On Timed Scope-bounded Context-sensitive Languages

Ramchandra Phawade

IIT Dharwad, India

Joint work with Krishna, Trivedi, and Bhave
DLT 2019, Warsaw Poland

Formal methods of verification

Given System M and Property P, if System M satisfies property P?
Meaning:

$$\text{All behaviours of } M \text{ should satisfy } P?$$

How?
Model M by A_M and P by A_P.

$$\text{Check if } Lang(A_M) \subseteq Lang(A_P) \text{ ?}$$

How it is done?
$Lang(A_M) \cap Lang(A_P^c) = \phi$?

Formal methods of verification

Given System M and Property P, if System M satisfies property P?
Meaning:

$$\text{All behaviours of } M \text{ should satisfy } P?$$

How?
Model M by A_M and P by A_P.

$$\text{Check if } Lang(A_M) \subseteq Lang(A_P) \text{ ?}$$

How it is done?
$Lang(A_M) \cap Lang(A_P^c) = \phi$?
Boolean closure: intersection, complement and union operations.
In addition decidability of emptiness checking (reachability) in the models.

Modeling systems (programs) with Automata

Control flow $\rightarrow$ (Finite) state machine.

Modeling systems (programs) with Automata

Control flow $\rightarrow$ (Finite) state machine.
Control flow $+$ call /return $\rightarrow$ Control $+$ stack
(Programs with recursion) $\rightarrow$ Push down automata (PDA).

Modeling systems (programs) with Automata

Control flow $\rightarrow$ (Finite) state machine.
Control flow $+$ call /return $\rightarrow$ Control $+$ stack
(Programs with recursion) $\rightarrow$ Push down automata (PDA).
Concurrent Programs with recursion $\rightarrow$ Control$+$ Multiple stacks
Communication: shared-memory $\rightarrow$ PDA with Multiple stacks.

Modeling systems (programs) with Automata

Control flow $\rightarrow$ (Finite) state machine.
Control flow + call /return $\rightarrow$ Control + stack
(Programs with recursion) $\rightarrow$ Push down automata (PDA).
Concurrent Programs with recursion $\rightarrow$ Control+ Multiple stacks
Communication: shared-memory $\rightarrow$ PDA with Multiple stacks.
Timing constraints in the control flow $\rightarrow$ Timed Control + Multi-stack

Modeling systems (programs) with Automata

Control flow $\rightarrow$ (Finite) state machine.
Control flow + call /return $\rightarrow$ Control + stack
(Programs with recursion) $\rightarrow$ Push down automata (PDA).
Concurrent Programs with recursion $\rightarrow$ Control+ Multiple stacks
Communication: shared-memory $\rightarrow$ PDA with Multiple stacks.
Timing constraints in the control flow $\rightarrow$ Timed Control + Multi-stack
Timing constraints + tracking time of calls /returns $\rightarrow$ Timed Control + Timed Multi-stack

Modeling systems (programs) with Automata

Control flow $\rightarrow$ (Finite) state machine.
Control flow + call /return $\rightarrow$ Control + stack
(Programs with recursion) $\rightarrow$ Push down automata (PDA).
Concurrent Programs with recursion $\rightarrow$ Control+ Multiple stacks
Communication: shared-memory $\rightarrow$ PDA with Multiple stacks.
Timing constraints in the control flow $\rightarrow$ Timed Control + Multi-stack
Timing constraints + tracking time of calls /returns $\rightarrow$ Timed Control + Timed Multi-stack
Reachability undecidable, No Boolean closure.

PDA

Context free languages:
Emptiness check decidable.
Not closed under intersection, complementation and determinization.

VPA

- Make alphabet visible –partition it into call, return and internal.
- Example : $\{a^n b^n : n \geq 0\}$
  a – call/push letter
  b– return/pop letter.
- VPA–Boolean closure (Alur and Madhusudan, STOC 2004).

2-stack VPA

Visibly pushdown alphabet for each stack.

2-stack VPA

Visibly pushdown alphabet for each stack.
$\Sigma_1 = \{a, \hat{a}, c, \hat{c}\}$, $\Gamma_1 = \{\alpha, \$\}$, and $\Sigma_2 = \{b, d\}$, $\Gamma_2 = \{\$\}$.

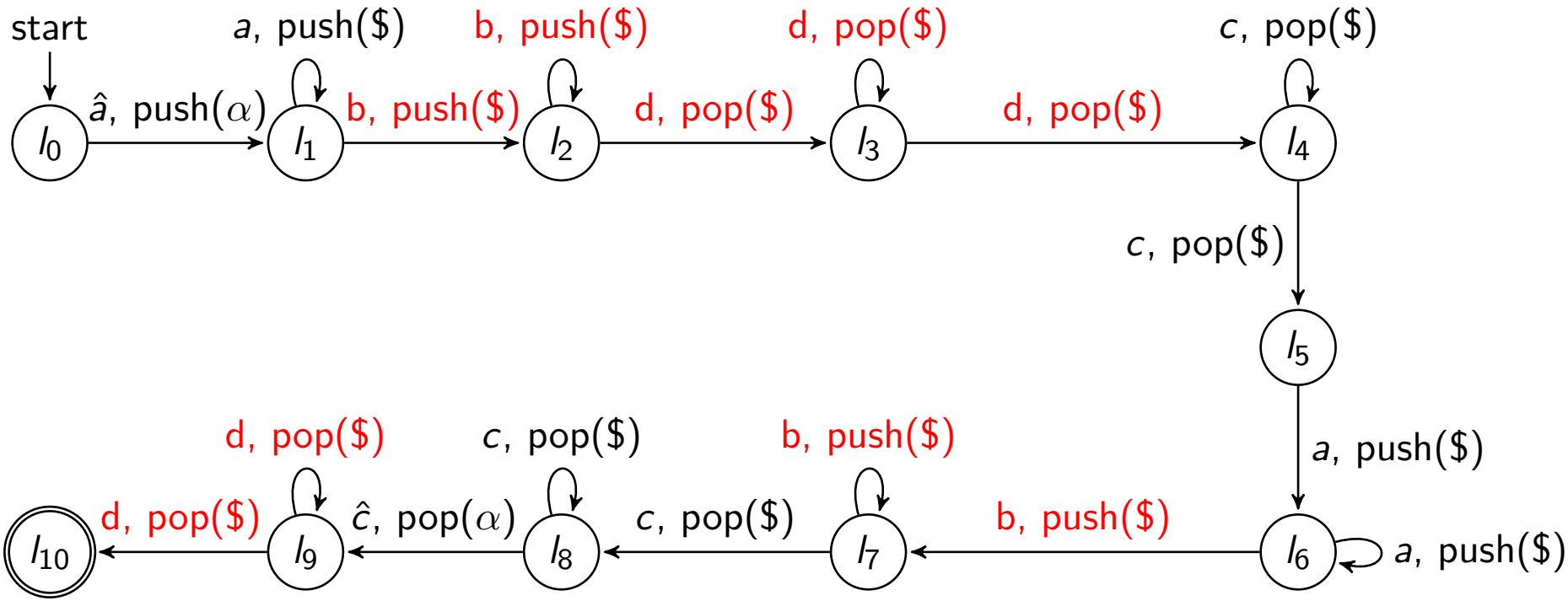


Figure: $\{\hat{a}a^x\ b^y d^z\ c^l a^m\ b^{y'}\ c^{x+m-l}\hat{c}\ d^{z'} \mid x, l, \geq 1,\ y, z \geq 2,\ z', y' \geq 1\}$

Multiple stacks VPAs : k-round

Multiple stacks– Put restrictions on access of stacks to get decidability of emptiness (reachability) check.

Multiple stacks VPAs : k-round

Multiple stacks– Put restrictions on access of stacks to get decidability of emptiness (reachability) check.

- k-round MVPA: Word divided into at most k-rounds.
  In each round each stack can be used at most once.

Multiple stacks VPAs : k-round

Multiple stacks– Put restrictions on access of stacks to get decidability of emptiness (reachability) check.

- k-round MVPA: Word divided into at most k-rounds.
  In each round each stack can be used at most once.
- $\{\hat{a}a^x\ b^y d^z \cdot c^l a^m\ b^{y'} \cdot\ c^{x+m-l}\hat{c}\ d^{z'} \mid x, l, \geq 1,\ y, z \geq 2,\ z', y' \geq 1\}$

Multiple stacks VPAs : k-round

Multiple stacks– Put restrictions on access of stacks to get decidability of emptiness (reachability) check.

- k-round MVPA: Word divided into at most k-rounds.
  In each round each stack can be used at most once.
- $\{\hat{a}a^x\ b^y d^z \cdot c^l a^m\ b^{y'} \cdot\ c^{x+m-l}\hat{c}\ d^{z'} \mid x, l, \geq 1,\ y, z \geq 2,\ z', y' \geq 1\}$
- 3-round MVPL.

Multiple stacks VPAs : k-round

Multiple stacks– Put restrictions on access of stacks to get decidability of emptiness (reachability) check.

- k-round MVPA: Word divided into at most k-rounds.
  In each round each stack can be used at most once.
- $\{\hat{a}a^x\ b^y d^z \cdot c^l a^m\ b^{y'} \cdot\ c^{x+m-l}\hat{c}\ d^{z'} \mid x, l, \geq 1,\ y, z \geq 2,\ z', y' \geq 1\}$
- 3-round MVPL.
- Closed under Boolean operations
  (Madhusudan et. al LATIN 2010).

Multiple stacks VPAs : k-scope

- k-scope MVPA: A popped symbol from some stack, it must have been pushed within k-contexts of same stack earlier.

Multiple stacks VPAs : k-scope

- k-scope MVPA: A popped symbol from some stack, it must have been pushed within k-contexts of same stack earlier.
- $\{\hat{a}a^x \ b^y d^z \ c^l a^m \ b^{y'} \ c^{x+m-l}\hat{c} \ d^{z'} \mid x, l, \geq 1, \ y, z \geq 2, \ z', y' \geq 1\}$

Multiple stacks VPAs : k-scope

- k-scope MVPA: A popped symbol from some stack, it must have been pushed within k-contexts of same stack earlier.
- $\{\hat{a}a^x \ b^y d^z \ c^l a^m \ b^{y'} \ c^{x+m-l}\hat{c} \ d^{z'} \mid x, l, \geq 1, \ y, z \geq 2, \ z', y' \geq 1\}$
- 3-scope MVPL.
- Closed under Boolean operations (La Torre s., et al. DLT 2014).

Time in control : Timed Automata

- Set of finite clocks with automata
- Contraints on transitions

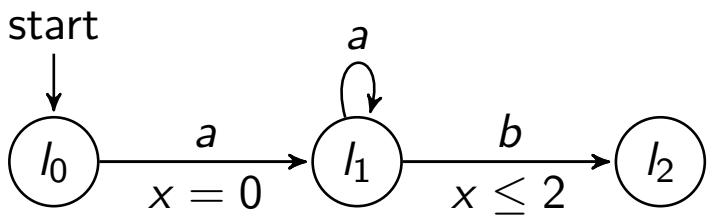


Figure: Timed Automata

- A Timed word : $(a, 0)(a, 0.5)(a, 1.99) \ldots (b, 2)$
- Timed automata – not closed under Boolean operations (Alur and Dill, TCS 1994).

Time in control : Event Clocks

- Event clocks–two for each symbol–one records time elapsed for each symbol, and another time predicted for next occurrence of it.
- Event Clock Automata closed under Boolean operations (Alur, Fix and Henzinger, TCS 1999).

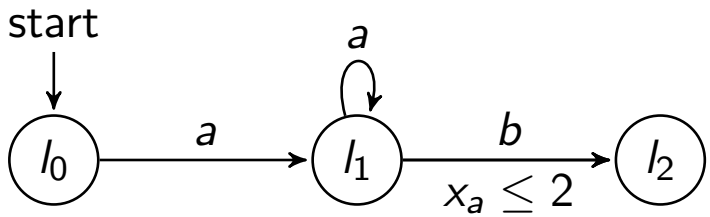


Figure: Event Clock Automata

An accepted timed word: $(a, 0)(a, 0.5) \ldots (a, 200)(b, 202)$

Time in Stack

- Stack elements have timestamps (age)
- Initialized to zero when pushed
- Age checked when popped from stack e.g., $age(\$) \in (3, 5]$?
- Ages grow uniformly across all stacks
- Timed automata (control) + Timed stack = Timed automata+ untimed stack – reachability decidable (Abdulla et. al, LICS 2012)
- No Boolean closure : PDA + TA

Event clock - VPA

- ECVPA with Untimed stack closed under Boolean operations. (Tang and Ogawa, SOFSEM 2009).

Event clock - VPA

- ECVPA with Untimed stack closed under Boolean operations. (Tang and Ogawa, SOFSEM 2009).
- ECVPA with timed stack is closed under Boolean operations. (with Krishna, Ashutosh, Vrunda, Devendra, **LATA 2016**)

Event clock - VPA

- ECVPA with Untimed stack closed under Boolean operations. (Tang and Ogawa, SOFSEM 2009).
- ECVPA with timed stack is closed under Boolean operations. (with Krishna, Ashutosh, Vrunda, Devendra, **LATA 2016**)
- k-round Multistack ECVPA with dense time stack: Boolean Closure (with Krishna, Ashutosh, Devendra, **DLT 2016**)

K-scope dense time multi stack VPA : An example

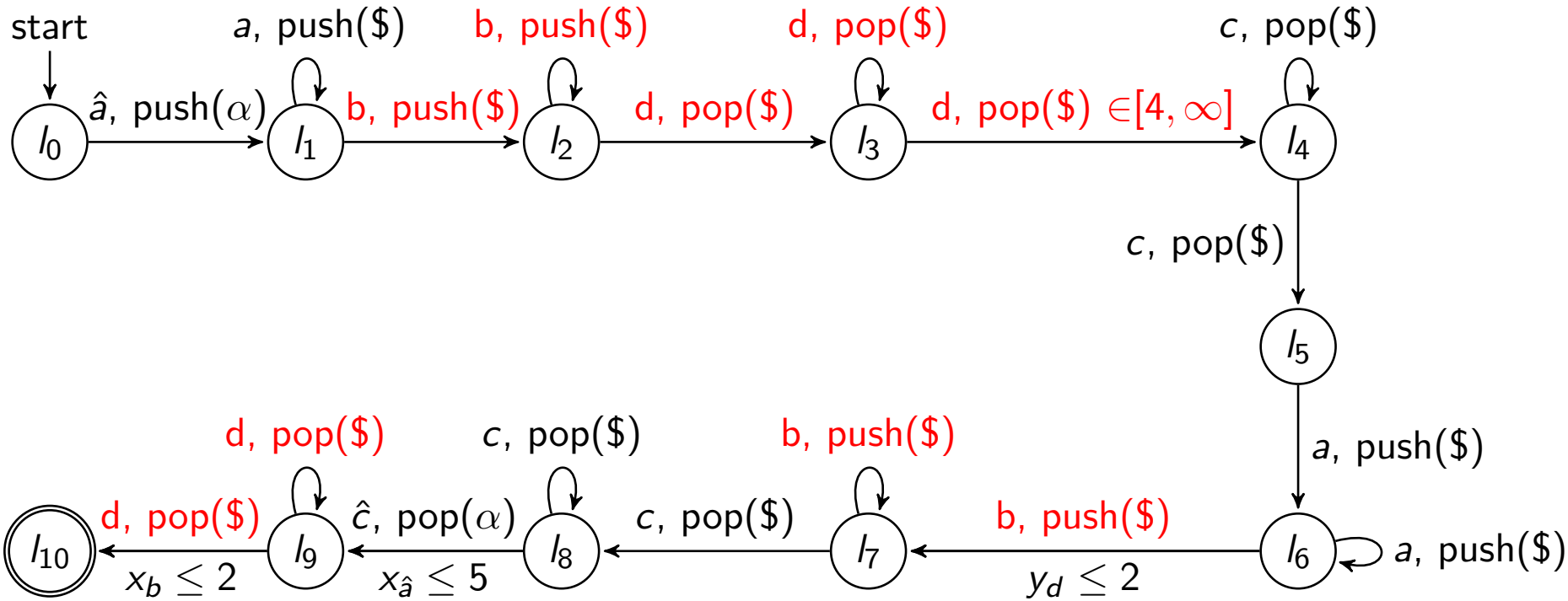


Figure: $\{\hat{a}a^x b^y d^z c^l a^m b^{y'} c^{x+m-l} \hat{c} d^{z'} \mid x, l, \geq 1,\ y, z \geq 2,\ z', y' \geq 1\}$

Result : K-scope dense time multi stack VPA

- Reachability decidable
- Boolean Closure
- Logical Characterization

Reachability of k-scope dtECMVPA : proof idea

1. Untime stacks to get k-scope ECMVPA
   (for single stack —- Parosh et al. 2012)

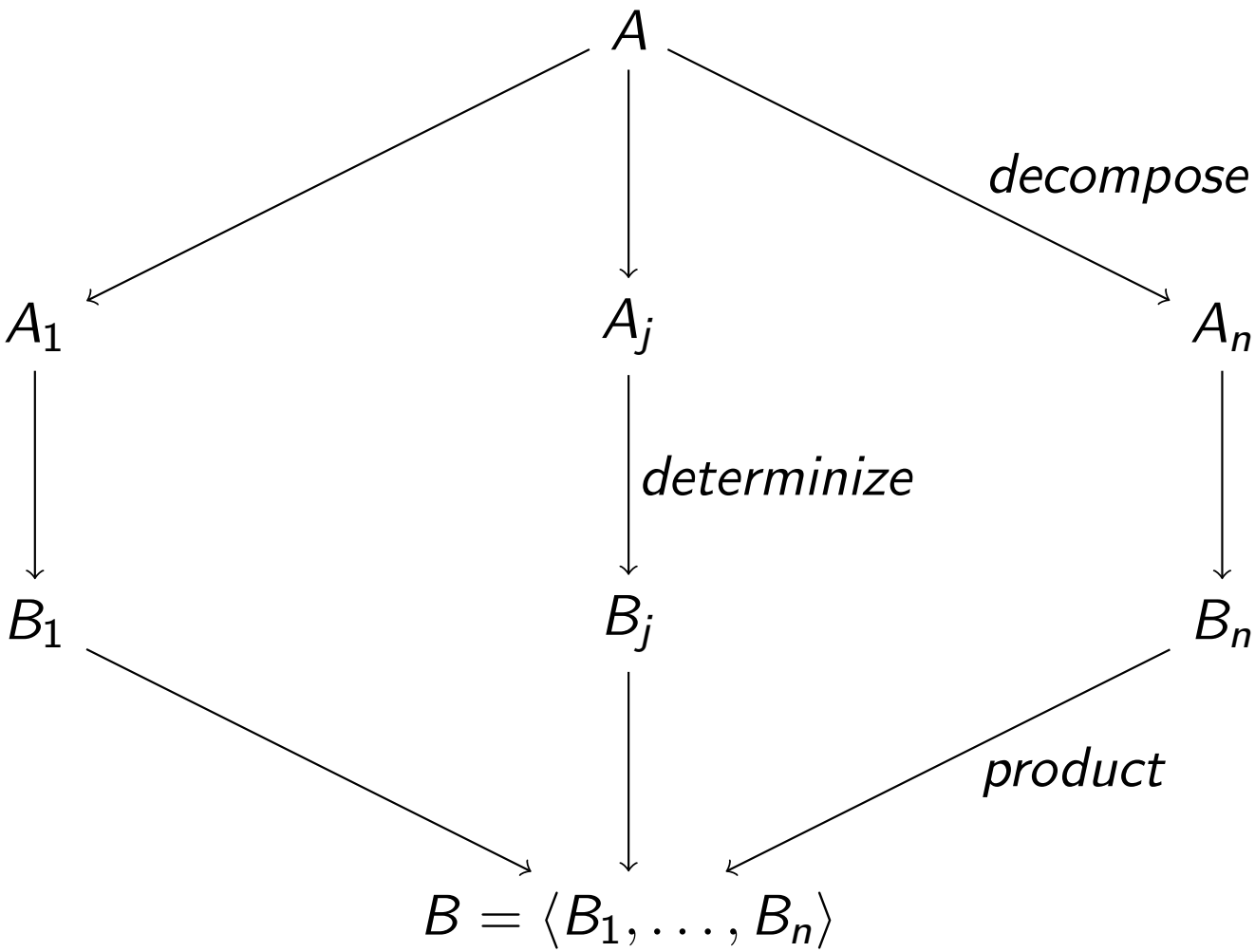
Reachability of k-scope dtECMVPA : proof idea

1. Untime stacks to get k-scope ECMVPA
   (for single stack —- Parosh et al. 2012)
2. Untime control to get k-scope MVPA
   (untimed k-scope MVPA — La Torre et al. 2014)

Determinization of dense time k-scope ECMVPA : Proof Idea

1. A : k-scope dense time ECMVPA
2. Untime stacks to get M' — k-scope ECMVPA
3. determinize M' to get Det(M') such that L(A)= h(L(Det(M')).
4. Construct B= Det(M) by erasing extra transitions in Det(M').

Determinization of k-scope ECMVPA : Proof Idea 1



Construction of A_j to mimic A on projection of w

- $w^j = u_1 u_2 \ldots u_m$
- behaviour of A on w^j : $l_1 \xrightarrow{u_1} l'_1 \ldots l_2 \xrightarrow{u_2} l'_2 \ldots l_m \xrightarrow{u_m} l'm$.

Construction of A_j to mimic A on projection of w

- $w^j = u_1 u_2 \ldots u_m$
- behaviour of A on w^j : $l_1 \xrightarrow{u_1} l'_1 \ldots l_2 \xrightarrow{u_2} l'_2 \ldots l_m \xrightarrow{u_m} l'm$.
- For each context guess the entering state, leaving state, and time interval of clock evaluations
- $\mathcal{I} = \{[0,0], (0,1), [1,1], (1,2), \ldots, [cmax, cmax], (cmax, \infty)\}$.

Construction of A_j to mimic A on projection of w

- $w^j = u_1 u_2 \dots u_m$
- behaviour of A on w^j : $l_1 \xrightarrow{u_1} l'_1 \dots l_2 \xrightarrow{u_2} l'_2 \dots l_m \xrightarrow{u_m} l'm$.
- For each context guess the entering state, leaving state, and time interval of clock evaluations
- $\mathcal{I} = \{[0,0],(0,1),[1,1],(1,2),\dots,[cmax,cmax],(cmax,\infty)\}$.
- $v^j = u_1 \cdot \hat{\#} \cdot u_2 \hat{\#} \dots \hat{\#} u_m \hat{\#}$,
  where $\hat{\#}$ marks end of context or empty stack.
- A_j equipped with switching vector $v_j \in (L \times \mathcal{I} \times L)^m$.
- New clocks, but when A_j moves we make sure that reading $\#$ consumes no time.

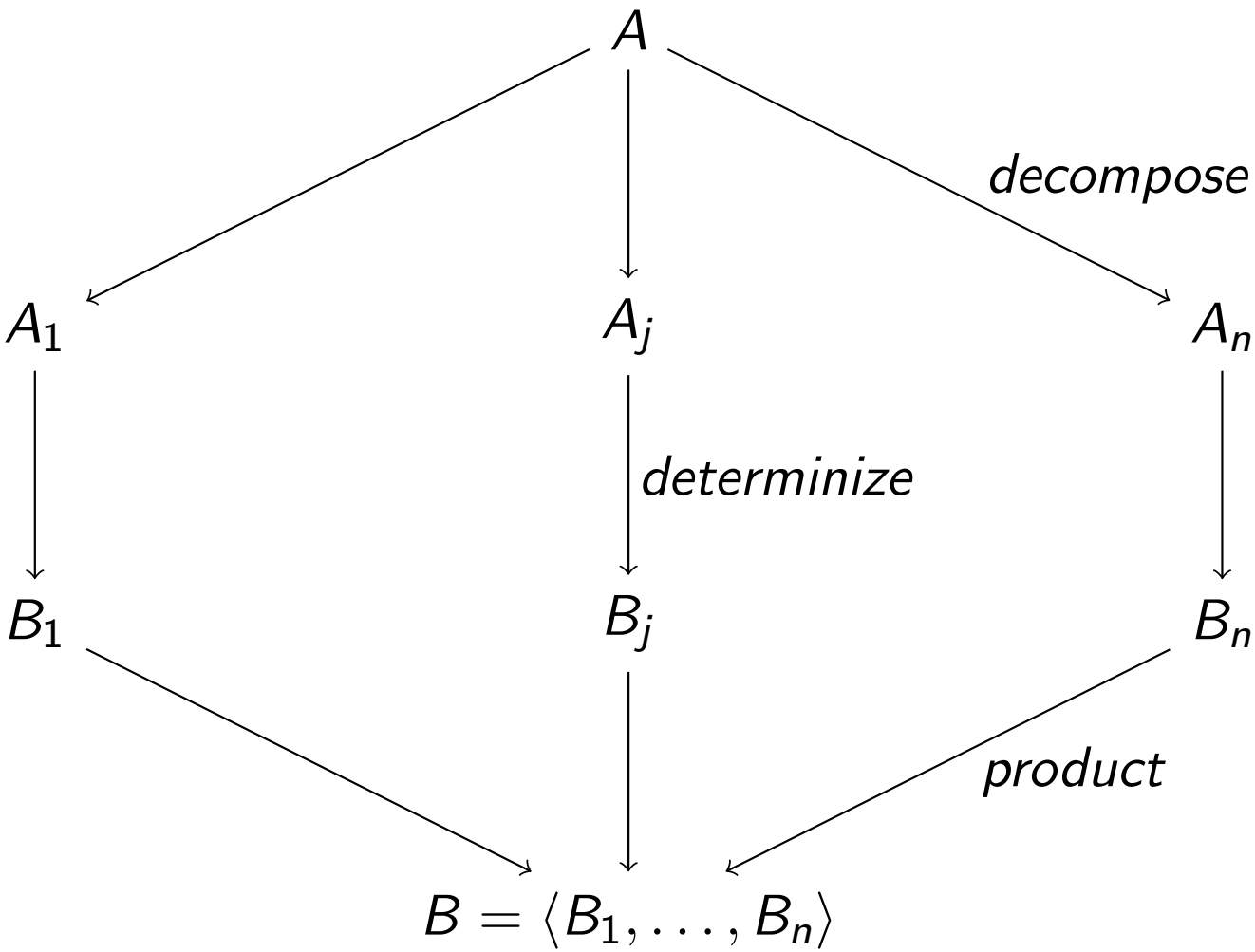
Construction of A_j : contd..

- Switching vector $v_j \in (L \times \mathcal{I} \times L)^m$
- Switching vector too big, arbitrarily long. m could be greater than k.

Construction of A_j : contd..

- Switching vector $v_j \in (L \times \mathcal{I} \times L)^m$
- Switching vector too big, arbitrarily long. m could be greater than k.
- solution : We guess SVs of length k repeatedly.
- why it works?
- It works because words are k-scoped. (Switching Lemma).

Determinization of k-scope ECMVPA : where were we?



Proof contd: simulation of k-scope MVPA A by product $\langle A_1, \ldots, A_n \rangle$

- Now build a product machine $\langle A_1, \ldots, A_j, \ldots, A_n \rangle$
- prove that product machine can simulate A.
- locations : $(p_1, \ldots, p_j, \ldots, p_n, , j)$
  j-whose context is it?
- control is passed from one to another, depending on the current context.

Proof contd: simulation of k-scope MVPA A by product $\langle A_1, \ldots, A_n \rangle$

- Now build a product machine $\langle A_1, \ldots, A_j, \ldots, A_n \rangle$
- prove that product machine can simulate A.
- locations : $(p_1, \ldots, p_j, \ldots, p_n, , j)$
  j-whose context is it?
- control is passed from one to another, depending on the current context.
- Possible only if guessed individual switching vectors are globally correct. That is this "handing over" control is done correctly.

Stitching Lemma

Thank you.

Future Problems

k-phase timed context sensitive languages?

Logical Characterization

$$\varphi := Q_a(x) \mid x \in X \mid \mu_j(x, y) \mid \lhd_a(x) \in I \mid \rhd_a(x) \in I \mid \theta_j(x) \in I \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X$$

where $a \in \Sigma$, $x_a \in C_\Sigma$, x is a first order variable and X is a second order variable.

- For an interval I, the predicate $\lhd_a(i) \in I$ evaluates to true on the word structure iff $\nu_i^w(x_a) \in I$ for recorder clock x_a.
- predicate $\theta_j(i) \in I$ which evaluates to true on the word structure iff $w[i] = (a, t_i)$ with $a \in \Sigma_r^j$ and $w[i] \in \Sigma_r^j$, and there is some $k < i$ such that $\mu_j(k, i)$ evaluates to true and $t_i - t_k \in I$.
  The predicate $\theta_j(i)$ measures time elapsed between position k where a call was made on the stack j, and position i, its matching return.