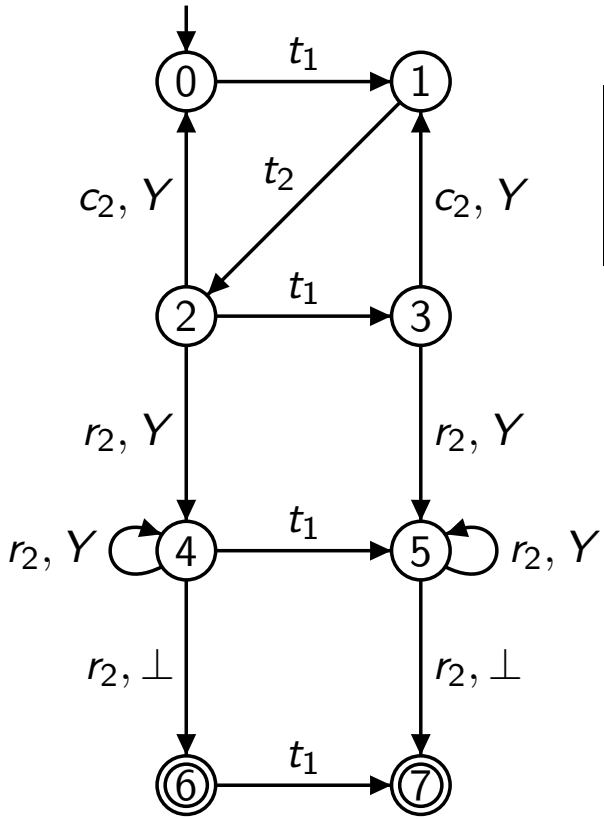
$\bot$ $\bot$

	t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1
$\sim$	t_1	t_2	c_1	c_2	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1
$\sim$	t_1	t_2	c_1	c_2	t_1	c_1	t_2	r_2	t_1	r_2	r_1	r_1	r_1

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

$\bot$

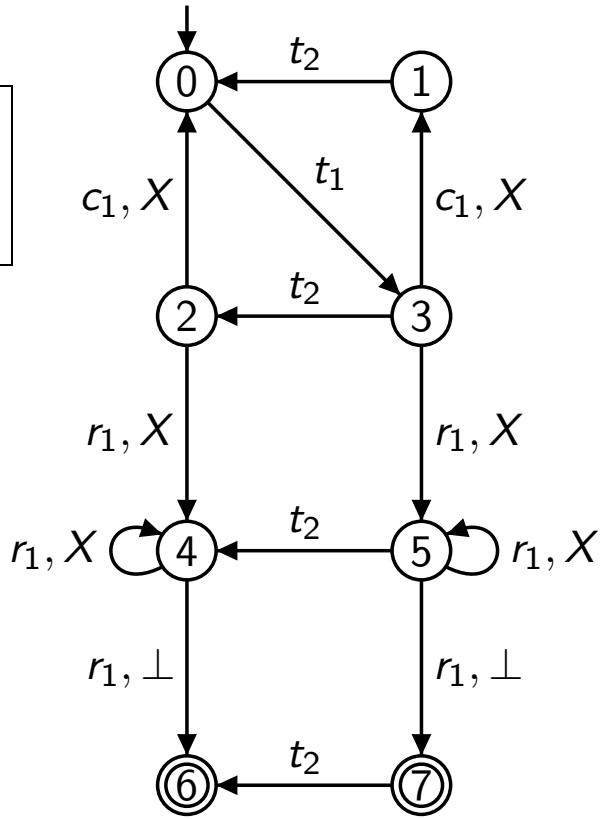
$\bot$

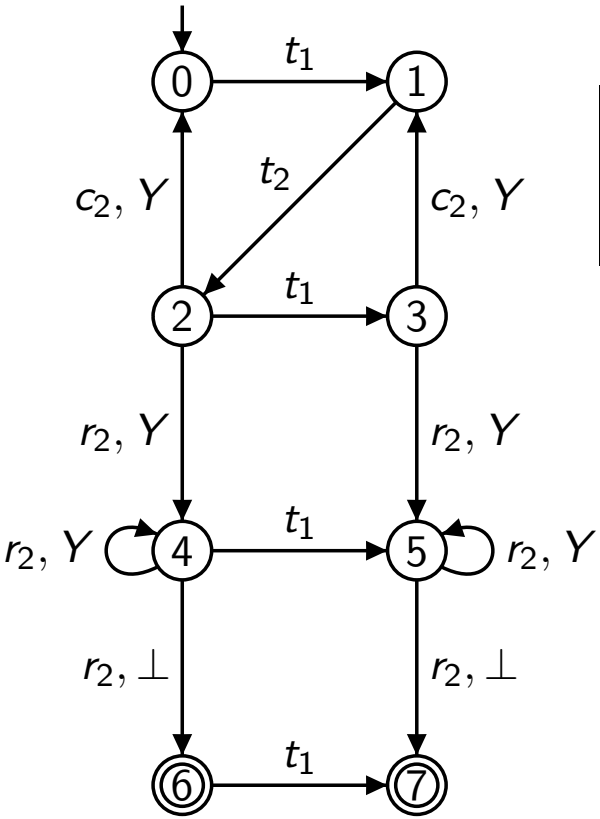
$$L(\mathcal{A}_p) = \{ t_1 ((c_1\ t_2 + t_2\ c_1)\ t_1)^n\ r_1^{\,i} (t_2 + \varepsilon)\ r_1^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

$$L(\mathcal{A}_q) = \{ t_1\ t_2 ((c_2\ t_1 + t_1\ c_2)\ t_2)^n\ r_2^{\,i} (t_1 + \varepsilon)\ r_2^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

Concurrent visibly pushdown automaton

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

$$L(\mathcal{A}_p) = \{ t_1 ((c_1\ t_2 + t_2\ c_1)\ t_1)^n\ r_1^{\,i} (t_2 + \varepsilon)\ r_1^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

$$L(\mathcal{A}_q) = \{ t_1\ t_2 ((c_2\ t_1 + t_1\ c_2)\ t_2)^n\ r_2^{\,i} (t_1 + \varepsilon)\ r_2^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid w \restriction \Sigma_p \in L(\mathcal{A}_p) \text{ and } w \restriction \Sigma_q \in L(\mathcal{A}_q)\}$$

The architecture of a concurrent recursive program

Definition

We fix the following parameters:

- $\mathcal{P}$ a finite set of processes
- $\widetilde{\Sigma} = ((\Sigma_p^{\text{call}}, \Sigma_p^{\text{ret}}, \Sigma_p^{\text{int}}))_{p \in \mathcal{P}}$ a concurrent pushdown alphabet:
  - Σ_p^{call}, Σ_p^{ret}, Σ_p^{int} are pairwise disjoint for all $p \in \mathcal{P}$
  - $(\Sigma_p^{\text{call}} \cup \Sigma_p^{\text{ret}}) \cap (\Sigma_q^{\text{call}} \cup \Sigma_q^{\text{ret}}) = \emptyset$ for all $p, q \in \mathcal{P}$ with $p \neq q$

Notation

- $\Sigma_p = \Sigma_p^{\text{call}} \cup \Sigma_p^{\text{ret}} \cup \Sigma_p^{\text{int}}$ and $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$
- $\Sigma^{\text{call}} = \bigcup_{p \in \mathcal{P}} \Sigma_p^{\text{call}}$ call actions
- $\Sigma^{\text{ret}} = \bigcup_{p \in \mathcal{P}} \Sigma_p^{\text{ret}}$ return actions
- $\Sigma^{\text{int}} = \Sigma \setminus (\Sigma^{\text{call}} \cup \Sigma^{\text{ret}})$ internal actions
- $proc(a) = \{p \in \mathcal{P} \mid a \in \Sigma_p\}$ processes involved in $a \in \Sigma$

The architecture of a concurrent recursive program

Definition

We fix the following parameters:

- $\mathcal{P}$ a finite set of processes
- $\widetilde{\Sigma} = ((\Sigma_p^{\text{call}}, \Sigma_p^{\text{ret}}, \Sigma_p^{\text{int}}))_{p \in \mathcal{P}}$ a concurrent pushdown alphabet:
  - Σ_p^{call}, Σ_p^{ret}, Σ_p^{int} are pairwise disjoint for all $p \in \mathcal{P}$
  - $(\Sigma_p^{\text{call}} \cup \Sigma_p^{\text{ret}}) \cap (\Sigma_q^{\text{call}} \cup \Sigma_q^{\text{ret}}) = \emptyset$ for all $p, q \in \mathcal{P}$ with $p \neq q$

Notation

- $\Sigma_p = \Sigma_p^{\text{call}} \cup \Sigma_p^{\text{ret}} \cup \Sigma_p^{\text{int}}$ and $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$
- $\Sigma^{\text{call}} = \bigcup_{p \in \mathcal{P}} \Sigma_p^{\text{call}}$ call actions
- $\Sigma^{\text{ret}} = \bigcup_{p \in \mathcal{P}} \Sigma_p^{\text{ret}}$ return actions
- $\Sigma^{\text{int}} = \Sigma \setminus (\Sigma^{\text{call}} \cup \Sigma^{\text{ret}})$ internal actions
- $\mathit{proc}(a) = \{p \in \mathcal{P} \mid a \in \Sigma_p\}$ processes involved in $a \in \Sigma$

Definition of concurrrent visibly pushdown automaton

Definition

A concurrent visibly pushdown automaton (CVPA) over $\widetilde{\Sigma}$ is a structure

$$((S_p)_{p\in\mathcal{P}}, \Gamma, (\delta_a)_{a\in\Sigma}, \iota, F)$$

- S_p is a finite set of local states
- Γ contains the stack symbols including a special symbol $\bot$
- $\delta_a \subseteq S_a \times (\Gamma \setminus \{\bot\}) \times S_a$ if $a \in \Sigma^{\text{call}}$
  $\delta_a \subseteq S_a \times \Gamma \times S_a$ if $a \in \Sigma^{\text{ret}}$
  $\delta_a \subseteq S_a \times S_a$ if $a \in \Sigma^{\text{int}}$
  where $S_a = \prod_{p\in proc(a)} S_p$ is the set of a-local states
- $\iota \in \prod_{p\in\mathcal{P}} S_p$ initial state
- $F \subseteq \prod_{p\in\mathcal{P}} S_p$ set of final states

Definition of concurrrent visibly pushdown automaton

Definition

A concurrent visibly pushdown automaton (CVPA) over $\widetilde{\Sigma}$ is a structure

$$((S_p)_{p\in\mathcal{P}}, \Gamma, (\delta_a)_{a\in\Sigma}, \iota, F)$$

- S_p is a finite set of local states
- Γ contains the stack symbols including a special symbol $\bot$
- $\delta_a \subseteq S_a \times (\Gamma \setminus \{\bot\}) \times S_a$ if $a \in \Sigma^{\text{call}}$
  $\delta_a \subseteq S_a \times \Gamma \times S_a$ if $a \in \Sigma^{\text{ret}}$
  $\delta_a \subseteq S_a \times S_a$ if $a \in \Sigma^{\text{int}}$
  where $S_a = \prod_{p\in proc(a)} S_p$ is the set of a-local states
- $\iota \in \prod_{p\in\mathcal{P}} S_p$ initial state
- $F \subseteq \prod_{p\in\mathcal{P}} S_p$ set of final states

Definition of concurrrent visibly pushdown automaton

Definition

A concurrent visibly pushdown automaton (CVPA) over $\widetilde{\Sigma}$ is a structure

$$((S_p)_{p\in\mathcal{P}}, \Gamma, (\delta_a)_{a\in\Sigma}, \iota, F)$$

- S_p is a finite set of local states
- Γ contains the stack symbols including a special symbol $\bot$
- $\delta_a \subseteq S_a \times (\Gamma \setminus \{\bot\}) \times S_a$ if $a \in \Sigma^{\text{call}}$

  $\delta_a \subseteq S_a \times \Gamma \times S_a$ if $a \in \Sigma^{\text{ret}}$

  $\delta_a \subseteq S_a \times S_a$ if $a \in \Sigma^{\text{int}}$

  where $S_a = \prod_{p\in proc(a)} S_p$ is the set of a-local states
- $\iota \in \prod_{p\in\mathcal{P}} S_p$ initial state
- $F \subseteq \prod_{p\in\mathcal{P}} S_p$ set of final states

Definition of concurrrent visibly pushdown automaton

Definition

A concurrent visibly pushdown automaton (CVPA) over $\widetilde{\Sigma}$ is a structure

$$((S_p)_{p\in\mathcal{P}}, \Gamma, (\delta_a)_{a\in\Sigma}, \iota, F)$$

- S_p is a finite set of local states
- Γ contains the stack symbols including a special symbol $\bot$
- $\delta_a \subseteq S_a \times (\Gamma \setminus \{\bot\}) \times S_a$ if $a \in \Sigma^{\text{call}}$

  $\delta_a \subseteq S_a \times \Gamma \times S_a$ if $a \in \Sigma^{\text{ret}}$

  $\delta_a \subseteq S_a \times S_a$ if $a \in \Sigma^{\text{int}}$

  where $S_a = \prod_{p\in proc(a)} S_p$ is the set of a-local states
- $\iota \in \prod_{p\in\mathcal{P}} S_p$ initial state
- $F \subseteq \prod_{p\in\mathcal{P}} S_p$ set of final states

Semantics of concurrent visibly pushdown automaton

Let $\mathcal{C} = ((S_p)_{p\in\mathcal{P}}, \Gamma, (\delta_a)_{a\in\Sigma}, \iota, F)$ be a CVPA.

Definition

- set of configurations of $\mathcal{C}$:

$$\prod_{p\in\mathcal{P}} S_p \times \prod_{p\in\mathcal{P}} (\Gamma^* \setminus \{\bot\})\{\bot\}$$

- global transition:

$$(s,\sigma) \stackrel{a}{\Longrightarrow} (s',\sigma')$$

$a \in \Sigma_p^{\text{call}}$ $(s_{proc(a)}, a, A, s'_{proc(a)}) \in \delta_a$ and $\sigma'_p = A \cdot \sigma_p$ for some $A \in \Gamma$

$a \in \Sigma_p^{\text{ret}}$ $(s_{proc(a)}, a, A, s'_{proc(a)}) \in \delta_a$ for some $A \in \Gamma$ such that either $A \neq \bot$ and $\sigma_p = A \cdot \sigma'_p$, or $A = \bot$ and $\sigma_p = \sigma'_p = \bot$

$a \in \Sigma^{\text{int}}$ $(s_{proc(a)}, a, s'_{proc(a)}) \in \delta_a$

All other components remain unchanged.

Semantics of concurrent visibly pushdown automaton

Let $\mathcal{C} = ((S_p)_{p\in\mathcal{P}}, \Gamma, (\delta_a)_{a\in\Sigma}, \iota, F)$ be a CVPA.

Definition

- set of configurations of $\mathcal{C}$:

$$\prod_{p\in\mathcal{P}} S_p \times \prod_{p\in\mathcal{P}} (\Gamma^* \setminus \{\bot\})\{\bot\}$$

- global transition:

$$(s,\sigma) \stackrel{a}{\Longrightarrow} (s',\sigma')$$

$a \in \Sigma_p^{\text{call}}$ $(s_{proc(a)}, a, A, s'_{proc(a)}) \in \delta_a$ and $\sigma'_p = A \cdot \sigma_p$ for some $A \in \Gamma$

$a \in \Sigma_p^{\text{ret}}$ $(s_{proc(a)}, a, A, s'_{proc(a)}) \in \delta_a$ for some $A \in \Gamma$ such that either $A \neq \bot$ and $\sigma_p = A \cdot \sigma'_p$, or $A = \bot$ and $\sigma_p = \sigma'_p = \bot$

$a \in \Sigma^{\text{int}}$ $(s_{proc(a)}, a, s'_{proc(a)}) \in \delta_a$

All other components remain unchanged.

Semantics of CVPA ...

... can also be described in terms of MVPA:

Definition ([LMP'07])

A multi-stack visibly pushdown automaton (MVPA) over $\widetilde{\Sigma}$ is a structure

$$\mathcal{A} = (S, \Gamma, \Delta, \iota, F)$$

- $\Delta \subseteq \quad S \times \Sigma^{\text{call}} \times (\Gamma \setminus \{\bot\}) \times S$
  $\cup \quad S \times \Sigma^{\text{ret}} \times \Gamma \times S$
  $\cup \quad S \times \Sigma^{\text{int}} \times S$

- set of configurations: $S \times \prod_{p \in \mathcal{P}} (\Gamma^* \setminus \{\bot\})\{\bot\}$
- notion of a 'process' meaningless
- processes only determine number of stacks

[LMP'07] La Torre & Madhusudan & Parlato. A Robust Class of Context-Sensitive Languages 2007.

Semantics of CVPA ...

... can also be described in terms of MVPA:

Definition ([LMP'07])

A multi-stack visibly pushdown automaton (MVPA) over $\widetilde{\Sigma}$ is a structure

$$\mathcal{A} = (S, \Gamma, \Delta, \iota, F)$$

- $\Delta \subseteq \quad S \times \Sigma^{\text{call}} \times (\Gamma \setminus \{\bot\}) \times S$
  $\cup \quad S \times \Sigma^{\text{ret}} \times \Gamma \times S$
  $\cup \quad S \times \Sigma^{\text{int}} \times S$

- set of configurations: $S \times \prod_{p \in \mathcal{P}} (\Gamma^* \setminus \{\bot\})\{\bot\}$
- notion of a 'process' meaningless
- processes only determine number of stacks

[LMP'07] La Torre & Madhusudan & Parlato. A Robust Class of Context-Sensitive Languages 2007.

Specification formalisms for distributed systems

Specification

finite automata
rational expressions
temporal logics (LTL, CTL, LTrL, ...)
MVPA
monadic second-order logic
...

Synthesis

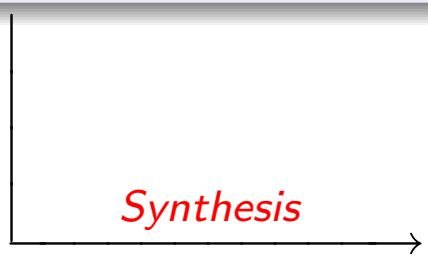
Implementation

asynchronous automata
message-passing automata
CVPA
...

Specification formalisms for distributed systems

Specification

finite automata
rational expressions
temporal logics (LTL, CTL, LTrL, ...)
MVPA
monadic second-order logic
...



Implementation

asynchronous automata
message-passing automata
CVPA
...

Specification formalisms for distributed systems

Specification

finite automata

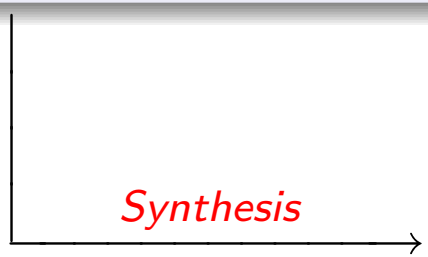
rational expressions

temporal logics (LTL, CTL, LTrL, ...)

MVPA

monadic second-order logic

...



Implementation

asynchronous automata

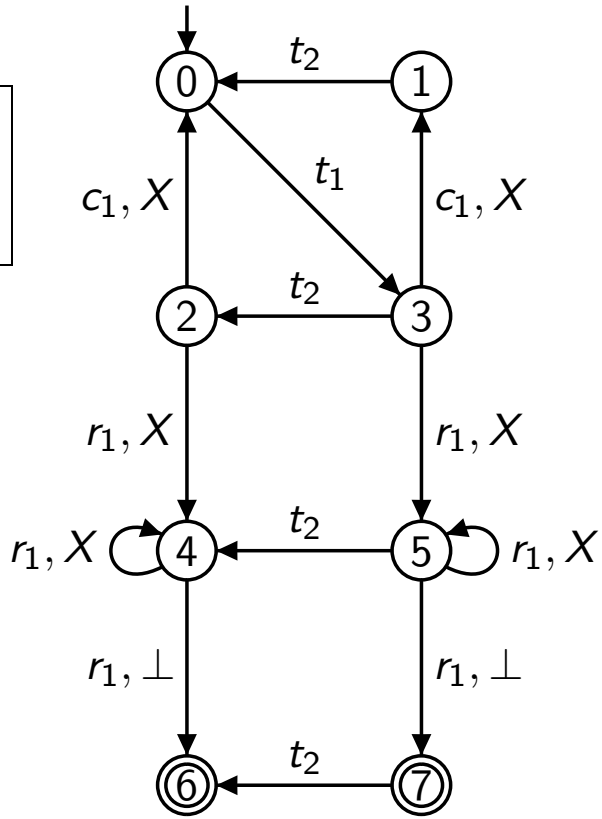
message-passing automata

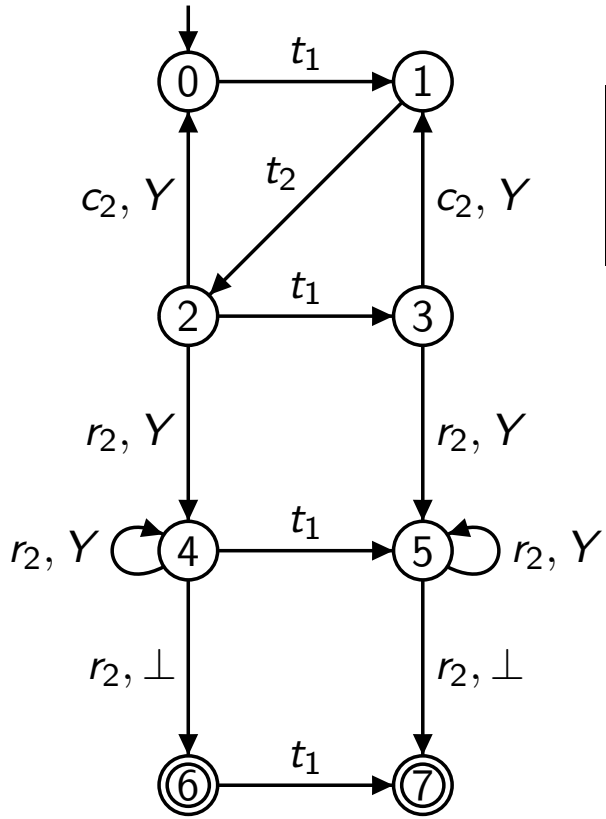
CVPA

...

Closure property of CVPA

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

	t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1
$\sim$	t_1	t_2	c_1	c_2	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1
$\sim$	t_1	t_2	c_1	c_2	t_1	c_1	t_2	r_2	t_1	r_2	r_1	r_1	r_1

Closure property of CVPA

Definition

$I_{\widetilde{\Sigma}} = \{(a, b) \in \Sigma \times \Sigma \mid proc(a) \cap proc(b) = \emptyset\}$

a and b are called independent if $(a, b) \in I_{\widetilde{\Sigma}}$

Definition

$\sim_{\widetilde{\Sigma}} \subseteq \Sigma^* \times \Sigma^*$ is the least congruence with $ab \sim_{\widetilde{\Sigma}} ba$ for all $(a, b) \in I_{\widetilde{\Sigma}}$.

Lemma

Let $\mathcal{C}$ be a CVPA. *For all $u, v \in \Sigma^*$ with $u \sim_{\widetilde{\Sigma}} v$:*

$$u \in L(\mathcal{C}) \text{ iff } v \in L(\mathcal{C})$$

Closure property of CVPA

Definition

$I_{\widetilde{\Sigma}} = \{(a, b) \in \Sigma \times \Sigma \mid proc(a) \cap proc(b) = \emptyset\}$

a and b are called independent if $(a, b) \in I_{\widetilde{\Sigma}}$

Definition

$\sim_{\widetilde{\Sigma}} \subseteq \Sigma^* \times \Sigma^*$ is the least congruence with $ab \sim_{\widetilde{\Sigma}} ba$ for all $(a, b) \in I_{\widetilde{\Sigma}}$.

Lemma

Let $\mathcal{C}$ be a CVPA. For all $u, v \in \Sigma^*$ with $u \sim_{\widetilde{\Sigma}} v$:

$$u \in L(\mathcal{C}) \text{ iff } v \in L(\mathcal{C})$$

Closure property of CVPA

Definition

$I_{\widetilde{\Sigma}} = \{(a, b) \in \Sigma \times \Sigma \mid proc(a) \cap proc(b) = \emptyset\}$

a and b are called independent if $(a, b) \in I_{\widetilde{\Sigma}}$

Definition

$\sim_{\widetilde{\Sigma}} \subseteq \Sigma^* \times \Sigma^*$ is the least congruence with $ab \sim_{\widetilde{\Sigma}} ba$ for all $(a, b) \in I_{\widetilde{\Sigma}}$.

Lemma

Let $\mathcal{C}$ be a CVPA. For all $u, v \in \Sigma^*$ with $u \sim_{\widetilde{\Sigma}} v$:

$$u \in L(\mathcal{C}) \text{ iff } v \in L(\mathcal{C})$$

Zielonka's Theorem

Let $\widetilde{\Sigma}$ be a concurrent pushdown alphabet.

Remark

Suppose $\Sigma = \Sigma^{\text{int}}$. Then,

- an MVPA over $\widetilde{\Sigma}$ is a finite automaton over Σ.
- a CVPA over $\widetilde{\Sigma}$ is an asynchronous automaton over $\widetilde{\Sigma}$.

Theorem ([Zie'87])

Suppose $\Sigma = \Sigma^{\text{int}}$. Let $L \subseteq \Sigma^*$ be a $\sim_{\widetilde{\Sigma}}$-closed regular language.

There is a CVPA $\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = L$.

[Zie'87] Zielonka. Notes on finite asynchronous automata. 1987.

From MVPA to CVPA

Theorem

Let $\mathcal{A}$ be an MVPA *over $\widetilde{\Sigma}$ such that $L(\mathcal{A})$ is $\sim_{\widetilde{\Sigma}}$-closed.*

There is a CVPA *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = L(\mathcal{A})$. The size of $\mathcal{C}$ is*

- *doubly exponential in $|\mathcal{A}|$*
- *triply exponential in $|\Sigma|$*

Proof

- Interpret $\mathcal{A} = (S, \Gamma, \Delta, \iota, F)$ as finite automaton over $\Sigma \times \Gamma$.
- Apply Zielonka's Theorem to obtain CVPA $\mathcal{C}$ over $((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.
- Interpret $\mathcal{C}$ as CVPA over $\widetilde{\Sigma}$.

From MVPA to CVPA

Theorem

Let $\mathcal{A}$ be an MVPA *over $\widetilde{\Sigma}$ such that $L(\mathcal{A})$ is $\sim_{\widetilde{\Sigma}}$-closed.*

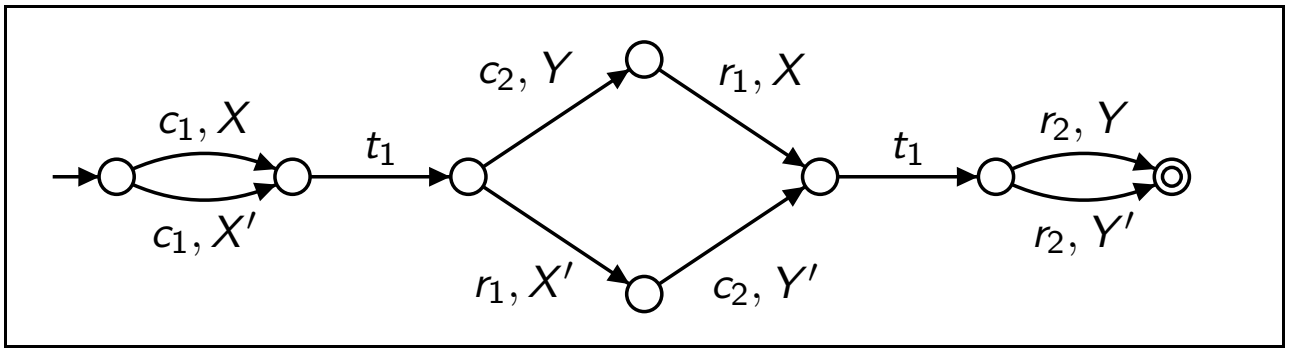
There is a CVPA *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = L(\mathcal{A})$. The size of $\mathcal{C}$ is*

- *doubly exponential in $|\mathcal{A}|$*
- *triply exponential in $|\Sigma|$*

Proof

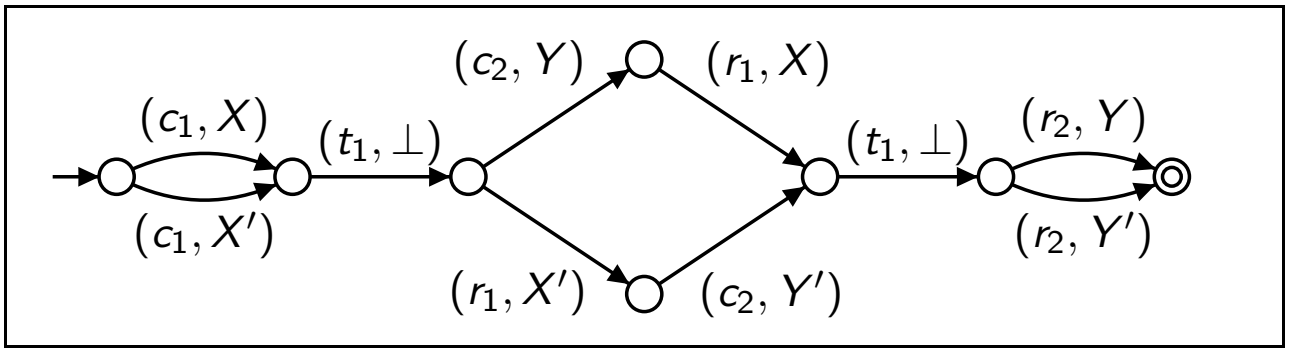
- Interpret $\mathcal{A} = (S, \Gamma, \Delta, \iota, F)$ as finite automaton over $\Sigma \times \Gamma$.
- Apply Zielonka's Theorem to obtain CVPA $\mathcal{C}$ over $((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.
- Interpret $\mathcal{C}$ as CVPA over $\widetilde{\Sigma}$.

From MVPA to CVPA



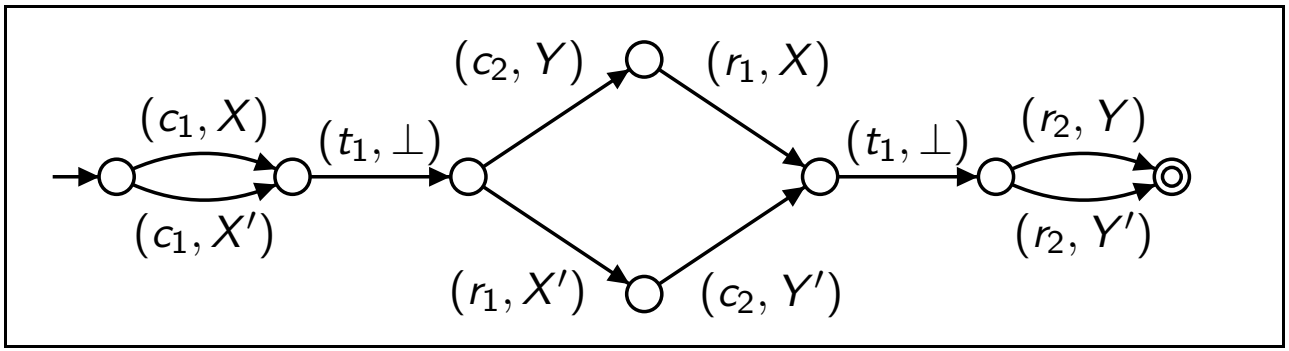
MVPA $\mathcal{A}$

From MVPA to CVPA



finite automaton $\mathcal{B}_1$ over $\Sigma \times \Gamma$

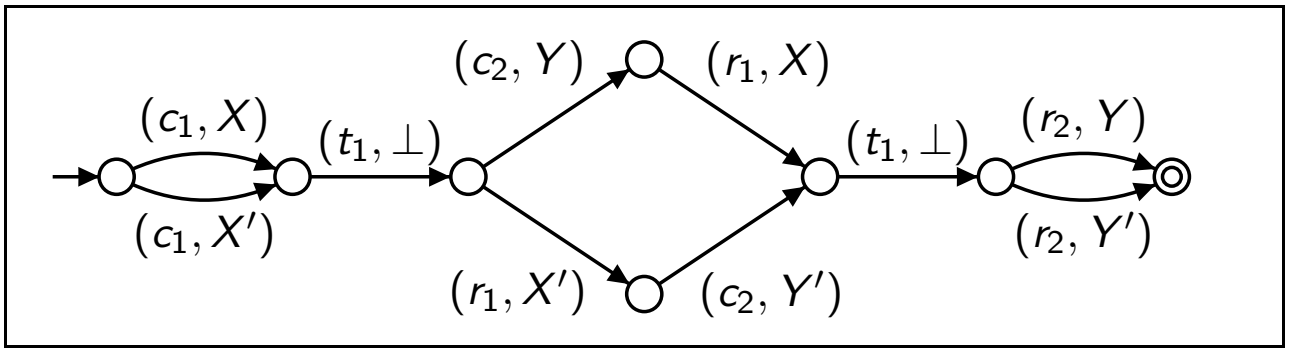
From MVPA to CVPA



finite automaton $\mathcal{B}_1$ over $\Sigma \times \Gamma$

- Consider the concurrent pushdown alphabet $\widetilde{\Omega} = ((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.

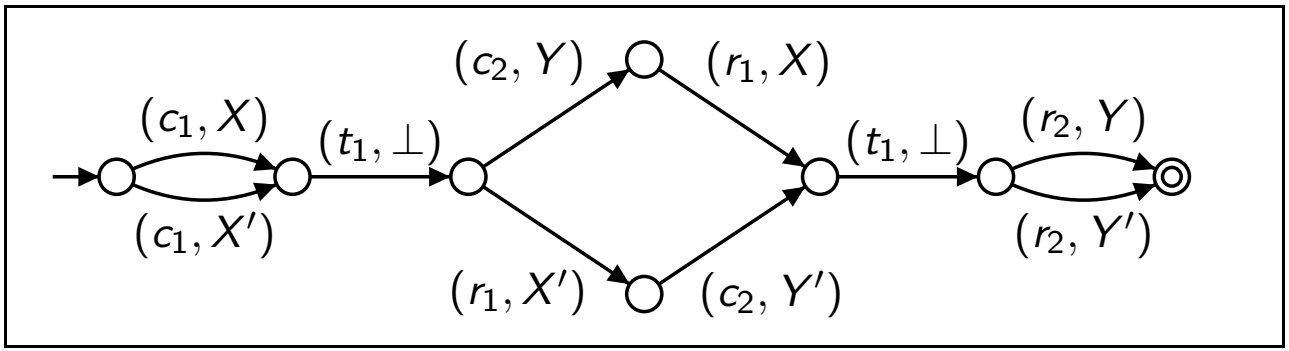
From MVPA to CVPA



finite automaton $\mathcal{B}_1$ over $\Sigma \times \Gamma$

- Consider the concurrent pushdown alphabet $\widetilde{\Omega} = ((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.
- Fix lexicographic order $c_2 <_{\text{lex}} r_1$.

From MVPA to CVPA

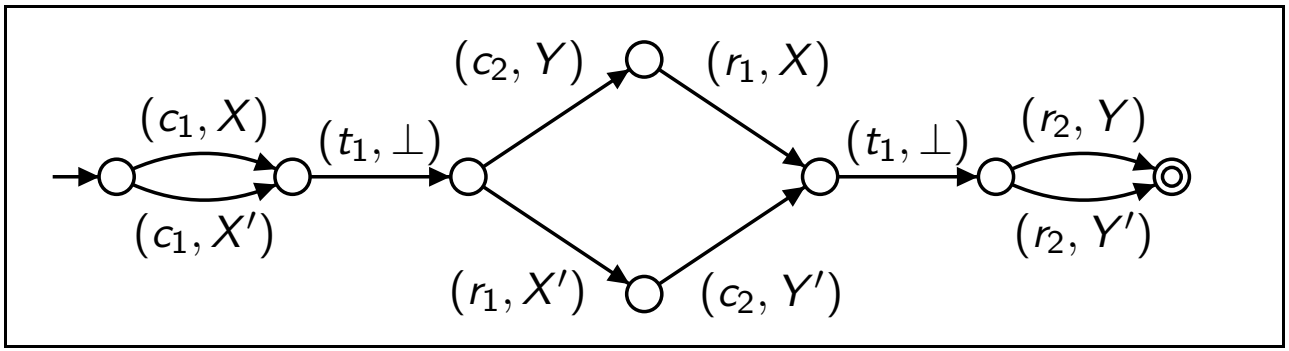


finite automaton $\mathcal{B}_1$ over $\Sigma \times \Gamma$

- Consider the concurrent pushdown alphabet $\widetilde{\Omega} = ((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.
- Fix lexicographic order $c_2 <_{\text{lex}} r_1$.
- There is a loop-connected finite automaton for the normal forms of $\mathcal{B}_1$ wrt. $<_{\text{lex}}$ of size $|\mathcal{B}_1| \cdot (|\Sigma| + 1)!$ [Kus'07].

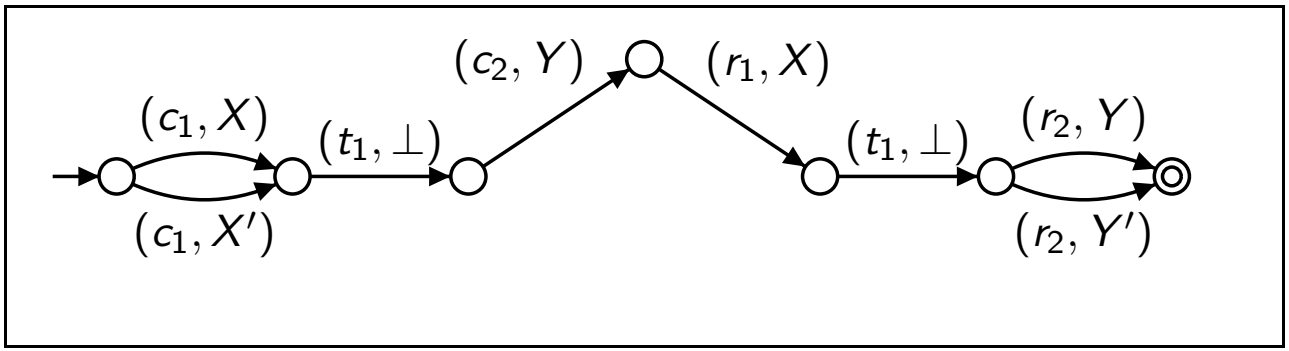
[Kus'07] Kuske. Weighted asynchronous cellular automata. 2007.

From MVPA to CVPA



finite automaton $\mathcal{B}_1$ over $\Sigma \times \Gamma$

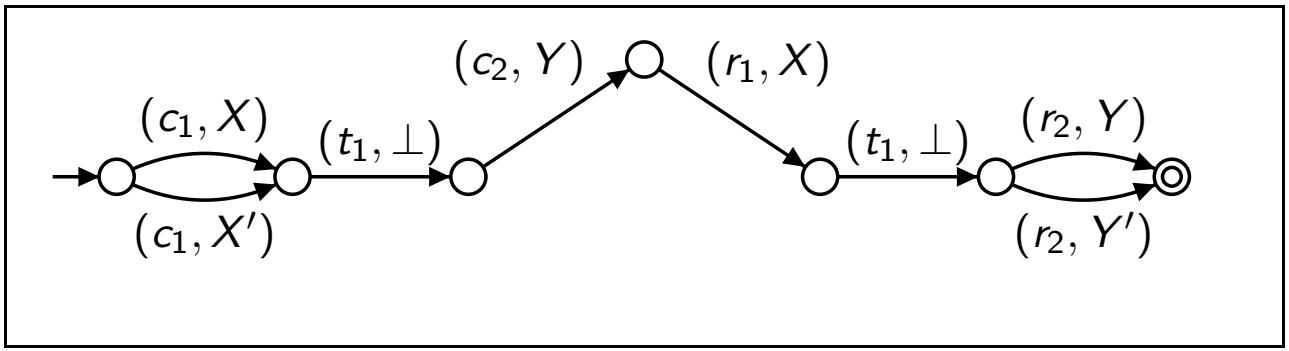
- Consider the concurrent pushdown alphabet $\widetilde{\Omega} = ((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.
- Fix lexicographic order $c_2 <_{\text{lex}} r_1$.
- There is a loop-connected finite automaton for the normal forms of $\mathcal{B}_1$ wrt. $<_{\text{lex}}$ of size $|\mathcal{B}_1| \cdot (|\Sigma| + 1)!$ [Kus'07].



loop-connected finite automaton $\mathcal{B}_2$ for normal forms

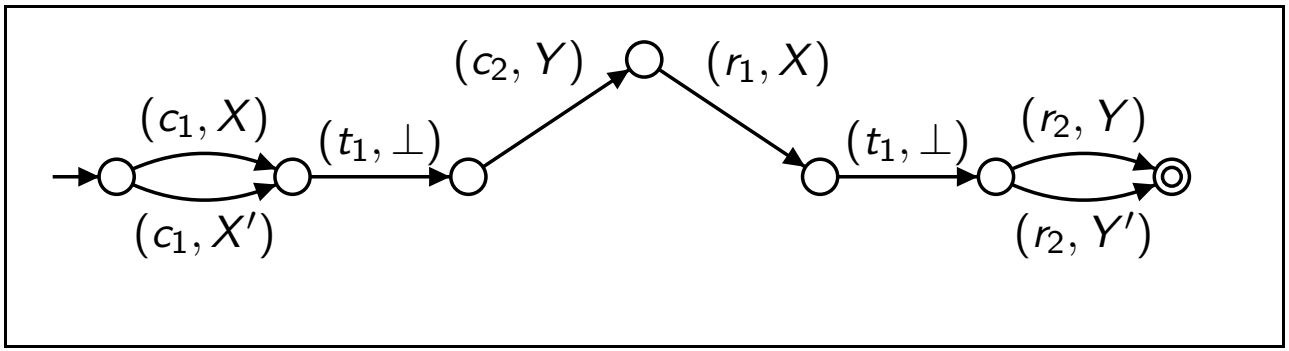
[Kus'07] Kuske. Weighted asynchronous cellular automata. 2007.

From MVPA to CVPA



loop-connected finite automaton $\mathcal{B}_2$ for normal forms

From MVPA to CVPA



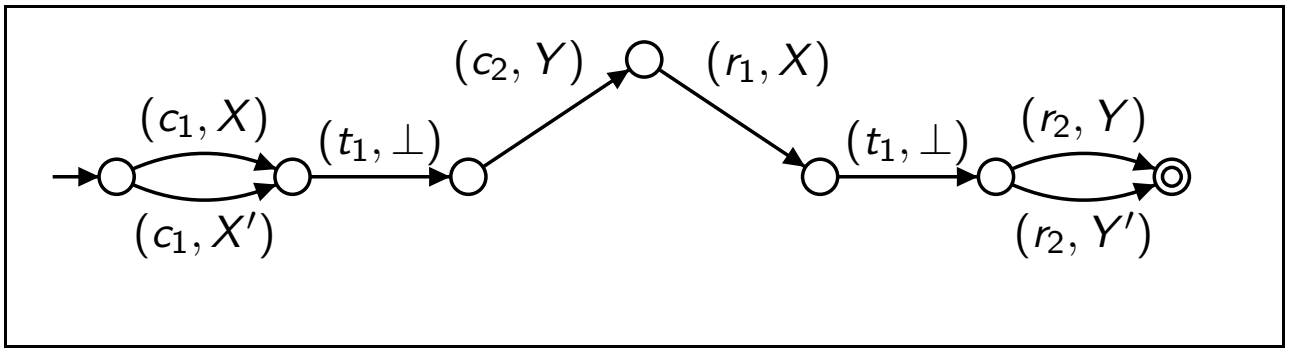
loop-connected finite automaton $\mathcal{B}_2$ for normal forms

- There is an I-diamond finite automaton for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$ of size $\exp(|\mathcal{B}_2|, |\Sigma|)$ [MP'99,Kus'07].

[MP'99] Muscholl & Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. 1999.

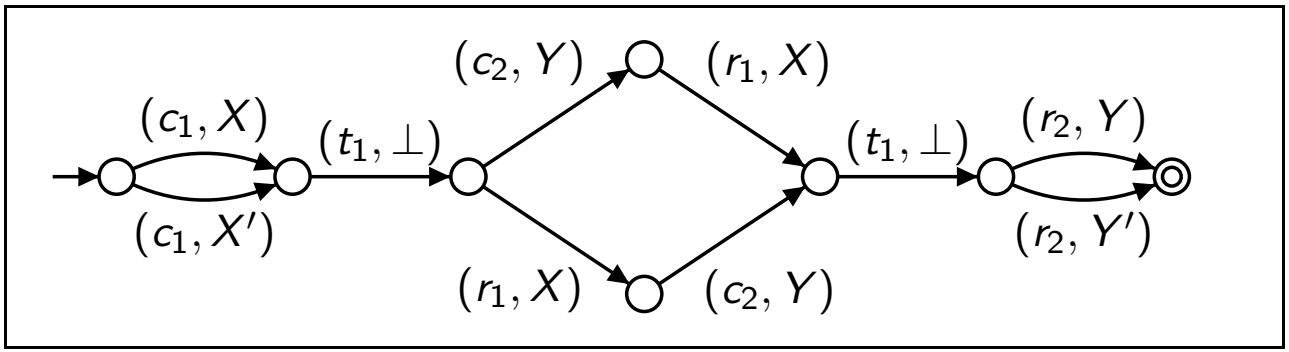
[Kus'07] Kuske. Weighted asynchronous cellular automata. 2007.

From MVPA to CVPA



loop-connected finite automaton $\mathcal{B}_2$ for normal forms

- There is an l-diamond finite automaton for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$ of size $\exp(|\mathcal{B}_2|, |\Sigma|)$ [MP'99,Kus'07].

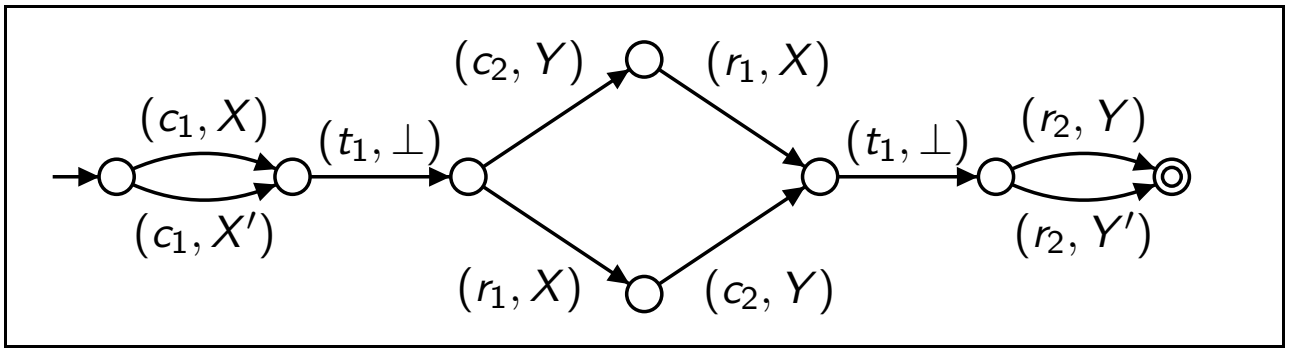


l-diamond finite automaton $\mathcal{B}_3$ for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$

[MP'99] Muscholl & Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. 1999.

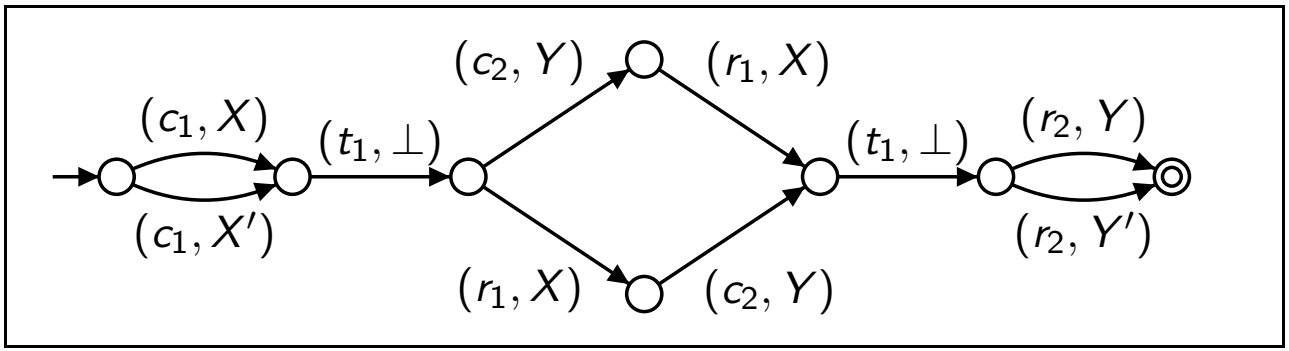
[Kus'07] Kuske. Weighted asynchronous cellular automata. 2007.

From MVPA to CVPA



l-diamond finite automaton $\mathcal{B}_3$ for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$

From MVPA to CVPA

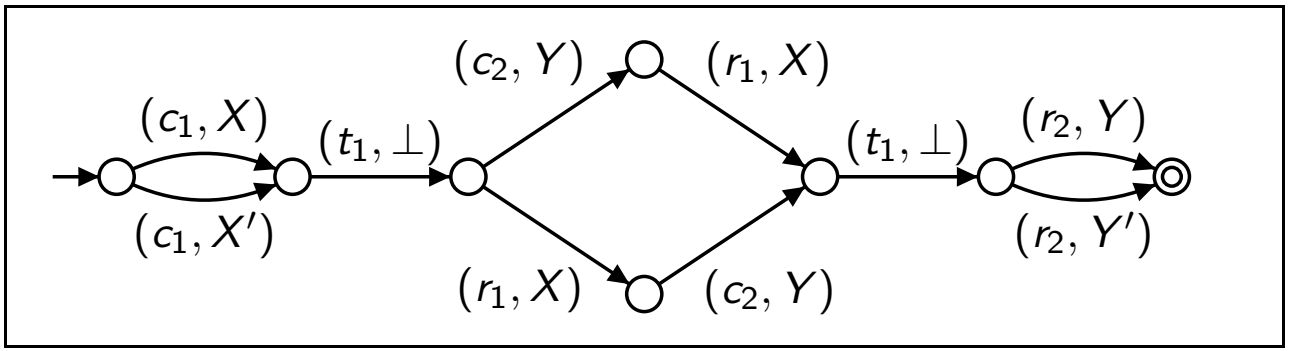


l-diamond finite automaton $\mathcal{B}_3$ for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$

- There is a CVPA $\mathcal{C}$ over $\widetilde{\Omega}$ such that $L(\mathcal{C}) = L(\mathcal{B}_3)$.
  It is of size $\exp(|\mathcal{B}_3|, |\Sigma|)$ [GM'06]

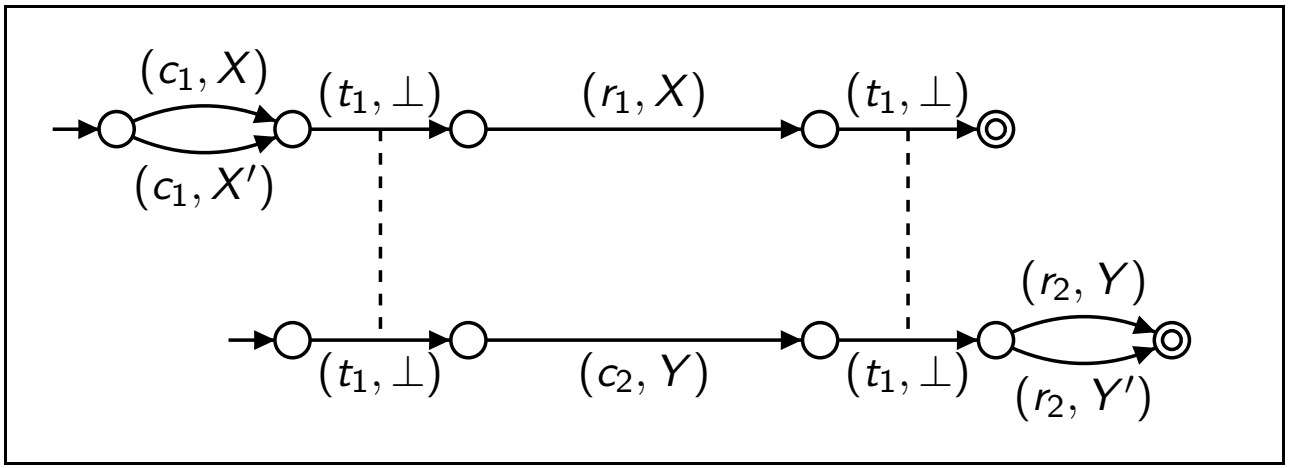
[GM'06] Genest & Muscholl. Constructing Exponential-size Deterministic Zielonka Automata. 2006.

From MVPA to CVPA



l-diamond finite automaton $\mathcal{B}_3$ for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$

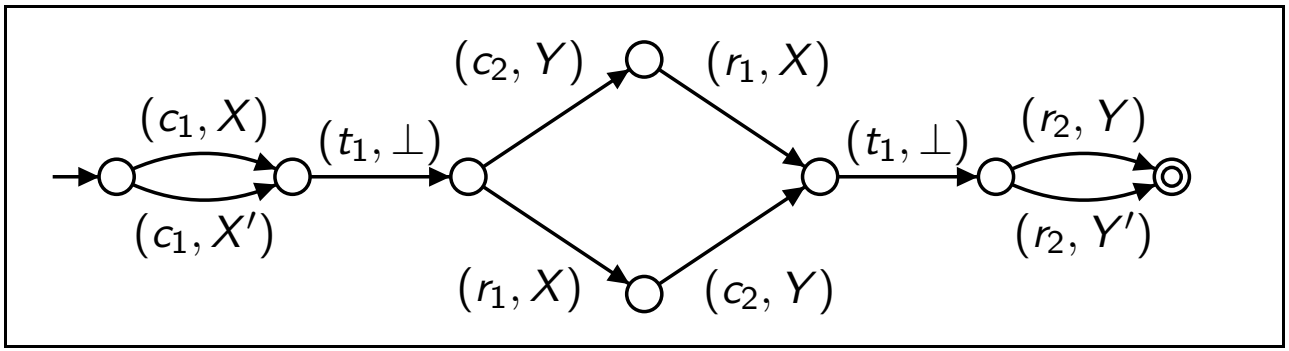
- There is a CVPA $\mathcal{C}$ over $\widetilde{\Omega}$ such that $L(\mathcal{C}) = L(\mathcal{B}_3)$.
  It is of size $\exp(|\mathcal{B}_3|, |\Sigma|)$ [GM'06]



CVPA $\mathcal{C}$ over $\widetilde{\Omega}$ for $L(\mathcal{B}_3)$

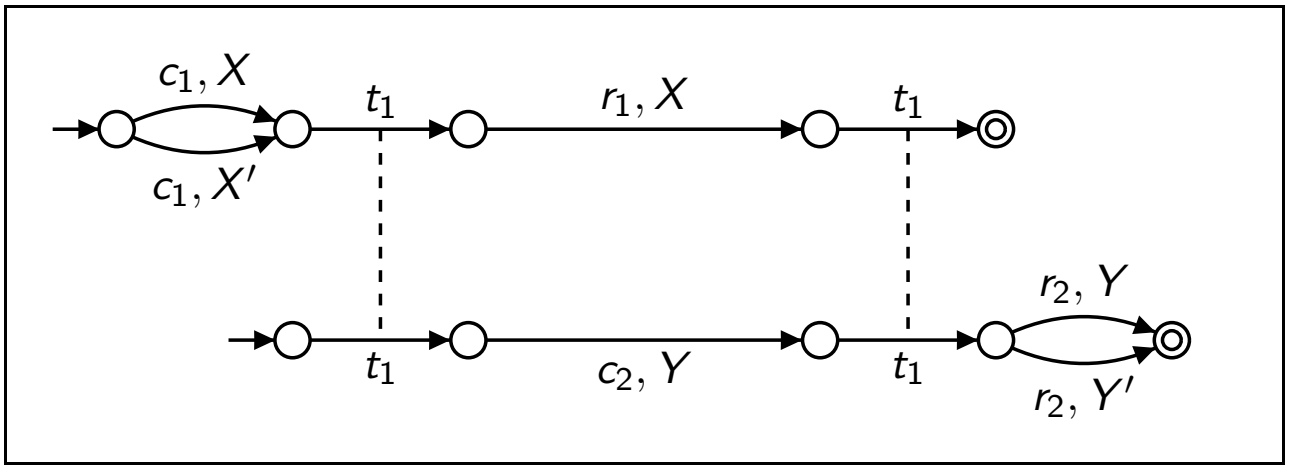
[GM'06] Genest & Muscholl. Constructing Exponential-size Deterministic Zielonka Automata. 2006.

From MVPA to CVPA



l-diamond finite automaton $\mathcal{B}_3$ for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$

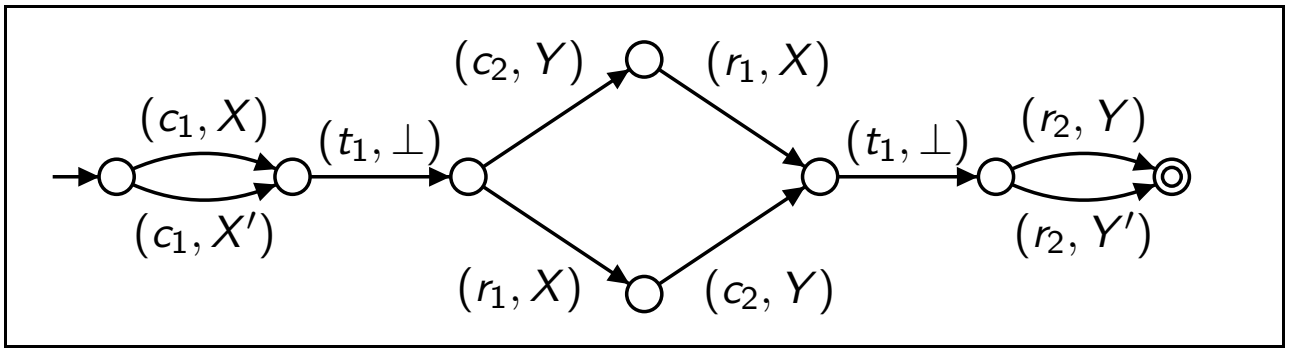
- There is a CVPA $\mathcal{C}$ over $\widetilde{\Omega}$ such that $L(\mathcal{C}) = L(\mathcal{B}_3)$.
  It is of size $\exp(|\mathcal{B}_3|, |\Sigma|)$ [GM'06]



CVPA $\mathcal{C}'$ over $\widetilde{\Sigma}$ for $L(\mathcal{A})$

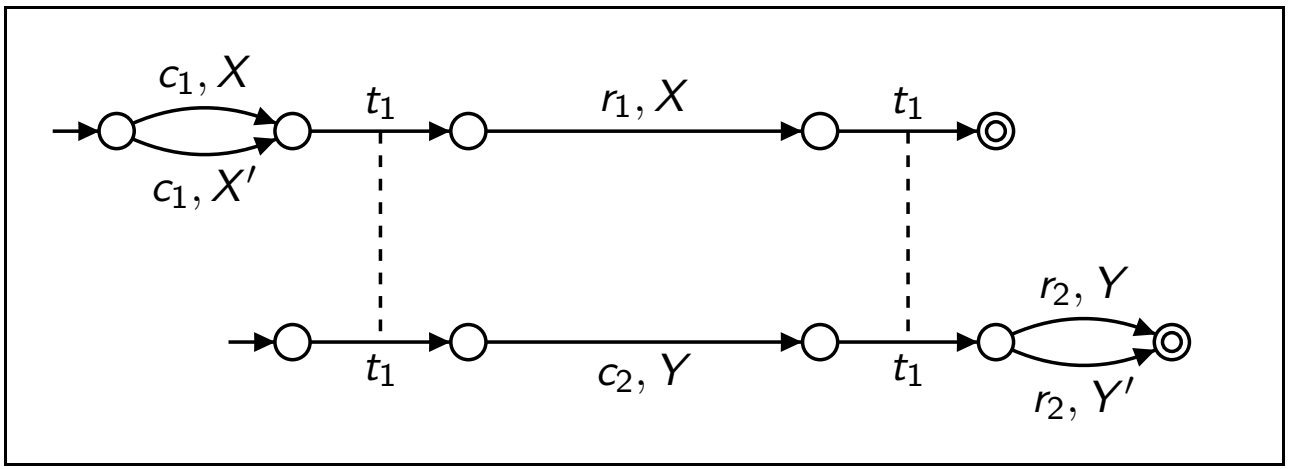
[GM'06] Genest & Muscholl. Constructing Exponential-size Deterministic Zielonka Automata. 2006.

From MVPA to CVPA



l-diamond finite automaton $\mathcal{B}_3$ for $[L(\mathcal{B}_2)]_{\sim_{\widetilde{\Omega}}}$

- There is a CVPA $\mathcal{C}$ over $\widetilde{\Omega}$ such that $L(\mathcal{C}) = L(\mathcal{B}_3)$.
  It is of size $\exp(|\mathcal{B}_3|, |\Sigma|)$ [GM'06] / $(3 \cdot |\Sigma| \cdot |\Gamma| \cdot |\mathcal{B}_3|)^{2^{|\Sigma| \cdot |\Gamma|}}$ [BM'06].



CVPA $\mathcal{C}'$ over $\widetilde{\Sigma}$ for $L(\mathcal{A})$

[GM'06] Genest & Muscholl. Constructing Exponential-size Deterministic Zielonka Automata. 2006.

[BM'06] Baudru & Morin. Unfolding Synthesis of Asynchronous Automata. 2006.

Is a specification implementable?

Theorem ([Zie'87])

Suppose $\Sigma = \Sigma^{\mathsf{int}}$. *Let* $L \subseteq \Sigma^*$ *be a* $\sim_{\widetilde{\Sigma}}$*-closed regular language.*

There is a CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$ *such that* $L(\mathcal{C}) = L$.

Theorem ([Mus'94,PWW'96])

Suppose $\Sigma = \Sigma^{\mathsf{int}}$. *The following problem is PSPACE-complete:*

INPUT:	MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$
QUESTION:	*Is* $L(\mathcal{A}) \sim_{\widetilde{\Sigma}}$*-closed?*

[Mus'94] Muscholl Über die Erkennbarkeit unendlicher Spuren. 1994

[PWW'96] Peled & Wilke & Wolper. An algorithmic approach for checking closure properties of temporal logic specifications and ω-regular languages. 1996.

Is a specification implementable?

Theorem

Let $\mathcal{A}$ be an MVPA *over $\widetilde{\Sigma}$ such that $L(\mathcal{A})$ is $\sim_{\widetilde{\Sigma}}$-closed.*

There is a CVPA *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = L$.*

Theorem

The following problem is undecidable:

INPUT: *Concurrent alphabet $\widetilde{\Sigma}$ and* MVPA *$\mathcal{A}$ over $\widetilde{\Sigma}$*
QUESTION: *Is $L(\mathcal{A})$ $\sim_{\widetilde{\Sigma}}$-closed?*

Restriction to k-phase words

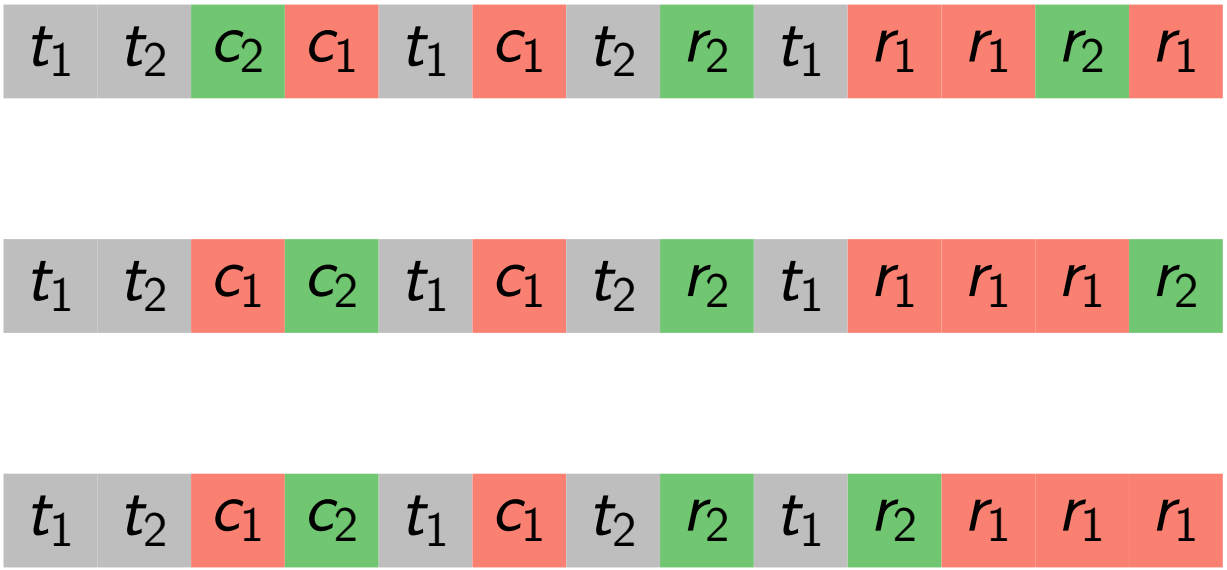
Definition

Let $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called a k-phase word over $\widetilde{\Sigma}$ if it can be written as $w_1 \cdot \ldots \cdot w_k$ where $w_i \in (\Sigma^{\text{call}} \cup \Sigma^{\text{int}} \cup \Sigma_p^{\text{ret}})^*$ for some $p \in \mathcal{P}$.

Restriction to k-phase words

Definition

Let $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called a k-phase word over $\widetilde{\Sigma}$ if it can be written as $w_1 \cdot \ldots \cdot w_k$ where $w_i \in (\Sigma^{\text{call}} \cup \Sigma^{\text{int}} \cup \Sigma_p^{\text{ret}})^*$ for some $p \in \mathcal{P}$.



Restriction to k-phase words

Definition

Let $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called a k-phase word over $\widetilde{\Sigma}$ if it can be written as $w_1 \cdot \ldots \cdot w_k$ where $w_i \in (\Sigma^{\text{call}} \cup \Sigma^{\text{int}} \cup \Sigma_p^{\text{ret}})^*$ for some $p \in \mathcal{P}$.

t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1
1									2		3	4

4-phase word

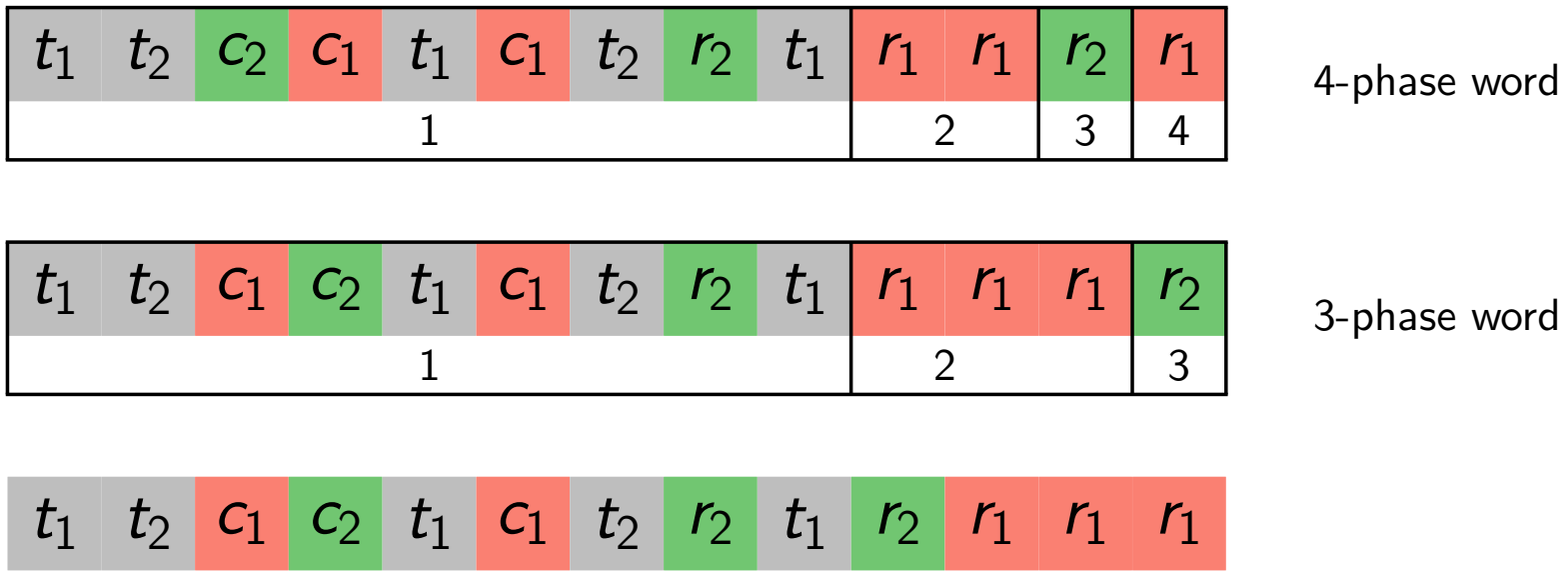
$\sim$ t_1 t_2 c_1 c_2 t_1 c_1 t_2 r_2 t_1 r_1 r_1 r_1 r_2

$\sim$ t_1 t_2 c_1 c_2 t_1 c_1 t_2 r_2 t_1 r_2 r_1 r_1 r_1

Restriction to k-phase words

Definition

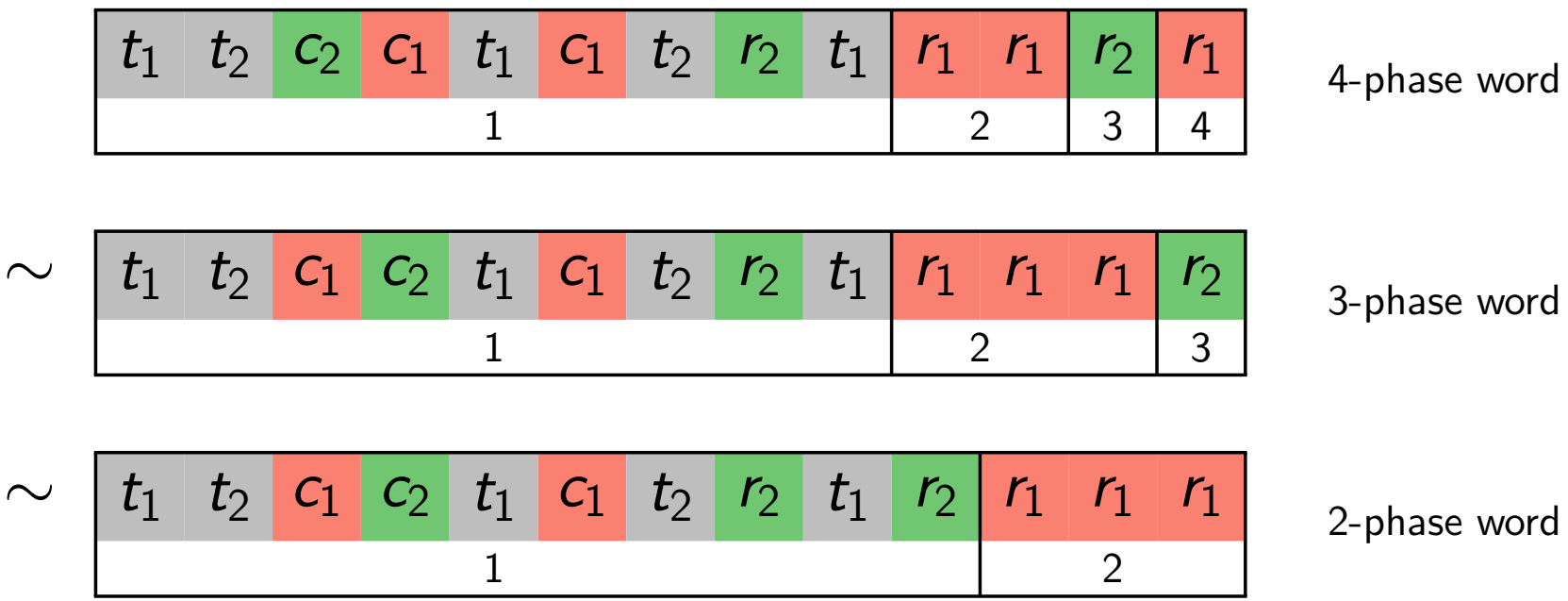
Let $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called a k-phase word over $\widetilde{\Sigma}$ if it can be written as $w_1 \cdot \ldots \cdot w_k$ where $w_i \in (\Sigma^{\text{call}} \cup \Sigma^{\text{int}} \cup \Sigma_p^{\text{ret}})^*$ for some $p \in \mathcal{P}$.



Restriction to k-phase words

Definition

Let $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called a k-phase word over $\widetilde{\Sigma}$ if it can be written as $w_1 \cdot \ldots \cdot w_k$ where $w_i \in (\Sigma^{\text{call}} \cup \Sigma^{\text{int}} \cup \Sigma_p^{\text{ret}})^*$ for some $p \in \mathcal{P}$.



MVPA and k-phase words

Notation

- Let $W_k(\widetilde{\Sigma})$ denote the set of k-phase words over $\widetilde{\Sigma}$.
- For an MVPA $\mathcal{A}$, let $L_k(\mathcal{A}) = L(\mathcal{A}) \cap W_k(\widetilde{\Sigma})$.

Theorem ([LMP'07])

The following problem is decidable:

INPUT: *Concurrent alphabet $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ over $\widetilde{\Sigma}$, and $k \in \mathbb{N}$*
QUESTION: *Does $L_k(\mathcal{A}) \neq \emptyset$ hold?*

The time complexity is doubly exponential wrt. k.

Theorem ([LMP'07])

Let $k \in \mathbb{N}$ and let $\mathcal{A}$ be an MVPA over $\widetilde{\Sigma}$. One can effectively construct an MVPA $\mathcal{A}'$ over $\widetilde{\Sigma}$ such that $L(\mathcal{A}') = \Sigma^* \setminus L_k(\mathcal{A})$.

[LMP'07] La Torre & Madhusudan & Parlato. A Robust Class of Context-Sensitive Languages 2007.

MVPA and k-phase words

Notation

- Let $W_k(\widetilde{\Sigma})$ denote the set of k-phase words over $\widetilde{\Sigma}$.
- For an MVPA $\mathcal{A}$, let $L_k(\mathcal{A}) = L(\mathcal{A}) \cap W_k(\widetilde{\Sigma})$.

Theorem ([LMP'07])

The following problem is decidable:

INPUT: *Concurrent alphabet* $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$, *and* $k \in \mathbb{N}$
QUESTION: *Does* $L_k(\mathcal{A}) \neq \emptyset$ *hold?*

The time complexity is doubly exponential wrt. k.

Theorem ([LMP'07])

Let $k \in \mathbb{N}$ *and let* $\mathcal{A}$ *be an* MVPA *over* $\widetilde{\Sigma}$. *One can effectively construct an* MVPA $\mathcal{A}'$ *over* $\widetilde{\Sigma}$ *such that* $L(\mathcal{A}') = \Sigma^* \setminus L_k(\mathcal{A})$.

[LMP'07] La Torre & Madhusudan & Parlato. A Robust Class of Context-Sensitive Languages 2007.

MVPA and k-phase words

Notation

- Let $W_k(\widetilde{\Sigma})$ denote the set of k-phase words over $\widetilde{\Sigma}$.
- For an MVPA $\mathcal{A}$, let $L_k(\mathcal{A}) = L(\mathcal{A}) \cap W_k(\widetilde{\Sigma})$.

Theorem ([LMP'07])

The following problem is decidable:

INPUT: *Concurrent alphabet* $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$, *and* $k \in \mathbb{N}$
QUESTION: *Does* $L_k(\mathcal{A}) \neq \emptyset$ *hold?*

The time complexity is doubly exponential wrt. k.

Theorem ([LMP'07])

Let $k \in \mathbb{N}$ *and let* $\mathcal{A}$ *be an* MVPA *over* $\widetilde{\Sigma}$. *One can effectively construct an* MVPA $\mathcal{A}'$ *over* $\widetilde{\Sigma}$ *such that* $L(\mathcal{A}') = \Sigma^* \setminus L_k(\mathcal{A})$.

[LMP'07] La Torre & Madhusudan & Parlato. A Robust Class of Context-Sensitive Languages 2007.

A decidable sufficient criterion for implementability

Definition

A set $L \subseteq W_k(\widetilde{\Sigma})$ is a k-phase representation if, for all $u, v \in \Sigma^*$ and $(a, b) \in I_{\widetilde{\Sigma}}$ with $\{uabv, ubav\} \subseteq W_k(\widetilde{\Sigma})$, we have $uabv \in L$ iff $ubav \in L$.

A decidable sufficient criterion for implementability

Definition

A set $L \subseteq \mathrm{W}_k(\widetilde{\Sigma})$ is a k-phase representation if, for all $u, v \in \Sigma^*$ and $(a, b) \in I_{\widetilde{\Sigma}}$ with $\{uabv, ubav\} \subseteq \mathrm{W}_k(\widetilde{\Sigma})$, we have $uabv \in L$ iff $ubav \in L$.

$$\left\{ u \; t_1 \; r_1^{\,m} \; r_2^{\,n} \;\middle|\; m, n \geq 2,\; u \in \{c_1, c_2\}^*,\; |u|_{c_1} = m,\; |u|_{c_2} = n \right\}$$

is a 2-phase representation.

A decidable sufficient criterion for implementability

Definition

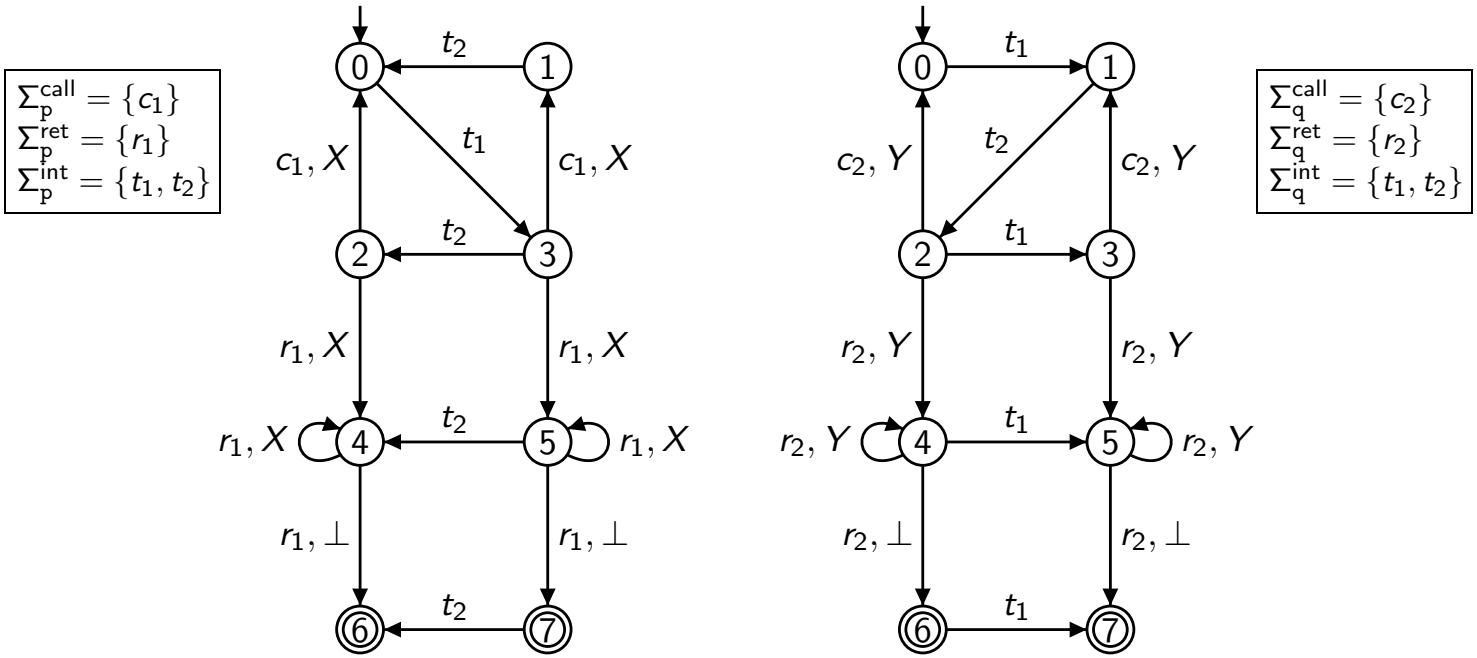
A set $L \subseteq \mathrm{W}_k(\widetilde{\Sigma})$ is a k-phase representation if, for all $u, v \in \Sigma^*$ and $(a, b) \in I_{\widetilde{\Sigma}}$ with $\{uabv, ubav\} \subseteq \mathrm{W}_k(\widetilde{\Sigma})$, we have $uabv \in L$ iff $ubav \in L$.

$$\left\{ u \; t_1 \; r_1^{\,m} \; r_2^{\,n} \;\middle|\; m, n \geq 2,\; u \in \{c_1, c_2\}^*,\; |u|_{c_1} = m,\; |u|_{c_2} = n \right\}$$

is a 2-phase representation.

Any $\sim_{\widetilde{\Sigma}}$-closed subset of $\mathrm{W}_k(\widetilde{\Sigma})$ is a k-phase representation.

The closure of a k-phase representation

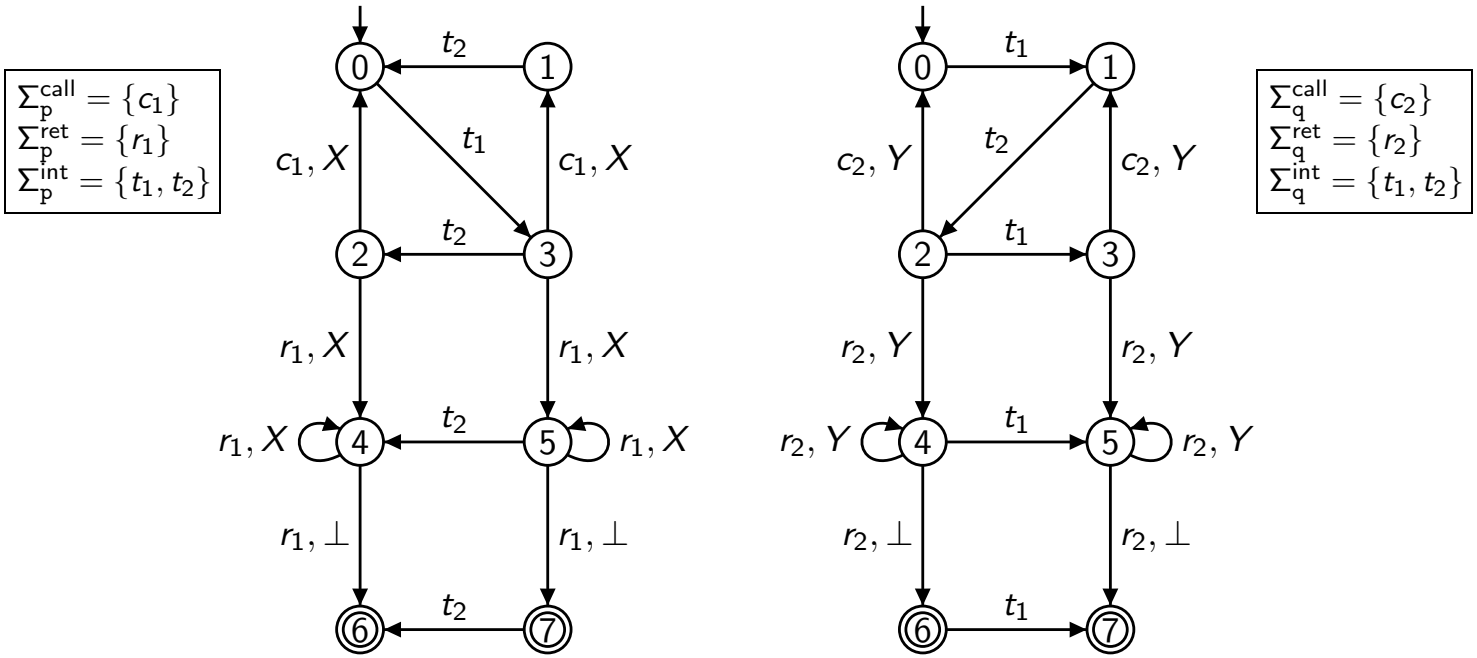


$$L(\mathcal{A}_p) = \{ t_1 ((c_1\ t_2 + t_2\ c_1)\ t_1)^n\ r_1^{\,i} (t_2 + \varepsilon)\ r_1^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

$$L(\mathcal{A}_q) = \{ t_1\ t_2 ((c_2\ t_1 + t_1\ c_2)\ t_2)^n\ r_2^{\,i} (t_1 + \varepsilon)\ r_2^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid w \restriction \Sigma_p \in L(\mathcal{A}_p) \text{ and } w \restriction \Sigma_q \in L(\mathcal{A}_q)\}$$

The closure of a k-phase representation



$$L(\mathcal{A}_p) = \{ t_1 ((c_1\ t_2 + t_2\ c_1)\ t_1)^n\ r_1^{\,i} (t_2 + \varepsilon)\ r_1^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

$$L(\mathcal{A}_q) = \{ t_1\ t_2 ((c_2\ t_1 + t_1\ c_2)\ t_2)^n\ r_2^{\,i} (t_1 + \varepsilon)\ r_2^{\,j} \mid n, i, j \in \mathbb{N},\ i + j = n + 1\}$$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid w \restriction \Sigma_p \in L(\mathcal{A}_p) \text{ and } w \restriction \Sigma_q \in L(\mathcal{A}_q)\}$$

$L_2(\mathcal{A})$ is a 2-phase representation and we have $[L_2(\mathcal{A})]_{\sim_{\widetilde{\Sigma}}} = L(\mathcal{A})$.

From MVPA to CVPA

Theorem

Let $\mathcal{A}$ *be an* MVPA *over* $\widetilde{\Sigma}$ *such that* $L_k(\mathcal{A})$ *is a* k*-phase representation.*

There is a CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$ *such that* $L(\mathcal{C}) = [L_k(\mathcal{A})]_{\sim_{\widetilde{\Sigma}}}$. *The size of* $\mathcal{C}$ *is*

- *doubly exponential in* $|\mathcal{A}|$ *and* k
- *triply exponential in* $|\Sigma|$

From Mvpa to Cvpa

Theorem

Let $\mathcal{A}$ be an Mvpa *over $\widetilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k-phase representation.*

There is a Cvpa *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = [L_k(\mathcal{A})]_{\sim_{\widetilde{\Sigma}}}$. The size of $\mathcal{C}$ is*

- *doubly exponential in $|\mathcal{A}|$ and k*
- *triply exponential in $|\Sigma|$*

Proof

- Set $\widetilde{\Omega} = ((\Sigma_p^{\mathsf{call}} \times \{1, \ldots, k\}, \Sigma_p^{\mathsf{ret}} \times \{1, \ldots, k\}, \Sigma_p^{\mathsf{int}} \times \{1, \ldots, k\}))_{p \in \mathcal{P}}$.
- Build Mvpa $\mathcal{B}$ over $\widetilde{\Omega}$ for words $(a_1, ph_1) \ldots (a_n, ph_n)$ such that $a_1 \ldots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \ldots a_i \text{ is } j\text{-phase word}\}$.
- Consider $<_{\mathsf{lex}} \subseteq \Omega^* \times \Omega^*$ such that $i < j$ implies $(a, i) <_{\mathsf{lex}} (b, j)$.
- $L(\mathcal{B})$ contains its normal forms wrt. $<_{\mathsf{lex}}$.
- Rest of construction as before.

From Mvpa to Cvpa

Theorem

Let $\mathcal{A}$ be an Mvpa *over $\widetilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k-phase representation.*

There is a Cvpa *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = [L_k(\mathcal{A})]_{\sim_{\widetilde{\Sigma}}}$. The size of $\mathcal{C}$ is*

- *doubly exponential in $|\mathcal{A}|$ and k*
- *triply exponential in $|\Sigma|$*

Proof

- Set $\widetilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \ldots, k\}, \Sigma_p^{\text{ret}} \times \{1, \ldots, k\}, \Sigma_p^{\text{int}} \times \{1, \ldots, k\}))_{p \in \mathcal{P}}$.
- Build Mvpa $\mathcal{B}$ over $\widetilde{\Omega}$ for words $(a_1, ph_1) \ldots (a_n, ph_n)$ such that $a_1 \ldots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \ldots a_i \text{ is } j\text{-phase word}\}$.
- Consider $<_{\text{lex}} \subseteq \Omega^* \times \Omega^*$ such that $i < j$ implies $(a, i) <_{\text{lex}} (b, j)$.
- $L(\mathcal{B})$ contains its normal forms wrt. $<_{\text{lex}}$.
- Rest of construction as before.

From Mvpa to Cvpa

Theorem

Let $\mathcal{A}$ be an Mvpa *over $\widetilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k-phase representation.*

There is a Cvpa *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = [L_k(\mathcal{A})]_{\sim_{\widetilde{\Sigma}}}$. The size of $\mathcal{C}$ is*

- *doubly exponential in $|\mathcal{A}|$ and k*
- *triply exponential in $|\Sigma|$*

Proof

- Set $\widetilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \ldots, k\}, \Sigma_p^{\text{ret}} \times \{1, \ldots, k\}, \Sigma_p^{\text{int}} \times \{1, \ldots, k\}))_{p \in \mathcal{P}}$.
- Build Mvpa $\mathcal{B}$ over $\widetilde{\Omega}$ for words $(a_1, ph_1) \ldots (a_n, ph_n)$ such that $a_1 \ldots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \ldots a_i \text{ is } j\text{-phase word}\}$.
- Consider $<_{\text{lex}} \subseteq \Omega^* \times \Omega^*$ such that $i < j$ implies $(a, i) <_{\text{lex}} (b, j)$.
- $L(\mathcal{B})$ contains its normal forms wrt. $<_{\text{lex}}$.
- Rest of construction as before.

From Mvpa to Cvpa

Theorem

Let $\mathcal{A}$ be an Mvpa *over $\widetilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k-phase representation.*

There is a Cvpa *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = [L_k(\mathcal{A})]_{\sim_{\widetilde{\Sigma}}}$. The size of $\mathcal{C}$ is*

- *doubly exponential in $|\mathcal{A}|$ and k*
- *triply exponential in $|\Sigma|$*

Proof

- Set $\widetilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \ldots, k\}, \Sigma_p^{\text{ret}} \times \{1, \ldots, k\}, \Sigma_p^{\text{int}} \times \{1, \ldots, k\}))_{p \in \mathcal{P}}$.
- Build Mvpa $\mathcal{B}$ over $\widetilde{\Omega}$ for words $(a_1, ph_1) \ldots (a_n, ph_n)$ such that $a_1 \ldots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \ldots a_i \text{ is } j\text{-phase word}\}$.
- Consider $<_{\text{lex}} \subseteq \Omega^* \times \Omega^*$ such that $i < j$ implies $(a, i) <_{\text{lex}} (b, j)$.
- $L(\mathcal{B})$ contains its normal forms wrt. $<_{\text{lex}}$.
- Rest of construction as before.

From Mvpa to Cvpa

Theorem

Let $\mathcal{A}$ be an Mvpa *over $\widetilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k-phase representation.*

There is a Cvpa *$\mathcal{C}$ over $\widetilde{\Sigma}$ such that $L(\mathcal{C}) = [L_k(\mathcal{A})]_{\sim_{\widetilde{\Sigma}}}$. The size of $\mathcal{C}$ is*

- *doubly exponential in $|\mathcal{A}|$ and k*
- *triply exponential in $|\Sigma|$*

Proof

- Set $\widetilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \ldots, k\}, \Sigma_p^{\text{ret}} \times \{1, \ldots, k\}, \Sigma_p^{\text{int}} \times \{1, \ldots, k\}))_{p \in \mathcal{P}}$.
- Build Mvpa $\mathcal{B}$ over $\widetilde{\Omega}$ for words $(a_1, ph_1) \ldots (a_n, ph_n)$ such that $a_1 \ldots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \ldots a_i \text{ is } j\text{-phase word}\}$.
- Consider $<_{\text{lex}} \subseteq \Omega^* \times \Omega^*$ such that $i < j$ implies $(a, i) <_{\text{lex}} (b, j)$.
- $L(\mathcal{B})$ contains its normal forms wrt. $<_{\text{lex}}$.
- Rest of construction as before.

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

INPUT: *Concurrent alphabet* $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$, *and* $k \in \mathbb{N}$

QUESTION 1: *Is* $L_k(\mathcal{A}) \sim_{\widetilde{\Sigma}}$*-closed?*

QUESTION 2: *Is* $L_k(\mathcal{A})$ *a* k*-phase representation?*

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

INPUT: *Concurrent alphabet* $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$, *and* $k \in \mathbb{N}$

QUESTION 1: *Is* $L_k(\mathcal{A})$ $\sim_{\widetilde{\Sigma}}$*-closed?*

QUESTION 2: *Is* $L_k(\mathcal{A})$ *a* k*-phase representation?*

Proof (Question 1)

- From $\mathcal{A}$ construct MVPA $\mathcal{A}'$ over $\widetilde{\Sigma}$ such that $L(\mathcal{A}') = \Sigma^* \setminus L_k(\mathcal{A})$.
- Build MVPA $\mathcal{B}$ over $\widetilde{\Sigma}$ recognizing words of the form $uabv$ with $(a, b) \in I_{\widetilde{\Sigma}}$, $uabv \in L(\mathcal{A})$, and $ubav \in L(\mathcal{A}')$.
- $L_k(\mathcal{A})$ is $\sim_{\widetilde{\Sigma}}$-closed iff $L_k(\mathcal{B}) \neq \emptyset$.
- For Question 2, build $\mathcal{A}'$ such that $L(\mathcal{A}') = (\Sigma^* \setminus L_k(\mathcal{A})) \cap \mathrm{W}_k(\widetilde{\Sigma})$.

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

INPUT: *Concurrent alphabet* $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$, *and* $k \in \mathbb{N}$

QUESTION 1: *Is* $L_k(\mathcal{A})$ $\sim_{\widetilde{\Sigma}}$*-closed?*

QUESTION 2: *Is* $L_k(\mathcal{A})$ *a* k*-phase representation?*

Proof (Question 1)

- From $\mathcal{A}$ construct MVPA $\mathcal{A}'$ over $\widetilde{\Sigma}$ such that $L(\mathcal{A}') = \Sigma^* \setminus L_k(\mathcal{A})$.
- Build MVPA $\mathcal{B}$ over $\widetilde{\Sigma}$ recognizing words of the form $uabv$ with $(a, b) \in I_{\widetilde{\Sigma}}$, $uabv \in L(\mathcal{A})$, and $ubav \in L(\mathcal{A}')$.
- $L_k(\mathcal{A})$ is $\sim_{\widetilde{\Sigma}}$-closed iff $L_k(\mathcal{B}) \neq \emptyset$.
- For Question 2, build $\mathcal{A}'$ such that $L(\mathcal{A}') = (\Sigma^* \setminus L_k(\mathcal{A})) \cap \mathrm{W}_k(\widetilde{\Sigma})$.

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

INPUT: *Concurrent alphabet* $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$, *and* $k \in \mathbb{N}$

QUESTION 1: *Is* $L_k(\mathcal{A})$ $\sim_{\widetilde{\Sigma}}$*-closed?*

QUESTION 2: *Is* $L_k(\mathcal{A})$ *a* k*-phase representation?*

Proof (Question 1)

- From $\mathcal{A}$ construct MVPA $\mathcal{A}'$ over $\widetilde{\Sigma}$ such that $L(\mathcal{A}') = \Sigma^* \setminus L_k(\mathcal{A})$.
- Build MVPA $\mathcal{B}$ over $\widetilde{\Sigma}$ recognizing words of the form $uabv$ with $(a, b) \in I_{\widetilde{\Sigma}}$, $uabv \in L(\mathcal{A})$, and $ubav \in L(\mathcal{A}')$.
- $L_k(\mathcal{A})$ is $\sim_{\widetilde{\Sigma}}$-closed iff $L_k(\mathcal{B}) \neq \emptyset$.
- For Question 2, build $\mathcal{A}'$ such that $L(\mathcal{A}') = (\Sigma^* \setminus L_k(\mathcal{A})) \cap \mathrm{W}_k(\widetilde{\Sigma})$.

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

INPUT: *Concurrent alphabet* $\widetilde{\Sigma}$, MVPA $\mathcal{A}$ *over* $\widetilde{\Sigma}$, *and* $k \in \mathbb{N}$

QUESTION 1: *Is* $L_k(\mathcal{A})$ $\sim_{\widetilde{\Sigma}}$*-closed?*

QUESTION 2: *Is* $L_k(\mathcal{A})$ *a* k*-phase representation?*

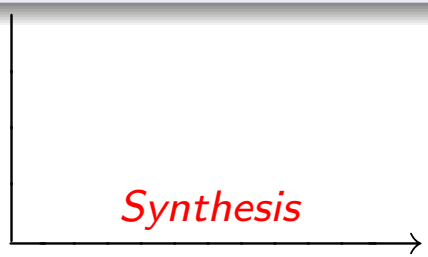
Proof (Question 1)

- From $\mathcal{A}$ construct MVPA $\mathcal{A}'$ over $\widetilde{\Sigma}$ such that $L(\mathcal{A}') = \Sigma^* \setminus L_k(\mathcal{A})$.
- Build MVPA $\mathcal{B}$ over $\widetilde{\Sigma}$ recognizing words of the form $uabv$ with $(a, b) \in I_{\widetilde{\Sigma}}$, $uabv \in L(\mathcal{A})$, and $ubav \in L(\mathcal{A}')$.
- $L_k(\mathcal{A})$ is $\sim_{\widetilde{\Sigma}}$-closed iff $L_k(\mathcal{B}) \neq \emptyset$.
- For Question 2, build $\mathcal{A}'$ such that $L(\mathcal{A}') = (\Sigma^* \setminus L_k(\mathcal{A})) \cap \mathrm{W}_k(\widetilde{\Sigma})$.

Specification formalisms for distributed systems

Specification

finite automata
rational expressions
temporal logics (LTL, CTL, LTrL, ...)
MVPA
monadic second-order logic
...



Implementation

asynchronous automata
message-passing automata
CVPA
...

Specification formalisms for distributed systems

Specification

finite automata

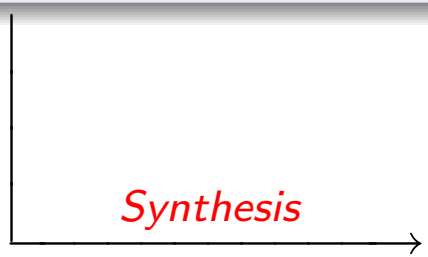
rational expressions

temporal logics (LTL, CTL, LTrL, ...)

MVPA

monadic second-order logic

...



Implementation

asynchronous automata

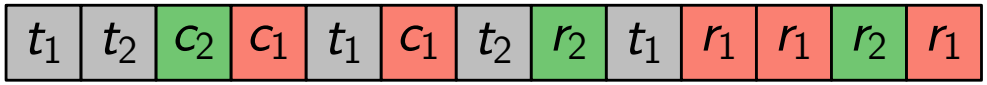
message-passing automata

CVPA

...

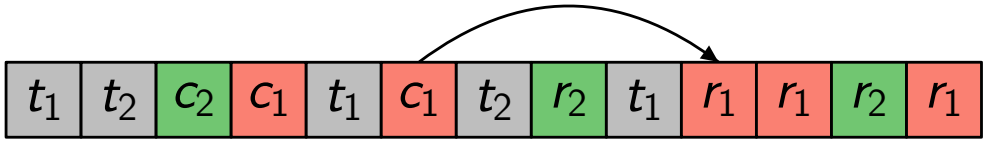
From words to nested Mazurkiewicz traces

word w



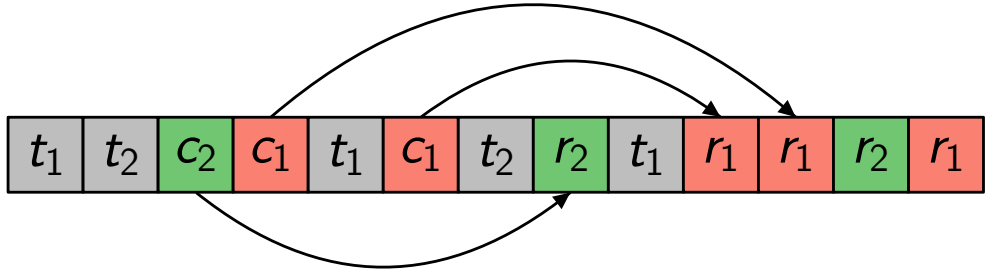
From words to nested Mazurkiewicz traces

word w



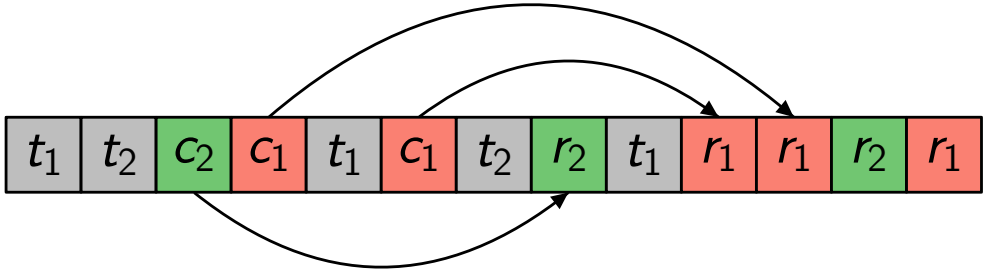
From words to nested Mazurkiewicz traces

nested word w

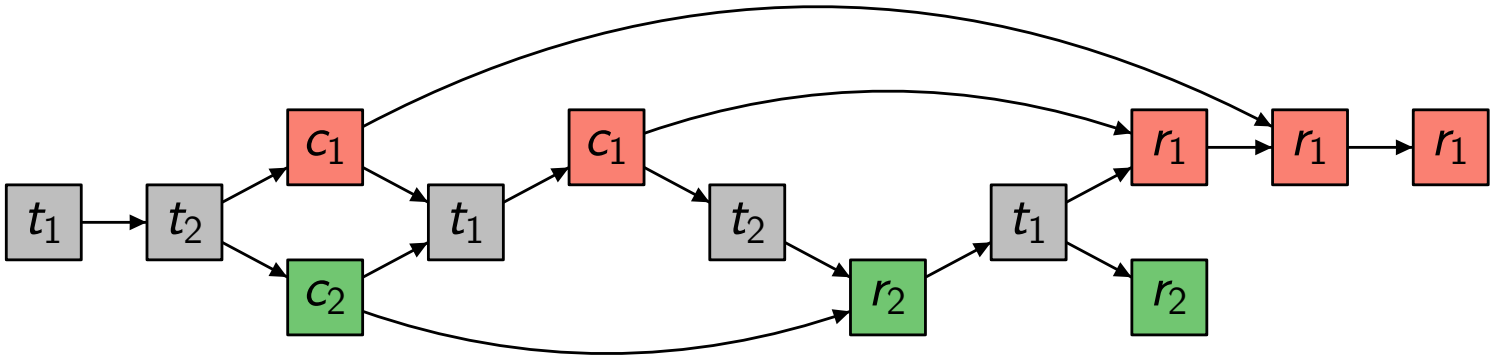


From words to nested Mazurkiewicz traces

nested word w

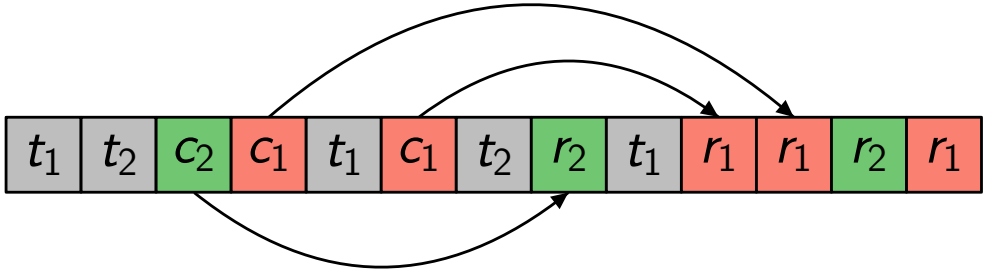


nested trace $T(w)$

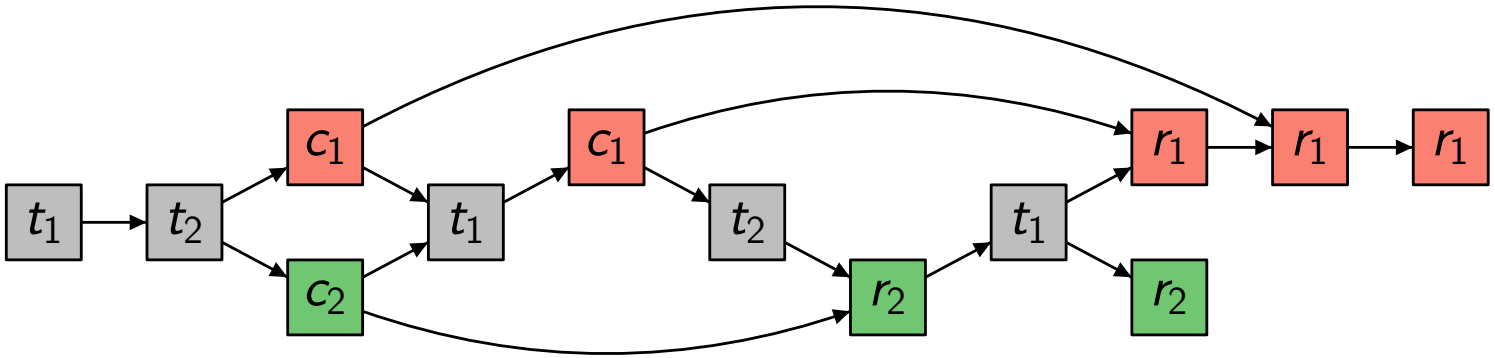


From words to nested Mazurkiewicz traces

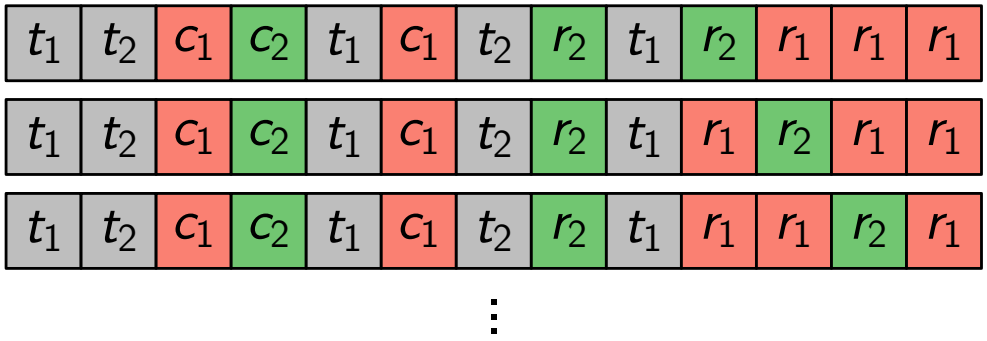
nested word w



nested trace $T(w)$

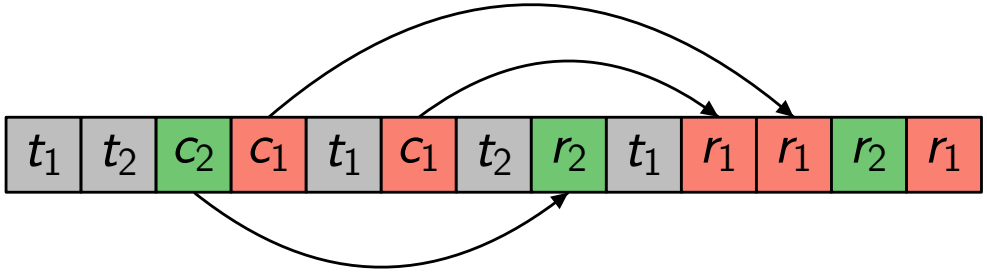


linearizations

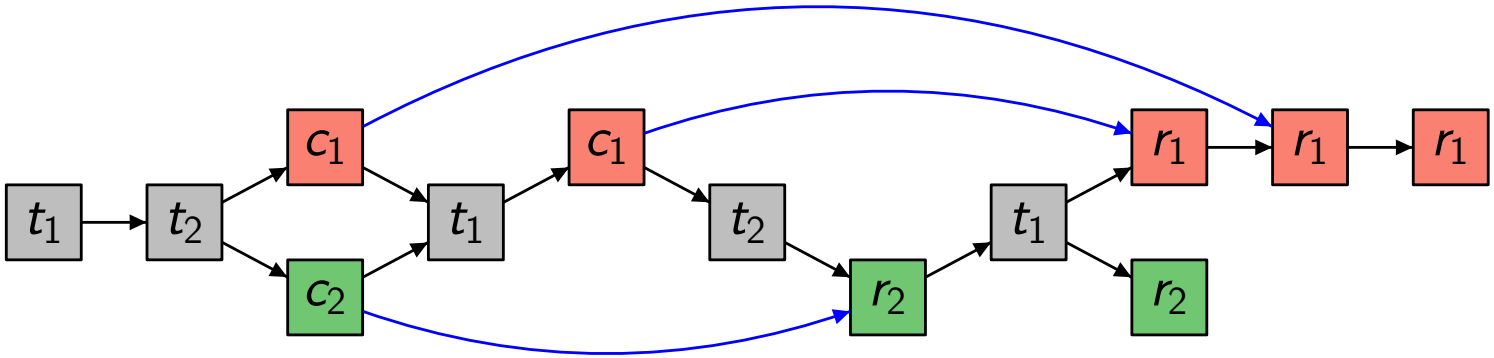


From words to nested Mazurkiewicz traces

nested word w



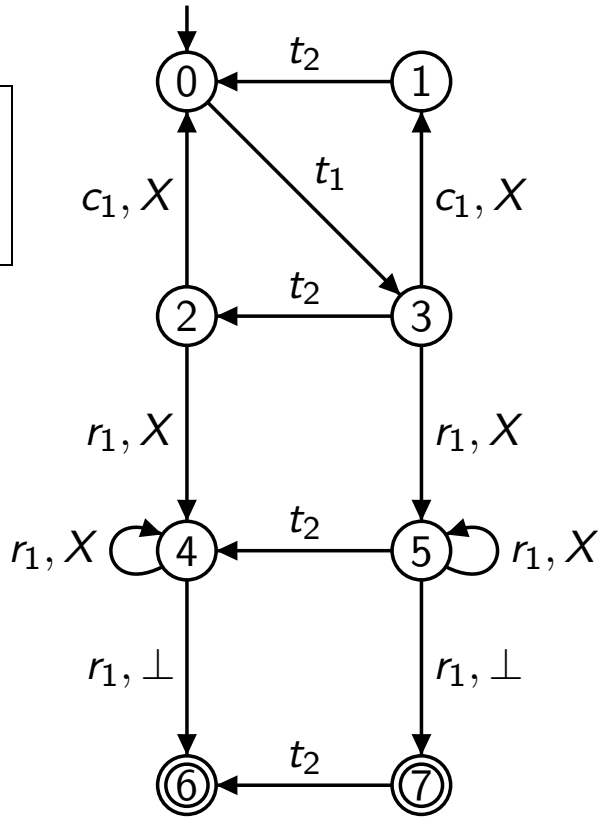
nested trace $T(w)$

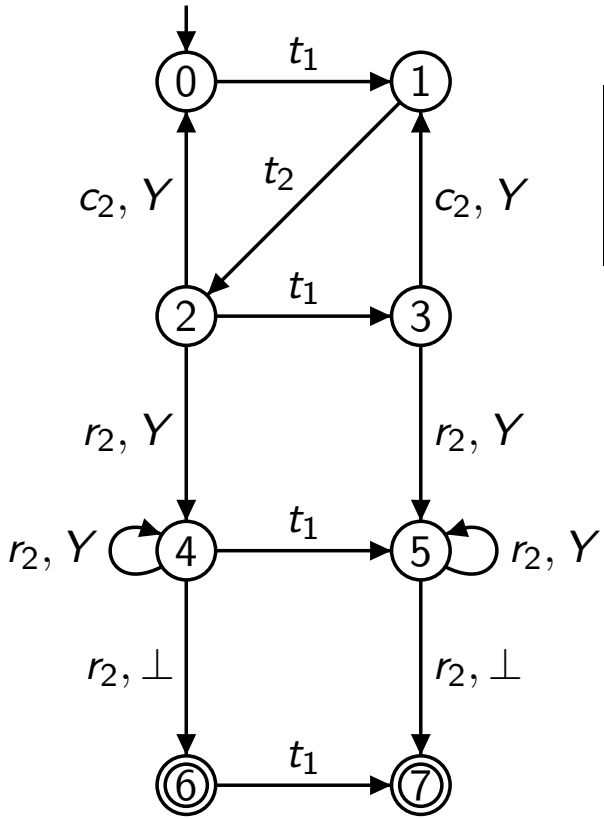


$T(w) = (E, \leq, \mu, \lambda)$ where $\leq, \mu \subseteq E \times E$ and $\lambda : E \to \Sigma$

The nested-trace language of a CVPA $\mathcal{C}$

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$



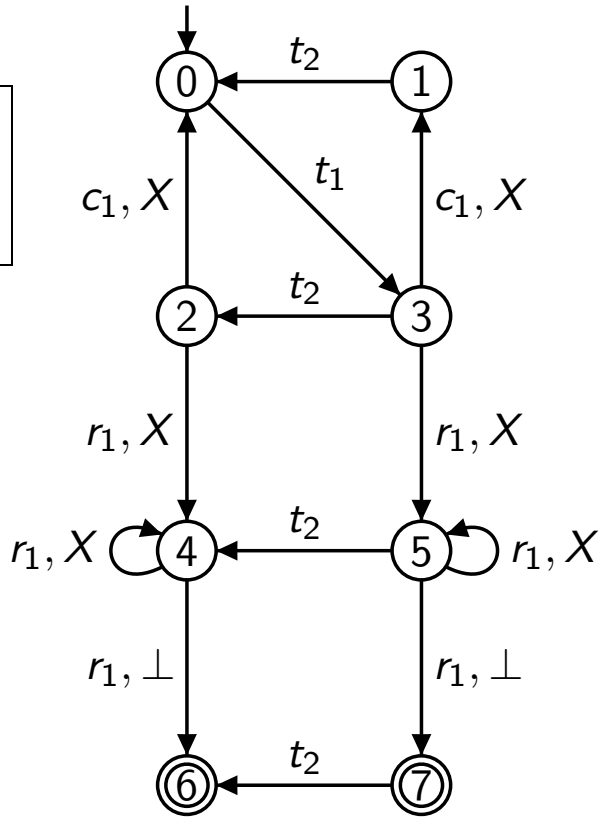


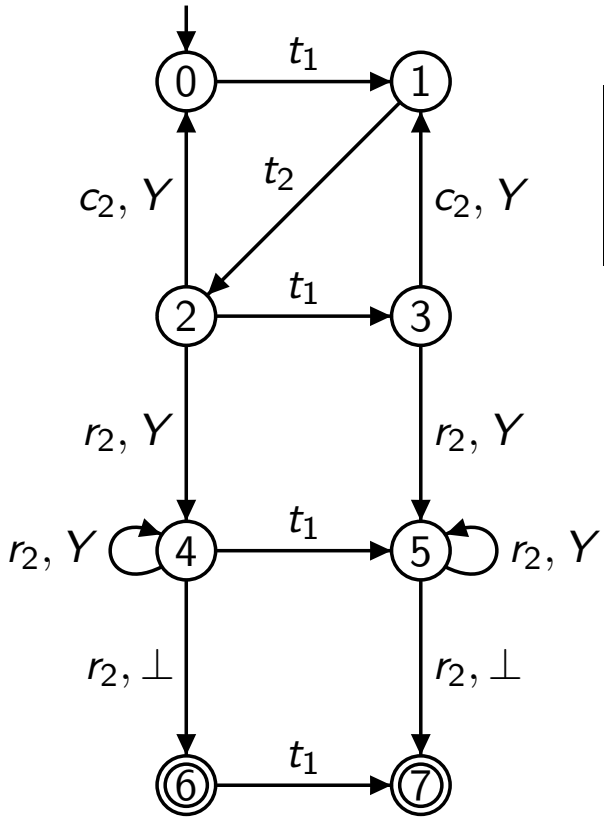
$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$

$$\mathcal{L}(\mathcal{C}) = \{T(w) \mid w \in L(\mathcal{C})\}$$

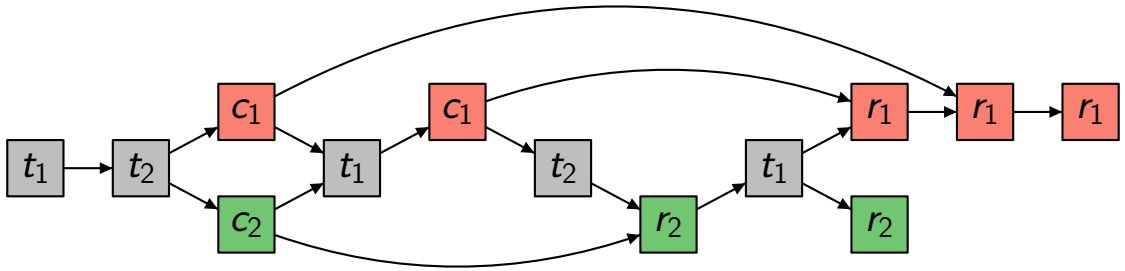
The nested-trace language of a CVPA $\mathcal{C}$

$\Sigma_p^{call} = \{c_1\}$
$\Sigma_p^{ret} = \{r_1\}$
$\Sigma_p^{int} = \{t_1, t_2\}$





$\Sigma_q^{call} = \{c_2\}$
$\Sigma_q^{ret} = \{r_2\}$
$\Sigma_q^{int} = \{t_1, t_2\}$



$\in \mathcal{L}(\mathcal{C})$

MSO logic for nested traces

Definition

Monadic second-order logic $\mathrm{MSO}(\widetilde{\Sigma})$:

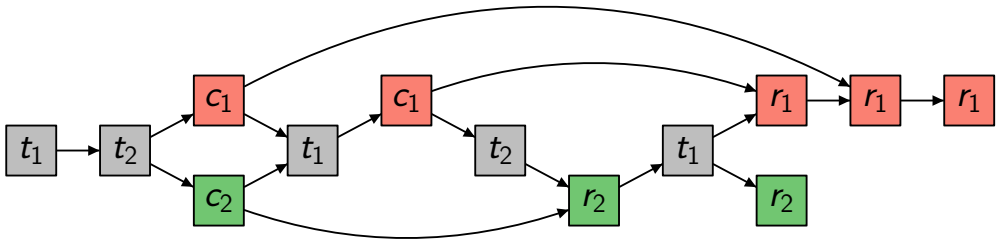
$$\varphi ::= x \leq y \mid (x, y) \in \mu \mid \lambda(x) = a \mid$$
$$x \in X \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x \varphi \mid \exists X \varphi$$

MSO logic for nested traces

Definition

Monadic second-order logic $\mathrm{MSO}(\widetilde{\Sigma})$:

$$\varphi ::= x \leq y \mid (x,y) \in \mu \mid \lambda(x) = a \mid$$
$$x \in X \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x \varphi \mid \exists X \varphi$$



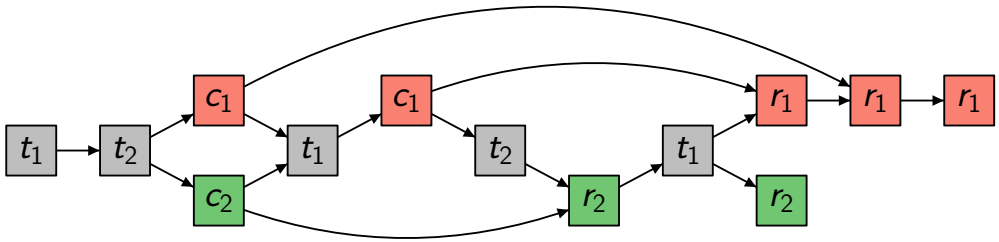
$$\models \quad \forall x \forall y [(\lambda(x) \in \{c_1, c_2\} \wedge \lambda(y) \in \{r_1, r_2\}) \rightarrow x \leq y]$$

MSO logic for nested traces

Definition

Monadic second-order logic $\mathrm{MSO}(\widetilde{\Sigma})$:

$$\varphi ::= x \leq y \mid (x,y) \in \mu \mid \lambda(x) = a \mid$$
$$x \in X \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x \varphi \mid \exists X \varphi$$



$$\models \quad \forall x \forall y[(\lambda(x) \in \{c_1, c_2\} \wedge \lambda(y) \in \{r_1, r_2\}) \rightarrow x \leq y]$$

$$\not\models \quad \forall y[\lambda(y) \in \{r_1, r_2\} \rightarrow \exists x (x,y) \in \mu]$$

Thomas' Theorem

Let $\widetilde{\Sigma}$ be a concurrent alphabet.

Remark

Suppose $\Sigma = \Sigma^{\text{int}}$. Then,

- an MSO formula over nested traces is an MSO formula over traces.
- a CVPA over $\widetilde{\Sigma}$ is an asynchronous automaton over $\widetilde{\Sigma}$.

Theorem ([Tho'90])

Suppose $\Sigma = \Sigma^{\text{int}}$. Let $\mathcal{L}$ be a set of (nested) traces over $\widetilde{\Sigma}$. The following are equivalent:

- *There is a CVPA $\mathcal{C}$ over $\widetilde{\Sigma}$ such that $\mathcal{L}(\mathcal{C}) = \mathcal{L}$.*
- *There is an MSO sentence φ over $\widetilde{\Sigma}$ such that $\mathcal{L}(\varphi) = \mathcal{L}$.*

[Tho'90] Thomas. On logical definability of trace languages. 1990.

CVPA cannot be complemented

Theorem ([B'08])

- MVPA/CVPA *can in general not be complemented.*
- MSO *is strictly more expressive than* MVPA/CVPA.

[B'08] B. On the Expressive Power of 2-Stack Visibly Pushdown Automata. 2008.

CVPA cannot be complemented

Theorem ([B'08])

- MVPA/CVPA *can in general not be complemented.*
- MSO *is strictly more expressive than* MVPA/CVPA.

$\rightsquigarrow$ restrict to k-phase traces

[B'08] B. On the Expressive Power of 2-Stack Visibly Pushdown Automata. 2008.

k-phase traces

Definition

Let $k \in \mathbb{N}$. A k-phase trace (over $\widetilde{\Sigma}$) is a nested trace T such that there is a k-phase word $w \in \Sigma^*$ with $T(w) = T$.

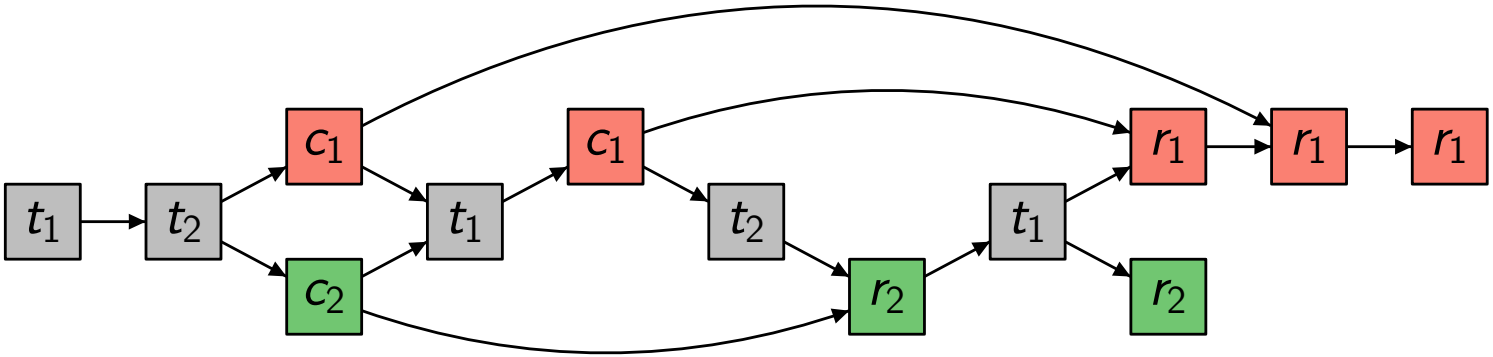
Let $\mathsf{Tr}_k(\widetilde{\Sigma})$ denote the set of k-phase traces.

k-phase traces

Definition

Let $k \in \mathbb{N}$. A k-phase trace (over $\widetilde{\Sigma}$) is a nested trace T such that there is a k-phase word $w \in \Sigma^*$ with $T(w) = T$.

Let $\mathsf{Tr}_k(\widetilde{\Sigma})$ denote the set of k-phase traces.



is a 2-phase trace

MSO characterization of CVPA wrt. k-phase traces

Theorem

For every CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$, *there is* $\varphi \in \mathrm{MSO}(\widetilde{\Sigma})$ *such that* $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C})$.

Theorem

Let $k \in \mathbb{N}$. *For every* $\varphi \in \mathrm{MSO}(\widetilde{\Sigma})$, *there is a* CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$ *such that*

$$\mathcal{L}(\mathcal{C}) = \mathcal{L}(\varphi) \cap \mathsf{Tr}_k(\widetilde{\Sigma})$$

Proof

- By induction.
- Lemma: If $\mathcal{L} \subseteq \mathsf{Tr}_k(\widetilde{\Sigma})$ is recognizable, then so is $\overline{\mathcal{L}} \cap \mathsf{Tr}_k(\widetilde{\Sigma})$.
- $\mathsf{Tr}_k(\widetilde{\Sigma})$ is recognizable.
- Atomic formulas standard.
- CVPA are closed under union, intersection, and projection.

MSO characterization of CVPA wrt. k-phase traces

Theorem

For every CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$, *there is* $\varphi \in \mathrm{MSO}(\widetilde{\Sigma})$ *such that* $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C})$.

Theorem

Let $k \in \mathbb{N}$. *For every* $\varphi \in \mathrm{MSO}(\widetilde{\Sigma})$, *there is a* CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$ *such that*

$$\mathcal{L}(\mathcal{C}) = \mathcal{L}(\varphi) \cap \mathsf{Tr}_k(\widetilde{\Sigma})$$

Proof

- By induction.
- Lemma: If $\mathcal{L} \subseteq \mathsf{Tr}_k(\widetilde{\Sigma})$ is recognizable, then so is $\overline{\mathcal{L}} \cap \mathsf{Tr}_k(\widetilde{\Sigma})$.
- $\mathsf{Tr}_k(\widetilde{\Sigma})$ is recognizable.
- Atomic formulas standard.
- CVPA are closed under union, intersection, and projection.

MSO characterization of CVPA wrt. k-phase traces

Theorem

For every CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$, *there is* $\varphi \in \mathrm{MSO}(\widetilde{\Sigma})$ *such that* $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C})$.

Theorem

Let $k \in \mathbb{N}$. *For every* $\varphi \in \mathrm{MSO}(\widetilde{\Sigma})$, *there is a* CVPA $\mathcal{C}$ *over* $\widetilde{\Sigma}$ *such that*

$$\mathcal{L}(\mathcal{C}) = \mathcal{L}(\varphi) \cap \mathsf{Tr}_k(\widetilde{\Sigma})$$

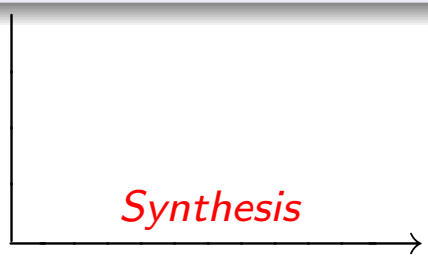
Proof

- By induction.
- Lemma: If $\mathcal{L} \subseteq \mathsf{Tr}_k(\widetilde{\Sigma})$ is recognizable, then so is $\overline{\mathcal{L}} \cap \mathsf{Tr}_k(\widetilde{\Sigma})$.
- $\mathsf{Tr}_k(\widetilde{\Sigma})$ is recognizable.
- Atomic formulas standard.
- CVPA are closed under union, intersection, and projection.

Summary

Specification

finite automata
rational expressions
temporal logics (LTL, CTL, LTrL, ...)
MVPA
MSO
...



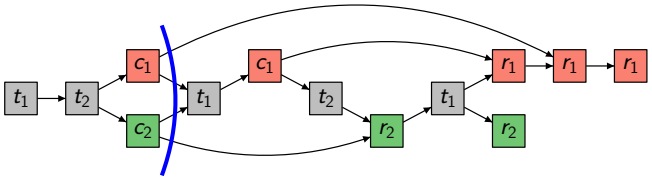
Implementation

asynchronous automata
message-passing automata
CVPA
...

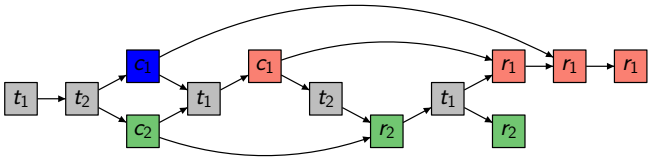
Future work: Temporal logic for nested traces

Combine works on

- temporal logic for nested words [AAB$^+$'08]
- temporal logic for traces
  - global [DG'02], interpreted over configurations:



  - local [GK'07], interpreted over events:



[AAB$^+$'08] Alur & Arenas & Barcelo & Etessami & Immerman & Libkin. First-Order and Temporal Logics for Nested Words. 2008.

[DG'02] Diekert & Gastin. LTL is expressively complete for Mazurkiewicz traces. 2002.

[GK'07] Gastin & Kuske. Uniform satisfiability in PSPACE for local temporal logics over Mazurkiewicz traces. 2007.

Future work: Nested MSCs

Extend Zielonka-like theorems and logical characterizations of

- unbounded message-passing automata [BL'06]
- existentially bounded message passing automata [GKM'06]
- universally bounded message-passing automata [HMN$^+$'05]

by visibly pushdown stacks.

[BL'06] B. & Leucker. Message-Passing Automata are expressively equivalent to EMSO Logic. 2006.

[GKM'06] Genest & Kuske & Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. 2006.

[HMNST'05] Henriksen & Mukund & Narayan Kumar & Sohoni & Thiagarjan. A Theory of Regular MSC Languages. 2005.

Thank you!