

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

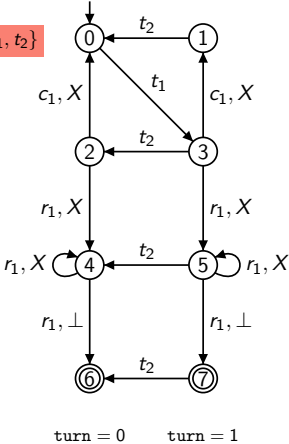
```

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

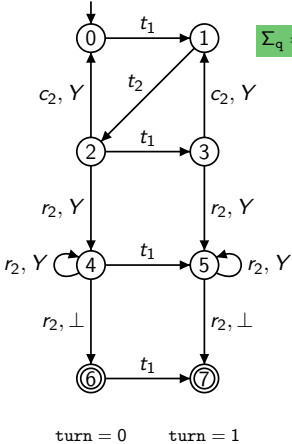
```

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

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



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

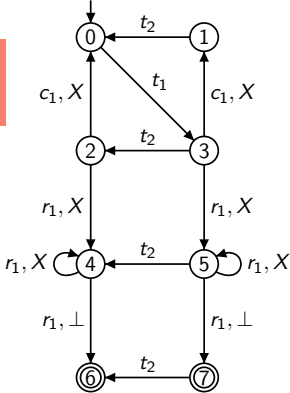


Example of a concurrent recursive program

```

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

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

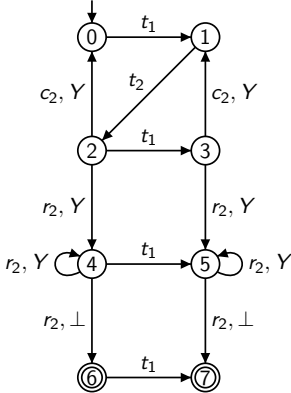


turn = 0 turn = 1

```

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

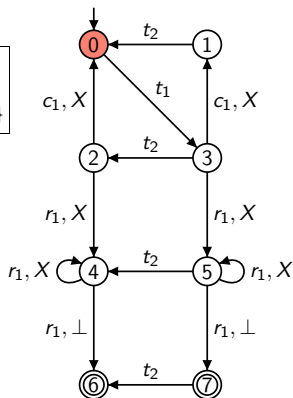
$\Sigma_q^{call} = \{c_2\}$
 $\Sigma_q^{ret} = \{r_2\}$
 $\Sigma_q^{int} = \{t_1, t_2\}$



turn = 0 turn = 1

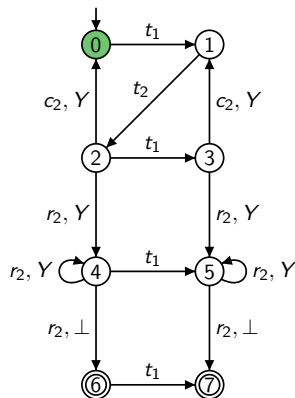
Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



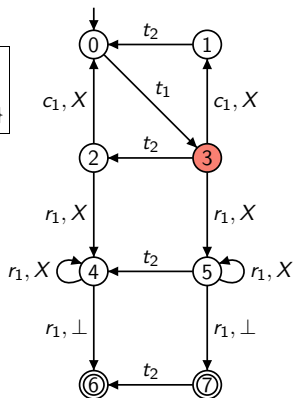
t_1

$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



Concurrent visibly pushdown automaton

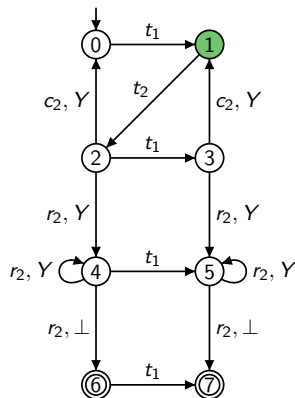
$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



\perp

t_1 t_2

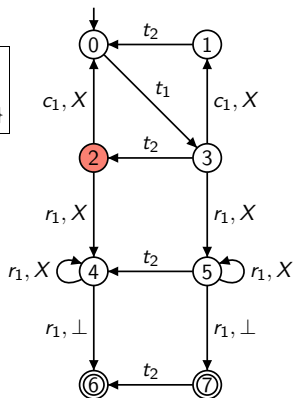
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



\perp

Concurrent visibly pushdown automaton

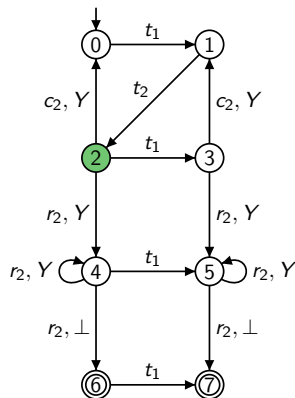
$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



⊥

t_1 t_2 c_2

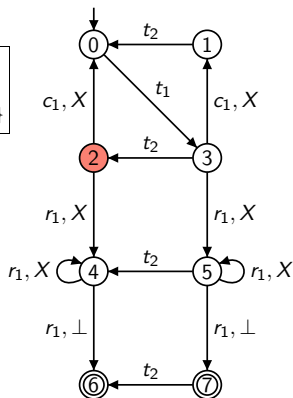
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



⊥

Concurrent visibly pushdown automaton

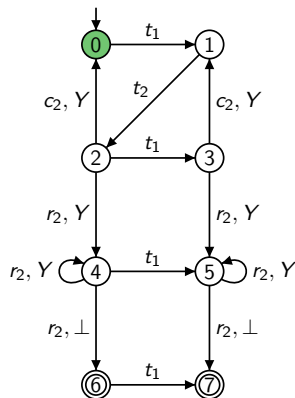
$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



\perp

t_1 t_2 c_2 c_1

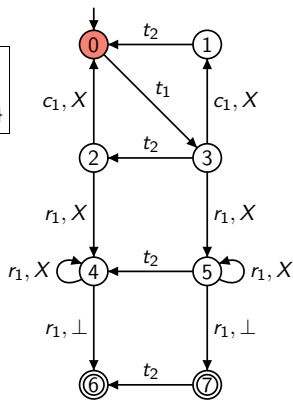
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



Y
 \perp

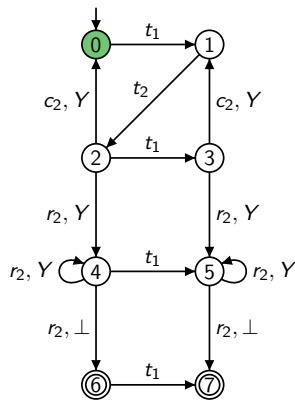
Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



X
⊥

$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$

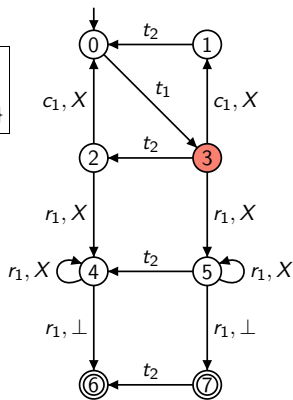


Y
⊥

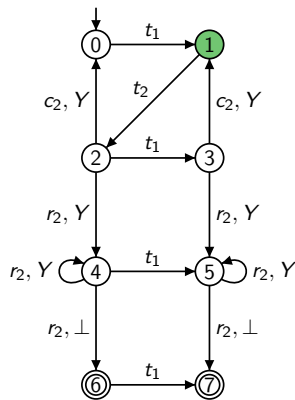
t_1 t_2 c_2 c_1 t_1

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



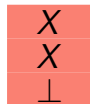
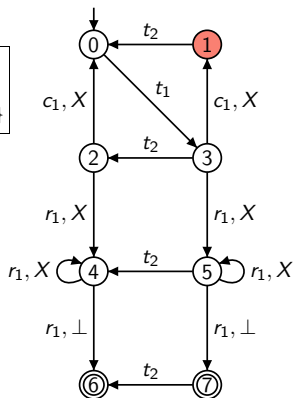
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



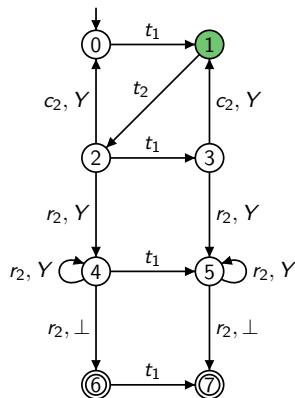
t_1 t_2 c_2 c_1 t_1 c_1

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



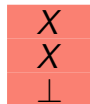
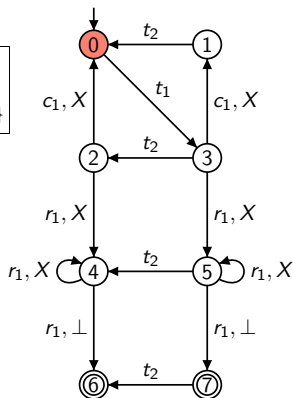
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



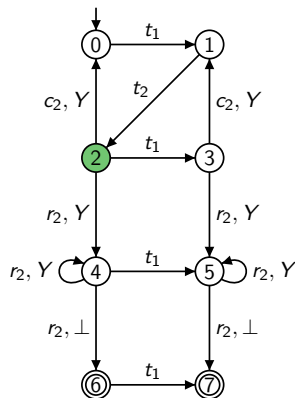
t_1 t_2 c_2 c_1 t_1 c_1 t_2

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



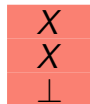
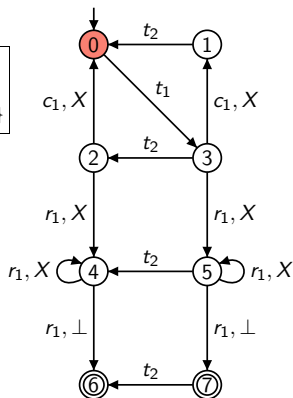
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



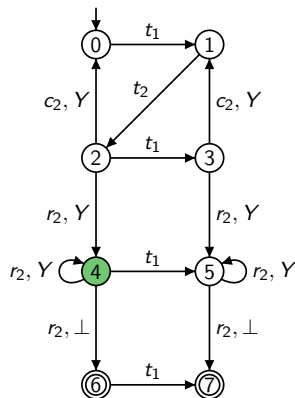
t_1 t_2 c_2 c_1 t_1 c_1 t_2 r_2

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



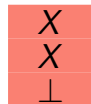
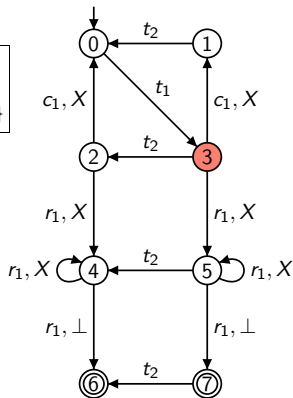
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



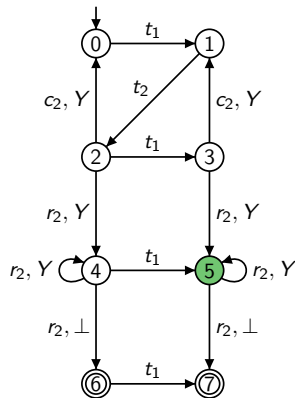
t_1 t_2 c_2 c_1 t_1 c_1 t_2 r_2 t_1

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



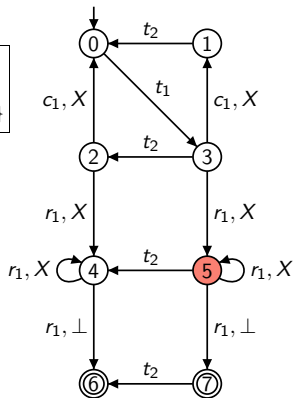
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



t_1 t_2 c_2 c_1 t_1 c_1 t_2 r_2 t_1 r_1

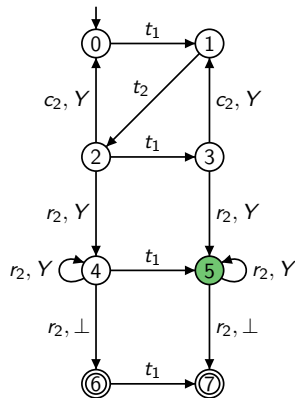
Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



X
⊥

$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$

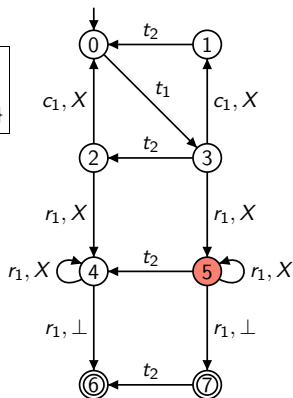


⊥

t_1 t_2 c_2 c_1 t_1 c_1 t_2 r_2 t_1 r_1 r_1

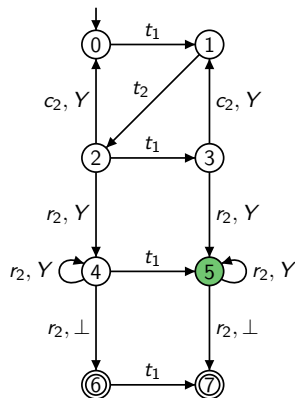
Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



⊥

$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$

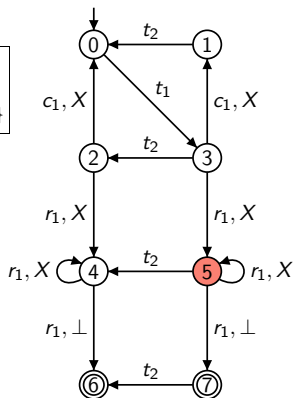


⊥

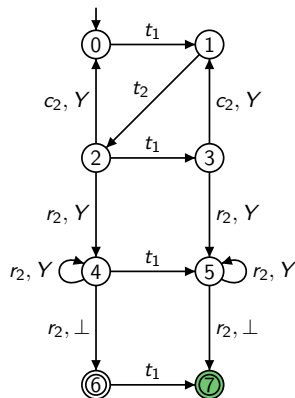
t_1 t_2 c_2 c_1 t_1 c_1 t_2 r_2 t_1 r_1 r_1 r_2

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



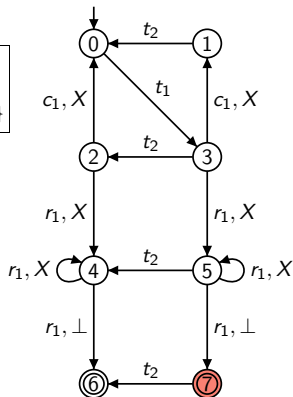
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



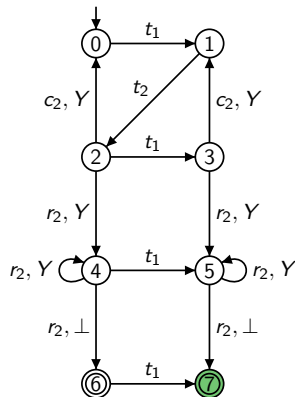
t_1 t_2 c_2 c_1 t_1 c_1 t_2 r_2 t_1 r_1 r_1 r_2 r_1

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



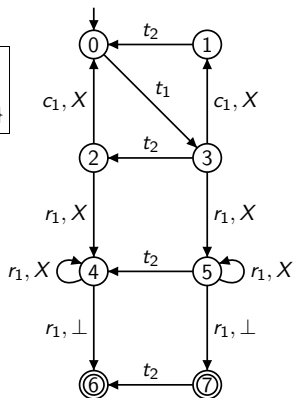
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



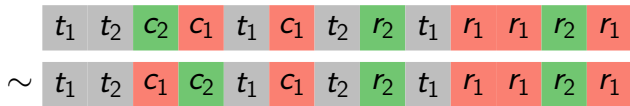
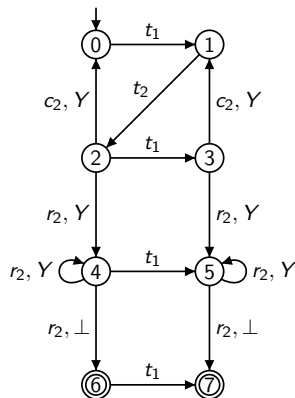
t_1 t_2 c_2 c_1 t_1 c_1 t_2 r_2 t_1 r_1 r_1 r_2 r_1

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$

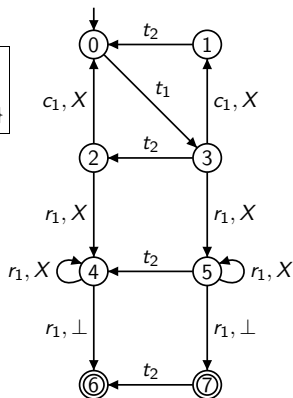


$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$

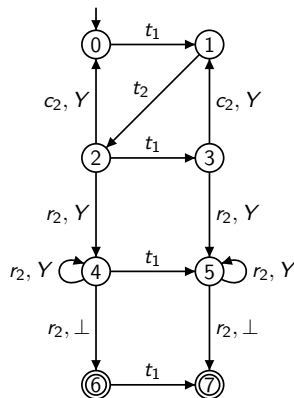


Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$

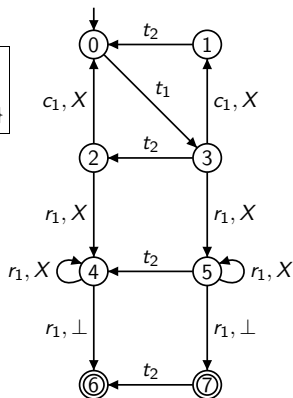


$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



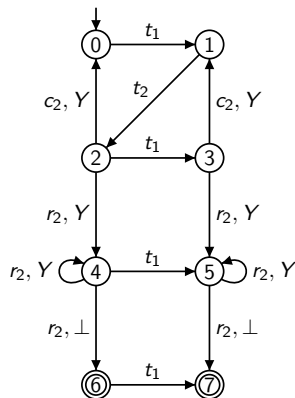
Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



⊥

$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



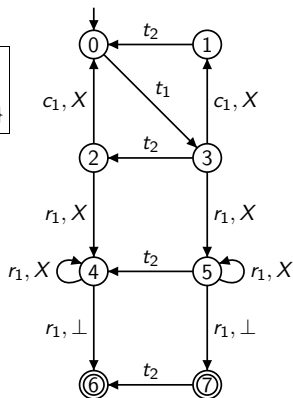
⊥

$$L(\mathcal{A}_p) = \{ t_1 ((c_1 t_2 + t_2 c_1) t_1)^n r_1^i (t_2 + \varepsilon) r_1^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

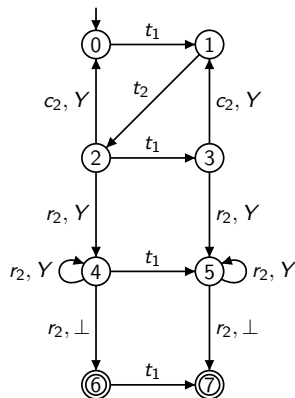
$$L(\mathcal{A}_q) = \{ t_1 t_2 ((c_2 t_1 + t_1 c_2) t_2)^n r_2^i (t_1 + \varepsilon) r_2^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

Concurrent visibly pushdown automaton

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



$$L(\mathcal{A}_p) = \{ t_1 ((c_1 t_2 + t_2 c_1) t_1)^n r_1^i (t_2 + \varepsilon) r_1^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

$$L(\mathcal{A}_q) = \{ t_1 t_2 ((c_2 t_1 + t_1 c_2) t_2)^n r_2^i (t_1 + \varepsilon) r_2^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

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

The architecture of a concurrent recursive program

Definition

We fix the following parameters:

- \mathcal{P} a finite set of **processes**
- $\tilde{\Sigma} = ((\Sigma_p^{\text{call}}, \Sigma_p^{\text{ret}}, \Sigma_p^{\text{int}}))_{p \in \mathcal{P}}$ a **concurrent pushdown alphabet**:
 - ▶ $\Sigma_p^{\text{call}}, \Sigma_p^{\text{ret}}, \Sigma_p^{\text{int}}$ 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
- $\text{proc}(a) = \{p \in \mathcal{P} \mid a \in \Sigma_p\}$ 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**
- $\tilde{\Sigma} = ((\Sigma_p^{\text{call}}, \Sigma_p^{\text{ret}}, \Sigma_p^{\text{int}}))_{p \in \mathcal{P}}$ a **concurrent pushdown alphabet**:
 - ▶ $\Sigma_p^{\text{call}}, \Sigma_p^{\text{ret}}, \Sigma_p^{\text{int}}$ 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
- $\text{proc}(a) = \{p \in \mathcal{P} \mid a \in \Sigma_p\}$ processes involved in $a \in \Sigma$

Definition of concurrent visibly pushdown automaton

Definition

A **concurrent visibly pushdown automaton** (CVPA) over $\tilde{\Sigma}$ 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 \perp
 - $\delta_a \subseteq S_a \times (\Gamma \setminus \{\perp\}) \times S_a$ 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 $S_a = \prod_{p \in \text{proc}(a)} S_p$ 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 concurrent visibly pushdown automaton

Definition

A **concurrent visibly pushdown automaton** (CVPA) over $\tilde{\Sigma}$ 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 \perp
 - $\delta_a \subseteq S_a \times (\Gamma \setminus \{\perp\}) \times S_a$ 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 $S_a = \prod_{p \in \text{proc}(a)} S_p$ 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 concurrent visibly pushdown automaton

Definition

A **concurrent visibly pushdown automaton** (CVPA) over $\tilde{\Sigma}$ 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 \perp
- $\delta_a \subseteq S_a \times (\Gamma \setminus \{\perp\}) \times S_a$ 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 $S_a = \prod_{p \in \text{proc}(a)} S_p$ 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 concurrent visibly pushdown automaton

Definition

A **concurrent visibly pushdown automaton** (CVPA) over $\tilde{\Sigma}$ 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 \perp
- $\delta_a \subseteq S_a \times (\Gamma \setminus \{\perp\}) \times S_a$ 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 $S_a = \prod_{p \in \text{proc}(a)} S_p$ 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} :

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

- global transition:

$$(s, \sigma) \xRightarrow{a} (s', \sigma')$$

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

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

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

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} :

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

- global transition:

$$(s, \sigma) \xRightarrow{a} (s', \sigma')$$

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

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

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

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 $\tilde{\Sigma}$ is a structure

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

$$\begin{aligned} \bullet \Delta \subseteq & S \times \Sigma^{\text{call}} \times (\Gamma \setminus \{\perp\}) \times S \\ & \cup S \times \Sigma^{\text{ret}} \times \Gamma \times S \\ & \cup S \times \Sigma^{\text{int}} \times S \end{aligned}$$

- set of configurations: $S \times \prod_{p \in \mathcal{P}} (\Gamma^* \setminus \{\perp\}) \{\perp\}$
- 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 $\tilde{\Sigma}$ is a structure

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

$$\begin{aligned} \bullet \Delta \subseteq & S \times \Sigma^{\text{call}} \times (\Gamma \setminus \{\perp\}) \times S \\ & \cup S \times \Sigma^{\text{ret}} \times \Gamma \times S \\ & \cup S \times \Sigma^{\text{int}} \times S \end{aligned}$$

- set of configurations: $S \times \prod_{p \in \mathcal{P}} (\Gamma^* \setminus \{\perp\}) \{\perp\}$
- 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
...

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
...

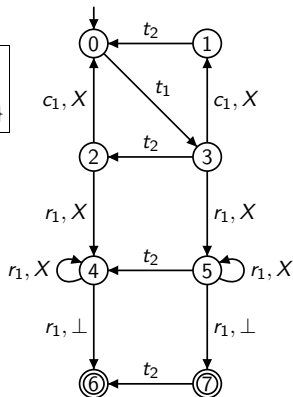
Synthesis

Implementation

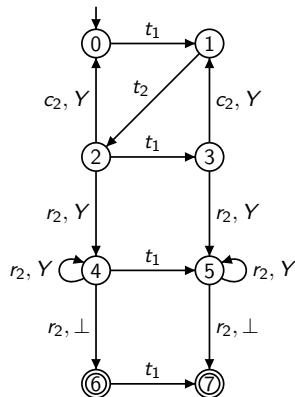
asynchronous automata
message-passing automata
CVPA
...

Closure property of CVPA

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



Closure property of CVPA

Definition

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

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

Definition

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

Lemma

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

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

Closure property of CVPA

Definition

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

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

Definition

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

Lemma

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

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

Closure property of CVPA

Definition

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

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

Definition

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

Lemma

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

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

Zielonka's Theorem

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

Remark

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

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

Theorem ([Zie'87])

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

There is a CVPA \mathcal{C} over $\tilde{\Sigma}$ 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 $\tilde{\Sigma}$ such that $L(\mathcal{A})$ is $\sim_{\tilde{\Sigma}}$ -closed.

There is a CVPA \mathcal{C} over $\tilde{\Sigma}$ 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 $\tilde{\Sigma}$.

From MVPA to CVPA

Theorem

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

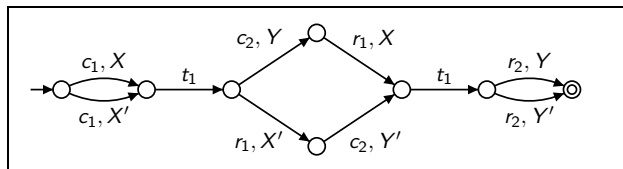
There is a CVPA \mathcal{C} over $\tilde{\Sigma}$ such that $L(\mathcal{C}) = L(\mathcal{A})$. The size of \mathcal{C} is

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

Proof

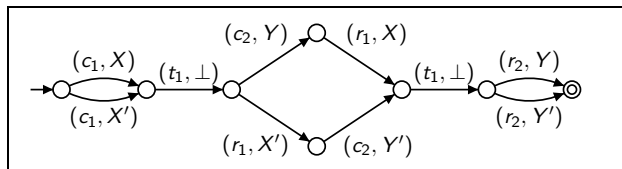
- Interpret $\mathcal{A} = (S, \Gamma, \Delta, \iota, F)$ as finite automaton over $\Sigma \times \Gamma$.
- Apply Zielonka's Theorem to obtain CVPA \mathcal{C} over $((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.
- Interpret \mathcal{C} as CVPA over $\tilde{\Sigma}$.

From MVPA to CVPA



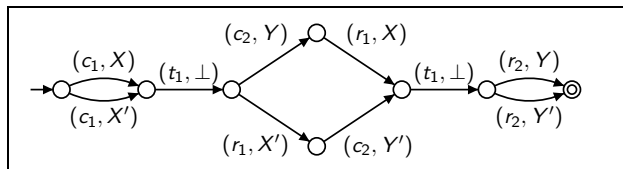
MVPA \mathcal{A}

From MVPA to CVPA



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

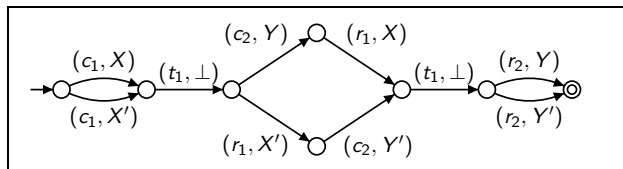
From MVPA to CVPA



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

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

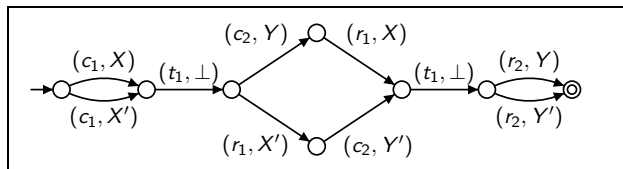
From MVPA to CVPA



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

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

From MVPA to CVPA

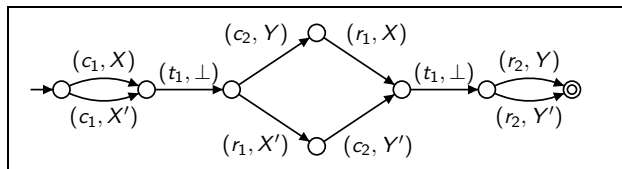


finite automaton \mathcal{B}_1
over $\Sigma \times \Gamma$

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

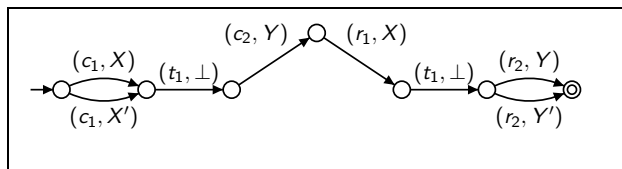
[Kus'07] Kuske. *Weighted asynchronous cellular automata*. 2007.

From MVPA to CVPA



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

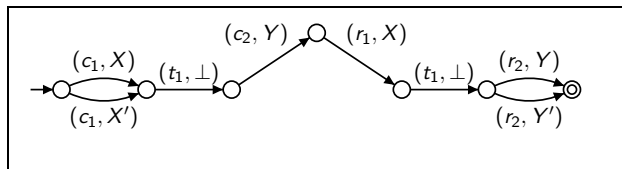
- Consider the concurrent pushdown alphabet $\tilde{\Omega} = ((\emptyset, \emptyset, \Sigma_p \times \Gamma))_{p \in \mathcal{P}}$.
- Fix lexicographic order $c_2 <_{\text{lex}} r_1$.
- There is a **loop-connected** finite automaton for the normal forms of \mathcal{B}_1 wrt. $<_{\text{lex}}$ of size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



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

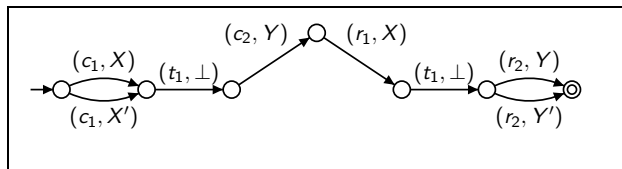
[Kus'07] Kuske. *Weighted asynchronous cellular automata*. 2007.

From MVPA to CVPA



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

From MVPA to CVPA

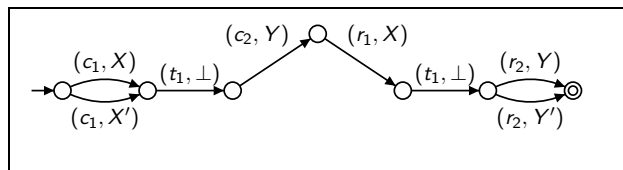


loop-connected
finite automaton \mathcal{B}_2
for normal forms

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

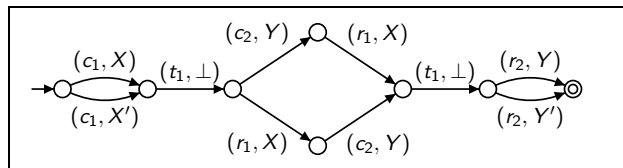
[MP'99] Muscholl & Peled. [Message sequence graphs and decision problems on Mazurkiewicz traces](#). 1999.
[Kus'07] Kuske. [Weighted asynchronous cellular automata](#). 2007.

From MVPA to CVPA



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

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

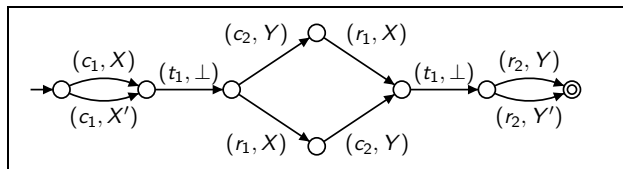


I-diamond
finite automaton \mathcal{B}_3
for $[L(\mathcal{B}_2)]_{\sim_{\bar{\Omega}}}$

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

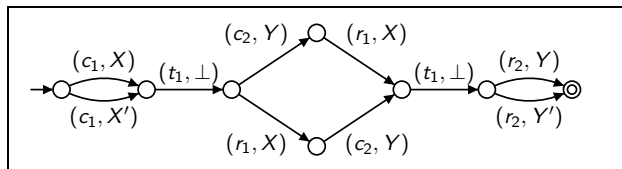
[Kus'07] Kuske. [Weighted asynchronous cellular automata](#). 2007.

From MVPA to CVPA



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

From MVPA to CVPA

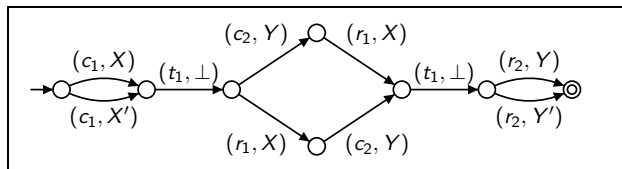


I-diamond
finite automaton \mathcal{B}_3
for $[L(\mathcal{B}_2)]_{\sim_{\tilde{\Omega}}}$

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

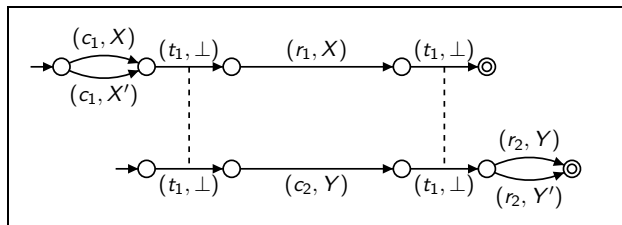
[GM'06] Genest & Muscholl. Constructing Exponential-size Deterministic Zielonka Automata. 2006.

From MVPA to CVPA



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

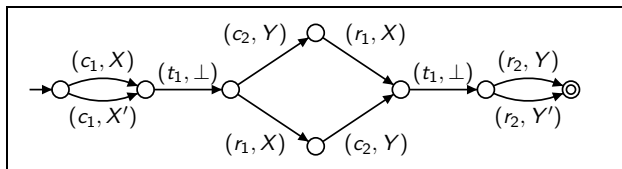
- There is a CVPA \mathcal{C} over $\tilde{\Omega}$ such that $L(\mathcal{C}) = L(\mathcal{B}_3)$.
It is of size $\exp(|\mathcal{B}_3|, |\Sigma|)$ [GM'06]



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

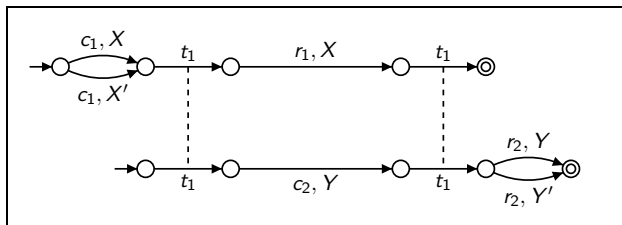
[GM'06] Genest & Muscholl. Constructing Exponential-size Deterministic Zielonka Automata. 2006.

From MVPA to CVPA



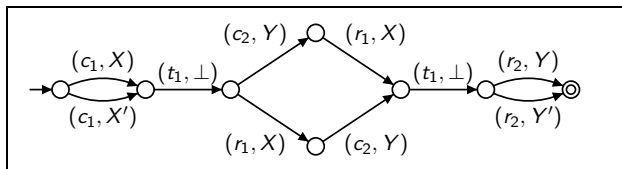
I-diamond
finite automaton \mathcal{B}_3
for $[L(\mathcal{B}_2)]_{\sim_{\tilde{\Omega}}}$

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



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

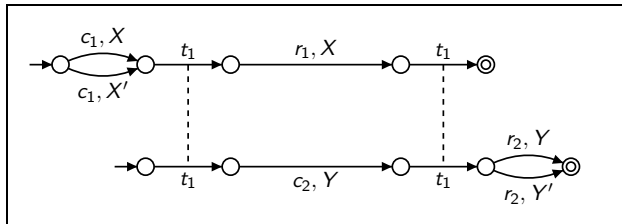
From MVPA to CVPA



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

- There is a CVPA \mathcal{C} over $\tilde{\Omega}$ 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 $\tilde{\Sigma}$
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 $\Sigma = \Sigma^{\text{int}}$. Let $L \subseteq \Sigma^*$ be a $\sim_{\tilde{\Sigma}}$ -closed regular language.

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

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

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

INPUT: MVPA \mathcal{A} over $\tilde{\Sigma}$
QUESTION: Is $L(\mathcal{A}) \sim_{\tilde{\Sigma}}$ -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 $\tilde{\Sigma}$ such that $L(\mathcal{A})$ is $\sim_{\tilde{\Sigma}}$ -closed.

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

Theorem

The following problem is *undecidable*:

INPUT: Concurrent alphabet $\tilde{\Sigma}$ and MVPA \mathcal{A} over $\tilde{\Sigma}$

QUESTION: Is $L(\mathcal{A})$ $\sim_{\tilde{\Sigma}}$ -closed?

Restriction to k -phase words

Definition

Let $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called a **k -phase word** over $\tilde{\Sigma}$ if it can be written as $w_1 \cdot \dots \cdot w_k$ 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 $\tilde{\Sigma}$ if it can be written as $w_1 \cdot \dots \cdot w_k$ 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

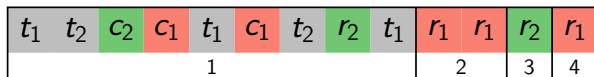
\sim t_1 t_2 c_1 c_2 t_1 c_1 t_2 r_2 t_1 r_1 r_1 r_1 r_2

\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

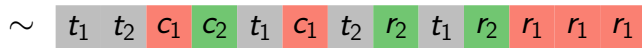
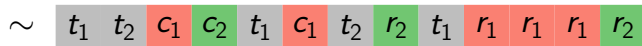
Restriction to k -phase words

Definition

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



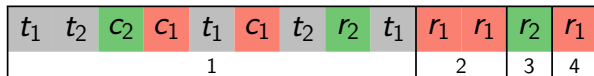
4-phase word



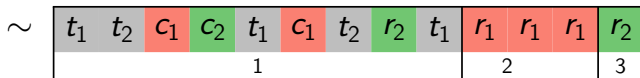
Restriction to k -phase words

Definition

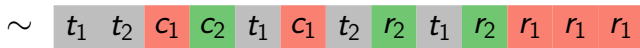
Let $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called a **k -phase word** over $\tilde{\Sigma}$ if it can be written as $w_1 \cdot \dots \cdot w_k$ where $w_i \in (\Sigma^{\text{call}} \cup \Sigma^{\text{int}} \cup \Sigma_p^{\text{ret}})^*$ for some $p \in \mathcal{P}$.



4-phase word



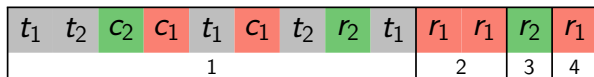
3-phase word



Restriction to k -phase words

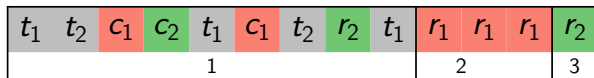
Definition

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



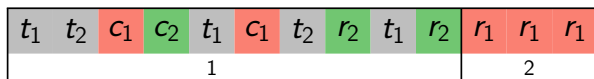
4-phase word

\sim



3-phase word

\sim



2-phase word

MVPA and k -phase words

Notation

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

Theorem ([LMP'07])

The following problem is decidable:

INPUT: Concurrent alphabet $\tilde{\Sigma}$, MVPA \mathcal{A} over $\tilde{\Sigma}$, 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 $\tilde{\Sigma}$. One can effectively construct an MVPA \mathcal{A}' over $\tilde{\Sigma}$ 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 $W_k(\tilde{\Sigma})$ denote the set of k -phase words over $\tilde{\Sigma}$.
- For an MVPA \mathcal{A} , let $L_k(\mathcal{A}) = L(\mathcal{A}) \cap W_k(\tilde{\Sigma})$.

Theorem ([LMP'07])

The following problem is decidable:

INPUT: Concurrent alphabet $\tilde{\Sigma}$, MVPA \mathcal{A} over $\tilde{\Sigma}$, 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 $\tilde{\Sigma}$. One can effectively construct an MVPA \mathcal{A}' over $\tilde{\Sigma}$ 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 $W_k(\tilde{\Sigma})$ denote the set of k -phase words over $\tilde{\Sigma}$.
- For an MVPA \mathcal{A} , let $L_k(\mathcal{A}) = L(\mathcal{A}) \cap W_k(\tilde{\Sigma})$.

Theorem ([LMP'07])

The following problem is decidable:

INPUT: Concurrent alphabet $\tilde{\Sigma}$, MVPA \mathcal{A} over $\tilde{\Sigma}$, 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 $\tilde{\Sigma}$. One can effectively construct an MVPA \mathcal{A}' over $\tilde{\Sigma}$ 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 $L \subseteq W_k(\tilde{\Sigma})$ is a **k -phase representation** if, for all $u, v \in \Sigma^*$ and $(a, b) \in I_{\tilde{\Sigma}}$ with $\{uabv, ubav\} \subseteq W_k(\tilde{\Sigma})$, we have $uabv \in L$ iff $ubav \in L$.

A decidable sufficient criterion for implementability

Definition

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

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

is a 2-phase representation.

A decidable sufficient criterion for implementability

Definition

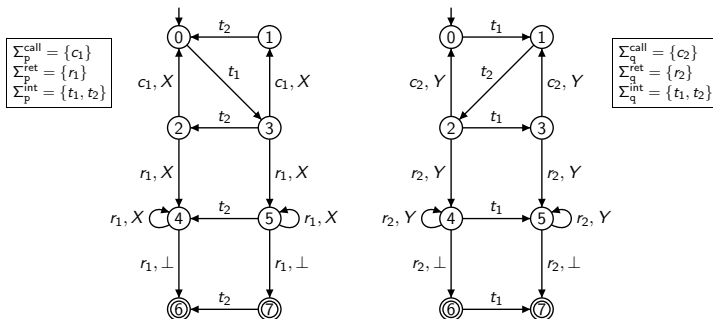
A set $L \subseteq W_k(\tilde{\Sigma})$ is a **k -phase representation** if, for all $u, v \in \Sigma^*$ and $(a, b) \in I_{\tilde{\Sigma}}$ with $\{uabv, ubav\} \subseteq W_k(\tilde{\Sigma})$, we have $uabv \in L$ iff $ubav \in L$.

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

is a 2-phase representation.

Any $\sim_{\tilde{\Sigma}}$ -closed subset of $W_k(\tilde{\Sigma})$ is a k -phase representation.

The closure of a k -phase representation

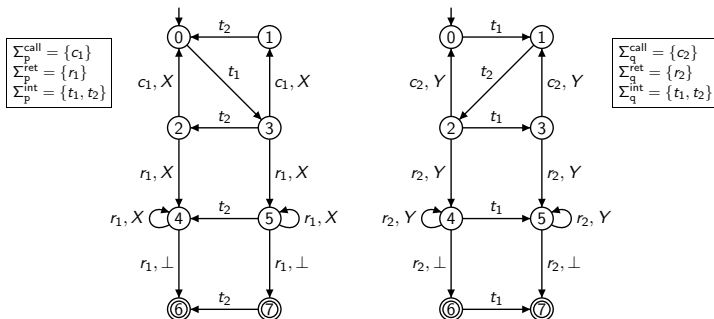


$$L(\mathcal{A}_p) = \{ t_1 ((c_1 t_2 + t_2 c_1) t_1)^n r_1^i (t_2 + \varepsilon) r_1^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

$$L(\mathcal{A}_q) = \{ t_1 t_2 ((c_2 t_1 + t_1 c_2) t_2)^n r_2^i (t_1 + \varepsilon) r_2^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

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

The closure of a k -phase representation



$$L(\mathcal{A}_p) = \{ t_1 ((c_1 t_2 + t_2 c_1) t_1)^n r_1^i (t_2 + \varepsilon) r_1^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

$$L(\mathcal{A}_q) = \{ t_1 t_2 ((c_2 t_1 + t_1 c_2) t_2)^n r_2^i (t_1 + \varepsilon) r_2^j \mid n, i, j \in \mathbb{N}, i + j = n + 1 \geq 2 \}$$

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

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

From MVPA to CVPA

Theorem

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

There is a CVPA \mathcal{C} over $\tilde{\Sigma}$ such that $L(\mathcal{C}) = [L_k(\mathcal{A})]_{\sim_{\tilde{\Sigma}}}$. 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 $\tilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k -phase representation.

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

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

Proof

- Set $\tilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \dots, k\}, \Sigma_p^{\text{ret}} \times \{1, \dots, k\}, \Sigma_p^{\text{int}} \times \{1, \dots, k\}))_{p \in \mathcal{P}}$.
- Build MVPA \mathcal{B} over $\tilde{\Omega}$ for words $(a_1, ph_1) \dots (a_n, ph_n)$ such that $a_1 \dots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \dots a_i \text{ is } j\text{-phase word}\}$.
- 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 $\tilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k -phase representation.

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

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

Proof

- Set $\tilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \dots, k\}, \Sigma_p^{\text{ret}} \times \{1, \dots, k\}, \Sigma_p^{\text{int}} \times \{1, \dots, k\}))_{p \in \mathcal{P}}$.
- Build MVPA \mathcal{B} over $\tilde{\Omega}$ for words $(a_1, ph_1) \dots (a_n, ph_n)$ such that $a_1 \dots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \dots a_i \text{ is } j\text{-phase word}\}$.
- 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 $\tilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k -phase representation.

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

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

Proof

- Set $\tilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \dots, k\}, \Sigma_p^{\text{ret}} \times \{1, \dots, k\}, \Sigma_p^{\text{int}} \times \{1, \dots, k\}))_{p \in \mathcal{P}}$.
- Build MVPA \mathcal{B} over $\tilde{\Omega}$ for words $(a_1, ph_1) \dots (a_n, ph_n)$ such that $a_1 \dots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \dots a_i \text{ is } j\text{-phase word}\}$.
- 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 $\tilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k -phase representation.

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

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

Proof

- Set $\tilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \dots, k\}, \Sigma_p^{\text{ret}} \times \{1, \dots, k\}, \Sigma_p^{\text{int}} \times \{1, \dots, k\}))_{p \in \mathcal{P}}$.
- Build MVPA \mathcal{B} over $\tilde{\Omega}$ for words $(a_1, ph_1) \dots (a_n, ph_n)$ such that $a_1 \dots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \dots a_i \text{ is } j\text{-phase word}\}$.
- 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 $\tilde{\Sigma}$ such that $L_k(\mathcal{A})$ is a k -phase representation.

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

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

Proof

- Set $\tilde{\Omega} = ((\Sigma_p^{\text{call}} \times \{1, \dots, k\}, \Sigma_p^{\text{ret}} \times \{1, \dots, k\}, \Sigma_p^{\text{int}} \times \{1, \dots, k\}))_{p \in \mathcal{P}}$.
- Build MVPA \mathcal{B} over $\tilde{\Omega}$ for words $(a_1, ph_1) \dots (a_n, ph_n)$ such that $a_1 \dots a_n \in L_k(\mathcal{A})$ and $ph_i = \min\{j \leq k \mid a_1 \dots a_i \text{ is } j\text{-phase word}\}$.
- 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 $\tilde{\Sigma}$, MVPA \mathcal{A} over $\tilde{\Sigma}$, and $k \in \mathbb{N}$

QUESTION 1: *Is $L_k(\mathcal{A}) \sim_{\tilde{\Sigma}}$ -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 $\tilde{\Sigma}$, MVPA \mathcal{A} over $\tilde{\Sigma}$, and $k \in \mathbb{N}$

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

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

Proof (Question 1)

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

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

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

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

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

Proof (Question 1)

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

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

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

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

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

Proof (Question 1)

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

Decidability of sufficient criterion

Theorem

The following problems are decidable in elementary time:

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

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

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

Proof (Question 1)

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

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
...

Synthesis

Implementation

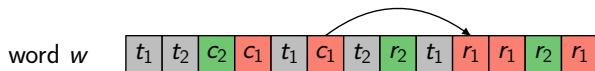
asynchronous automata
message-passing automata
CVPA
...

From words to nested Mazurkiewicz traces

word w

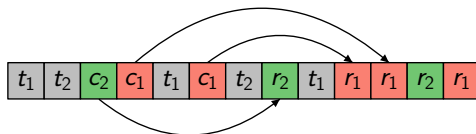
t_1	t_2	c_2	c_1	t_1	c_1	t_2	r_2	t_1	r_1	r_1	r_2	r_1
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

From words to nested Mazurkiewicz traces



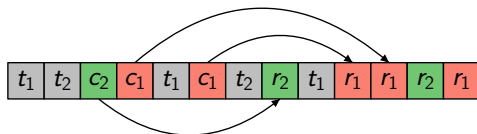
From words to nested Mazurkiewicz traces

nested word w

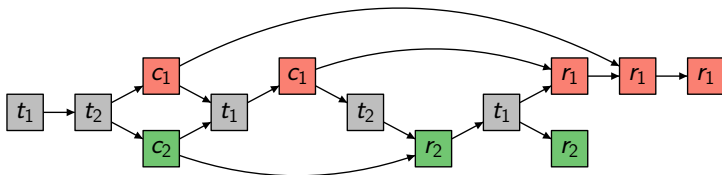


From words to nested Mazurkiewicz traces

nested word w

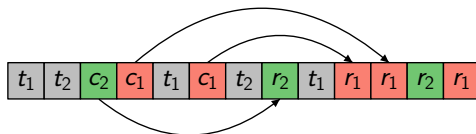


nested trace
 $T(w)$

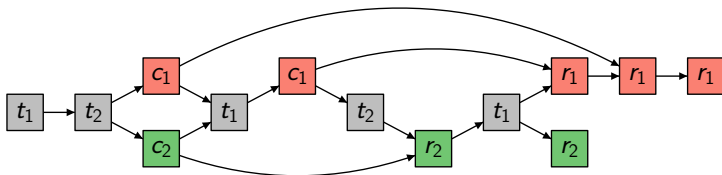


From words to nested Mazurkiewicz traces

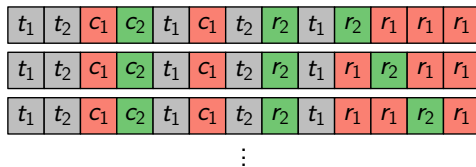
nested word w



nested trace
 $T(w)$

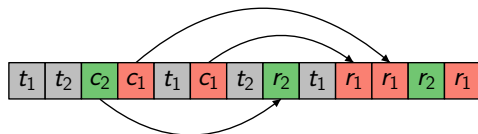


linearizations

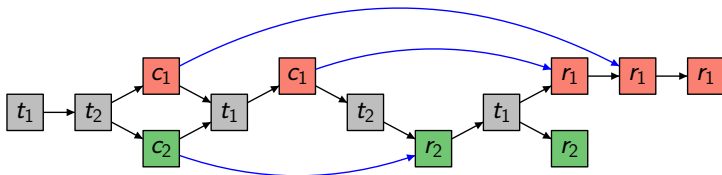


From words to nested Mazurkiewicz traces

nested word w



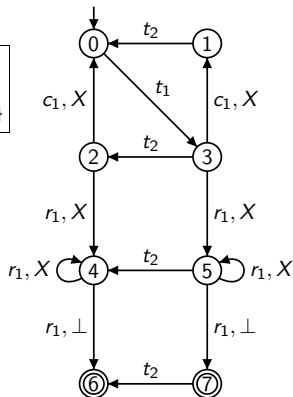
nested trace
 $T(w)$



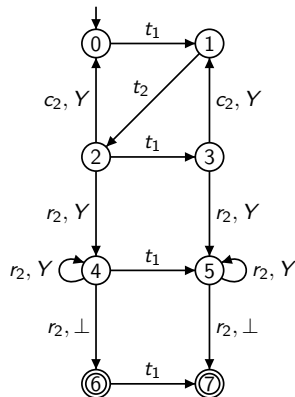
$$T(w) = (E, \leq, \mu, \lambda) \text{ where } \leq, \mu \subseteq E \times E \text{ and } \lambda : E \rightarrow \Sigma$$

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

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



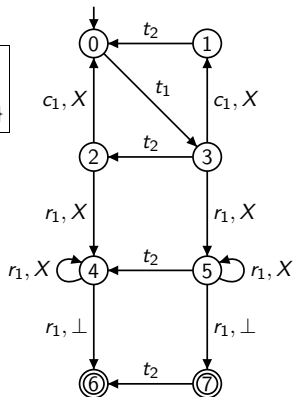
$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



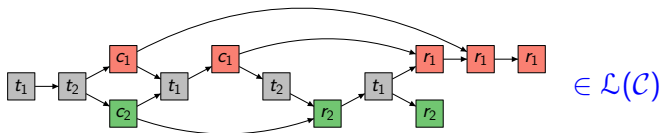
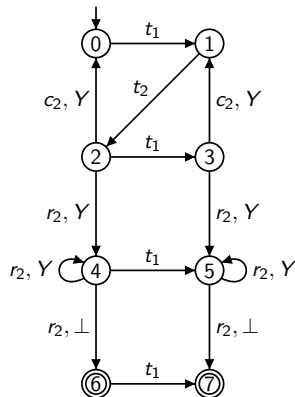
$$\mathcal{L}(\mathcal{C}) = \{T(w) \mid w \in L(\mathcal{C})\}$$

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

$$\begin{aligned} \Sigma_p^{\text{call}} &= \{c_1\} \\ \Sigma_p^{\text{ret}} &= \{r_1\} \\ \Sigma_p^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



$$\begin{aligned} \Sigma_q^{\text{call}} &= \{c_2\} \\ \Sigma_q^{\text{ret}} &= \{r_2\} \\ \Sigma_q^{\text{int}} &= \{t_1, t_2\} \end{aligned}$$



MSO logic for nested traces

Definition

Monadic second-order logic $\text{MSO}(\tilde{\Sigma})$:

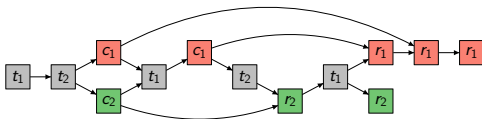
$$\begin{aligned} \varphi ::= & x \leq y \mid (x, y) \in \mu \mid \lambda(x) = a \mid \\ & x \in X \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x \varphi \mid \exists X \varphi \end{aligned}$$

MSO logic for nested traces

Definition

Monadic second-order logic $\text{MSO}(\tilde{\Sigma})$:

$$\begin{aligned} \varphi ::= & x \leq y \mid (x, y) \in \mu \mid \lambda(x) = a \mid \\ & x \in X \mid \neg \varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x \varphi \mid \exists X \varphi \end{aligned}$$



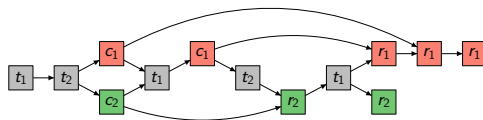
$$\models \forall x \forall y [(\lambda(x) \in \{c_1, c_2\} \wedge \lambda(y) \in \{r_1, r_2\}) \rightarrow x \leq y]$$

MSO logic for nested traces

Definition

Monadic second-order logic $\text{MSO}(\tilde{\Sigma})$:

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



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

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

Thomas' Theorem

Let $\tilde{\Sigma}$ 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 $\tilde{\Sigma}$ is an asynchronous automaton over $\tilde{\Sigma}$.

Theorem ([Tho'90])

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

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

↪ 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 $\tilde{\Sigma}$) is a nested trace T such that there is a k -phase word $w \in \Sigma^*$ with $T(w) = T$.

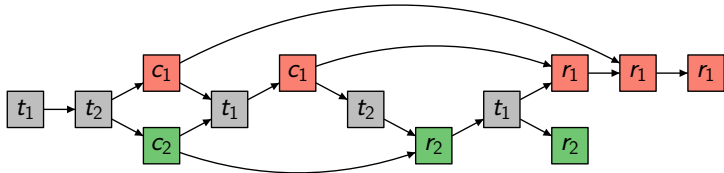
Let $\text{Tr}_k(\tilde{\Sigma})$ denote the set of k -phase traces.

k -phase traces

Definition

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

Let $\text{Tr}_k(\tilde{\Sigma})$ 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 $\tilde{\Sigma}$, there is $\varphi \in \text{MSO}(\tilde{\Sigma})$ such that $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C})$.

Theorem

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

$$\mathcal{L}(\mathcal{C}) = \mathcal{L}(\varphi) \cap \text{Tr}_k(\tilde{\Sigma})$$

Proof

- By induction.
- Lemma: If $\mathcal{L} \subseteq \text{Tr}_k(\tilde{\Sigma})$ is recognizable, then so is $\bar{\mathcal{L}} \cap \text{Tr}_k(\tilde{\Sigma})$.
- $\text{Tr}_k(\tilde{\Sigma})$ 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 $\tilde{\Sigma}$, there is $\varphi \in \text{MSO}(\tilde{\Sigma})$ such that $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C})$.

Theorem

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

$$\mathcal{L}(\mathcal{C}) = \mathcal{L}(\varphi) \cap \text{Tr}_k(\tilde{\Sigma})$$

Proof

- By induction.
- Lemma: If $\mathcal{L} \subseteq \text{Tr}_k(\tilde{\Sigma})$ is recognizable, then so is $\bar{\mathcal{L}} \cap \text{Tr}_k(\tilde{\Sigma})$.
- $\text{Tr}_k(\tilde{\Sigma})$ 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 $\tilde{\Sigma}$, there is $\varphi \in \text{MSO}(\tilde{\Sigma})$ such that $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C})$.

Theorem

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

$$\mathcal{L}(\mathcal{C}) = \mathcal{L}(\varphi) \cap \text{Tr}_k(\tilde{\Sigma})$$

Proof

- By induction.
- Lemma: If $\mathcal{L} \subseteq \text{Tr}_k(\tilde{\Sigma})$ is recognizable, then so is $\bar{\mathcal{L}} \cap \text{Tr}_k(\tilde{\Sigma})$.
- $\text{Tr}_k(\tilde{\Sigma})$ 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

...

Synthesis

Implementation

asynchronous automata

message-passing automata

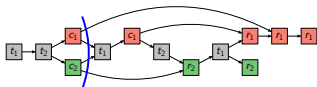
CVPA

...

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!