Some open problems in deciding bisimulation equivalence

Petr Jančar

Dept of Computer Science
Technical University Ostrava (FEI VŠB-TUO), Czech Republic
www.cs.vsb.cz/jancar

Open Problems in Concurrency Theory
Bertinoro, Italy, 18–21 June, 2014

Outline

- bisimulation equivalence on labelled transition systems (LTSs)

Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)

Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)
- a line of research started by Baeten, Bergstra, Klop (JACM 1993): bisimilarity decidable for normed BPA

Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)
- a line of research started by Baeten, Bergstra, Klop (JACM 1993): bisimilarity decidable for normed BPA
  - the current best time-complexity bound $O(n^4 polylog(n))$ (PhD thesis W. Czerwinski 2012).

Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)
- a line of research started by Baeten, Bergstra, Klop (JACM 1993): bisimilarity decidable for normed BPA
  - the current best time-complexity bound $O(n^4 polylog(n))$ (PhD thesis W. Czerwinski 2012).
  - for (unnormed) BPA in [ExpTime ... 2-ExpTime]

Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)
- a line of research started by Baeten, Bergstra, Klop (JACM 1993): bisimilarity decidable for normed BPA
  - the current best time-complexity bound $O(n^4 polylog(n))$ (PhD thesis W. Czerwinski 2012).
  - for (unnormed) BPA in [ExpTime ... 2-ExpTime]
- Sénizergues (SIAM J.Comput 2005): bisimilarity decidable for (an equivalent of) FO-grammars

Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)
- a line of research started by Baeten, Bergstra, Klop (JACM 1993): bisimilarity decidable for normed BPA
  - the current best time-complexity bound $O(n^4 polylog(n))$ (PhD thesis W. Czerwinski 2012).
  - for (unnormed) BPA in [ExpTime ... 2-ExpTime]
- Sénizergues (SIAM J.Comput 2005): bisimilarity decidable for (an equivalent of) FO-grammars
  - new proof J. ICALP'14 (arxiv.org/abs/1405.7923)

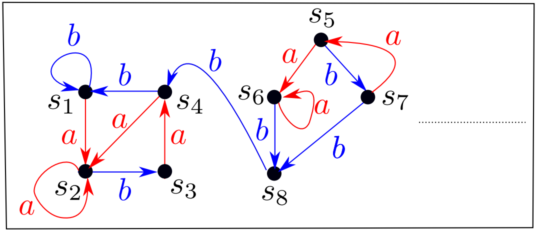
Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)
- a line of research started by Baeten, Bergstra, Klop (JACM 1993): bisimilarity decidable for normed BPA
  - the current best time-complexity bound $O(n^4 polylog(n))$ (PhD thesis W. Czerwinski 2012).
  - for (unnormed) BPA in [ExpTime ... 2-ExpTime]
- Sénizergues (SIAM J.Comput 2005): bisimilarity decidable for (an equivalent of) FO-grammars
  - new proof J. ICALP'14 (arxiv.org/abs/1405.7923)
  - Ackermann-hard (J. FoSSaCS'14); TOWER-hard when no ε-transitions (Benedikt, Göller, Kiefer, Murawski at LiCS'13).

Outline

- bisimulation equivalence on labelled transition systems (LTSs)
- here generated by sequential systems (sorry :-)):
  - context-free grammars (BPA processes)
  - pushdown automata (pushdown processes)
  - first-order grammars, or FO-grammars (also pushdown processes)
- a line of research started by Baeten, Bergstra, Klop (JACM 1993): bisimilarity decidable for normed BPA
  - the current best time-complexity bound $O(n^4 polylog(n))$ (PhD thesis W. Czerwinski 2012).
  - for (unnormed) BPA in [ExpTime ... 2-ExpTime]
- Sénizergues (SIAM J.Comput 2005): bisimilarity decidable for (an equivalent of) FO-grammars
  - new proof J. ICALP'14 (arxiv.org/abs/1405.7923)
  - Ackermann-hard (J. FoSSaCS'14); TOWER-hard when no ε-transitions (Benedikt, Göller, Kiefer, Murawski at LiCS'13).
  - branching bisimilarity (van Glabbeek, Weijland, JACM 1996); recent interesting twists by Y. Fu (ICALP'13) and others: BPA, PDA

Labelled transition systems; bisimulation equivalence

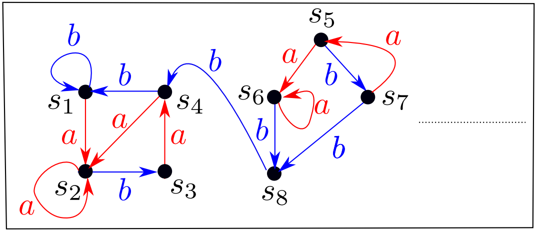


$$\mathcal{L} = (\mathcal{S}, \mathcal{A}, (\xrightarrow{a})_{a \in \mathcal{A}})$$

$$\mathcal{S} = \{s_1, s_2, s_3, \ldots\}$$

$$\mathcal{A} = \{a, b\} \qquad \xrightarrow{a} \subseteq \mathcal{S} \times \mathcal{S} \qquad \xrightarrow{b} \subseteq \mathcal{S} \times \mathcal{S}$$

Labelled transition systems; bisimulation equivalence



$s_1 \xrightarrow{ab} s_3 \xrightarrow{a}$

$s_5 \xrightarrow{ab} s_8 \stackrel{a}{\not\rightarrow}$

$s_1 \underset{\sim_2}{\not\sim_3} s_5$

$EL(s_1, s_5) = 2$

$s \sim_0 t$ (for all s, t)

$s \sim_{k+1} t$:

$\forall (s \xrightarrow{a} s') \; \exists (t \xrightarrow{a} t') : s' \sim_k t'$

$\forall (t \xrightarrow{a} t') \; \exists (s \xrightarrow{a} s') : s' \sim_k t'$

$s \sim_\omega t$ ... $\forall k \in \mathbf{N} : s \sim_k t$

$EL(s,t) = \max\{k \mid s \sim_k t\}$

Bisimulation equivalence as a game

Assume LTS $\mathcal{L} = (\mathcal{S}, \mathcal{A}, (\xrightarrow{a})_{a \in \mathcal{A}})$.

In a round starting with a position (s, t),

1. Attacker chooses either some $s \xrightarrow{a} s'$ or some $t \xrightarrow{a} t'$.
2. Defender responses by some $t \xrightarrow{a} t'$ or some $s \xrightarrow{a} s'$, respectively.

The new position is (s', t').

The rounds are repeated. If a player is stuck, then (s)he loses.
An infinite play is a win of Defender.

Bisimulation equivalence as a game

Assume LTS $\mathcal{L} = (\mathcal{S}, \mathcal{A}, (\xrightarrow{a})_{a \in \mathcal{A}})$.

In a round starting with a position (s, t),

1. Attacker chooses either some $s \xrightarrow{a} s'$ or some $t \xrightarrow{a} t'$.
2. Defender responses by some $t \xrightarrow{a} t'$ or some $s \xrightarrow{a} s'$, respectively.

The new position is (s', t').

The rounds are repeated. If a player is stuck, then (s)he loses.
An infinite play is a win of Defender.

We have $s \sim t$ iff Defender has a winning strategy from position (s, t),
and $s \sim_k t$ iff Defender can survive k rounds.

Bisimulation equivalence as a game

Assume LTS $\mathcal{L} = (\mathcal{S}, \mathcal{A}, (\xrightarrow{a})_{a \in \mathcal{A}})$.

In a round starting with a position (s, t),

1. Attacker chooses either some $s \xrightarrow{a} s'$ or some $t \xrightarrow{a} t'$.
2. Defender responses by some $t \xrightarrow{a} t'$ or some $s \xrightarrow{a} s'$, respectively.

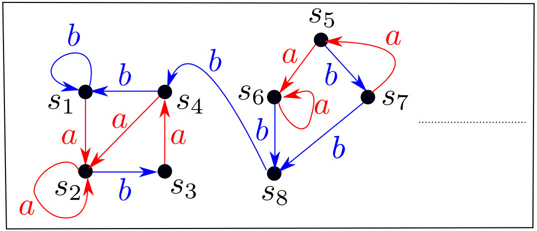
The new position is (s', t').

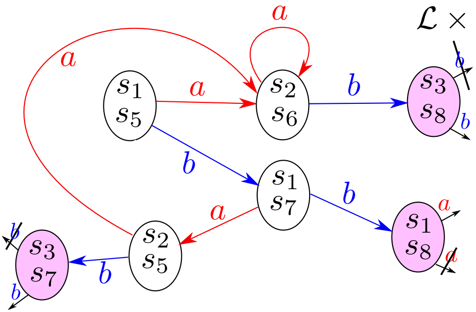
The rounds are repeated. If a player is stuck, then (s)he loses.
An infinite play is a win of Defender.

We have $s \sim t$ iff Defender has a winning strategy from position (s, t), and $s \sim_k t$ iff Defender can survive k rounds.

Observation. For deterministic LTSs, bisimulation equivalence coincides with trace equivalence.

Labelled transition systems; bisimulation equivalence





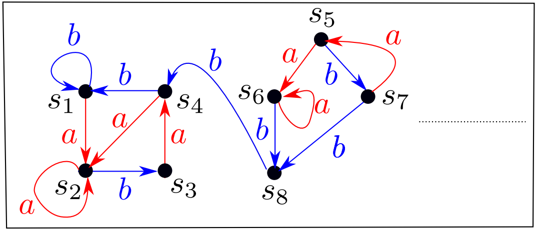
$$\mathcal{L}\times\mathcal{L} = (\mathcal{S}\times\mathcal{S}, \mathcal{A}, (\xrightarrow{a})_{a\in\mathcal{A}})$$

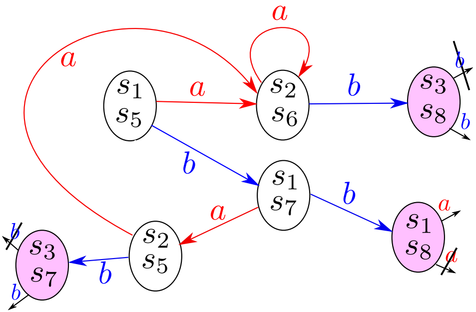
$$(r_1, r_2) \xrightarrow{a} (r_1', r_2')$$

$$\Updownarrow$$

$$r_1 \xrightarrow{a} r_1' \;\wedge\; r_2 \xrightarrow{a} r_2'$$

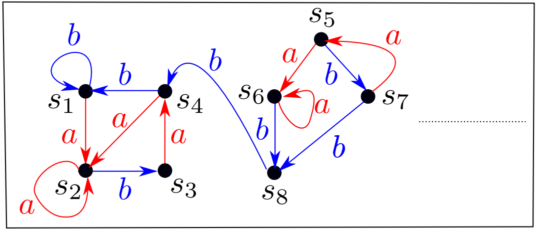
Labelled transition systems; bisimulation equivalence

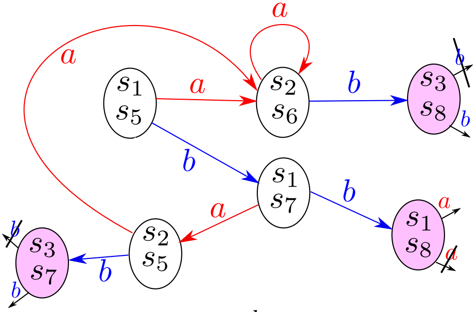




$(s_1, s_5) \xrightarrow{a} (s_2, s_6) \xrightarrow{b} (s_3, s_8)$ ... an optimal play
(EL drops by 1 in each step)

Labelled transition systems; bisimulation equivalence





Observation:

$$r \sim_{k+1} \boxed{s \begin{smallmatrix}\not\sim_{k+1}\\ \sim_k\end{smallmatrix} t}$$

$$\Downarrow$$

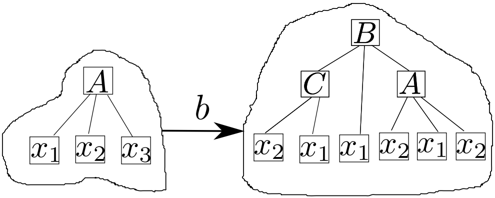
$$\boxed{r \begin{smallmatrix}\not\sim_{k+1}\\ \sim_k\end{smallmatrix} t}$$

$(s_1, s_5) \xrightarrow{a} (s_2, s_6) \xrightarrow{b} (s_3, s_8)$ ... an optimal play
(EL drops by 1 in each step)

FO-grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ ... rules $A(x_1, \ldots, x_m) \xrightarrow{a} E$

$$A(x_1, x_2, x_3) \xrightarrow{b} B(C(x_2, x_1), x_1, A(x_2, x_1, x_2))$$

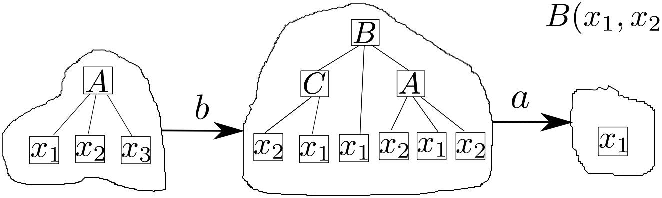
$$B(x_1, x_2, x_3) \xrightarrow{a} x_2$$



FO-grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ ... rules $A(x_1, \ldots, x_m) \xrightarrow{a} E$

$A(x_1, x_2, x_3) \xrightarrow{b} B(C(x_2, x_1), x_1, A(x_2, x_1, x_2))$

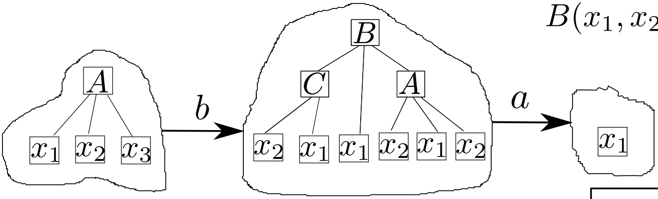
$B(x_1, x_2, x_3) \xrightarrow{a} x_2$



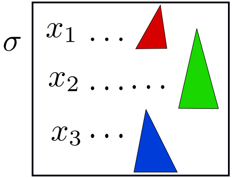
FO-grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ ... rules $A(x_1, \ldots, x_m) \xrightarrow{a} E$

$A(x_1, x_2, x_3) \xrightarrow{b} B(C(x_2, x_1), x_1, A(x_2, x_1, x_2))$

$B(x_1, x_2, x_3) \xrightarrow{a} x_2$



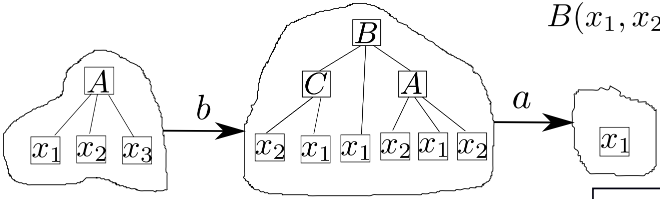
$F \xrightarrow{a} G$ implies $F\sigma \xrightarrow{a} G\sigma$



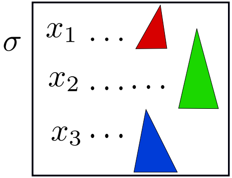
FO-grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ ... rules $A(x_1, \ldots, x_m) \xrightarrow{a} E$

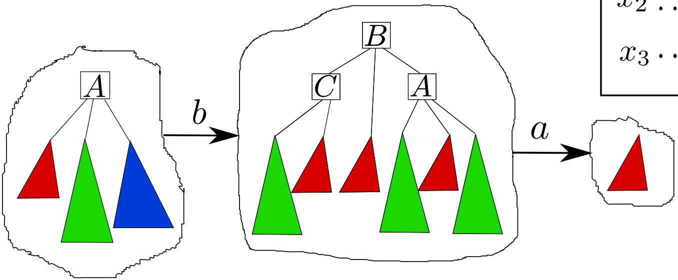
$A(x_1, x_2, x_3) \xrightarrow{b} B(C(x_2, x_1), x_1, A(x_2, x_1, x_2))$

$B(x_1, x_2, x_3) \xrightarrow{a} x_2$



$F \xrightarrow{a} G$ implies $F\sigma \xrightarrow{a} G\sigma$

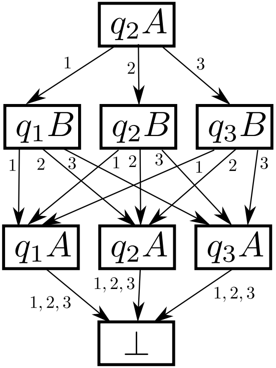


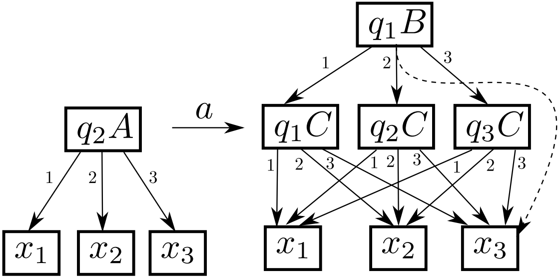


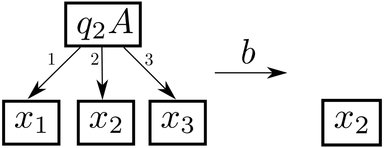
(D)pda from a first-order term perspective

$Q = \{q_1, q_2, q_3\}$
configuration q_2ABA

(pushing) rule $q_2A \xrightarrow{a} q_1BC$

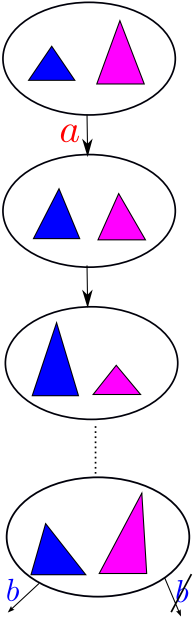






(popping) rule $q_2A \xrightarrow{b} q_2$ $\qquad$ $q_2C \xrightarrow{\varepsilon} q_3$

Bounding lengths of witnesses (where EL keeps dropping)

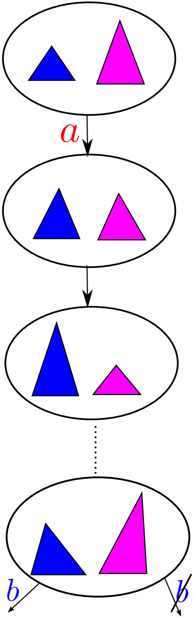


Theorem.

> There is an elementary function g such that for any
> det-FO grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ and $T \not\sim U$ of size n we have
>
> $$EL(T, U) \leq tower(g(n)).$$

$tower(0) = 1$
$tower(n+1) = 2^{tower(n)}$

Bounding lengths of witnesses (where EL keeps dropping)



Theorem.

There is an elementary function g such that for any
det-FO grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ and $T \not\sim U$ of size n we have

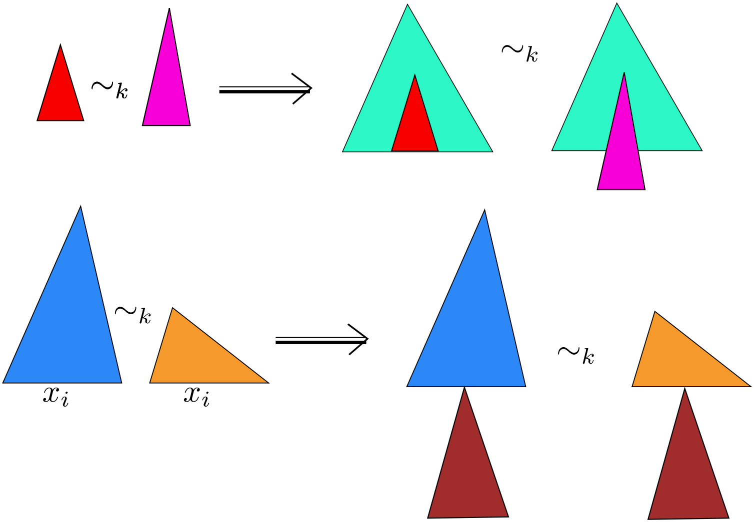
$$EL(T, U) \leq tower(g(n)).$$

$tower(0) = 1$
$tower(n{+}1) = 2^{tower(n)}$

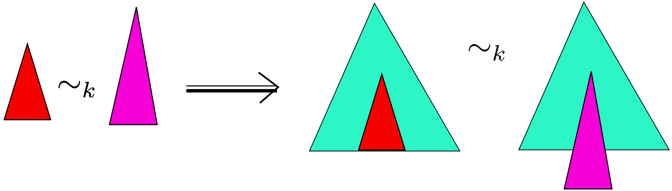
Proof is based on two ideas:

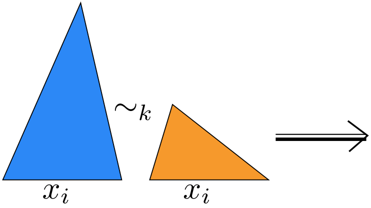
1. "Synchronize" the growth of lhs-terms and rhs-terms *while not changing the respective eq-levels*. (Hence no repeat.)
2. Derive a tower-bound on the size of terms in the (modified) sequence.

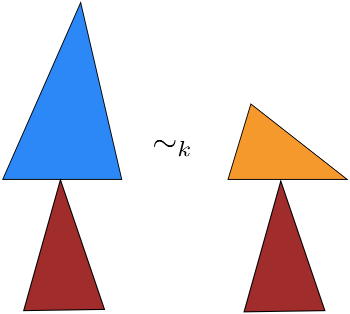
Congruence properties of $\sim_k$ and $\sim$



Congruence properties of $\sim_k$ and $\sim$

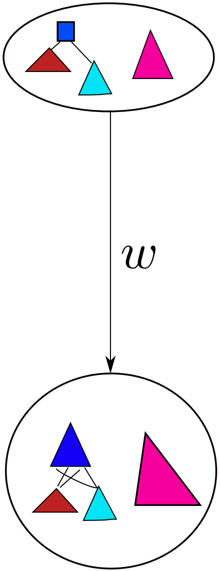




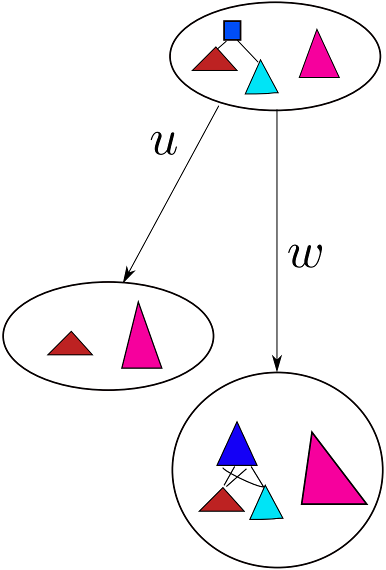


$$x_i \overset{\not\sim_1}{\sim_0} H \quad (\text{for } H \neq x_i)$$

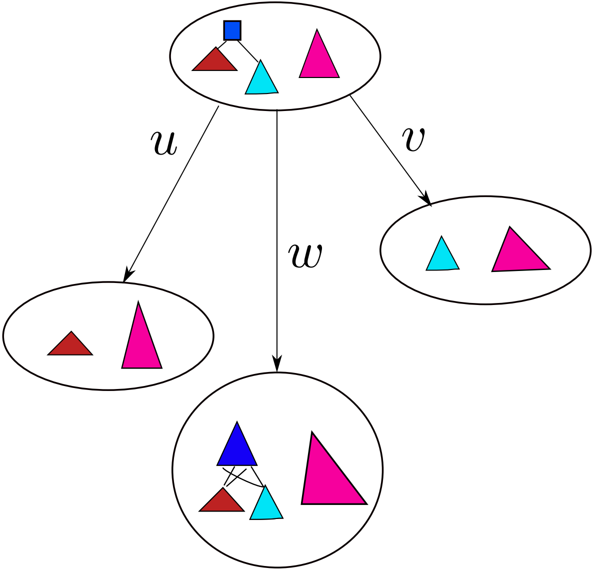
Balancing (the crucial tool for “synchronizing”)



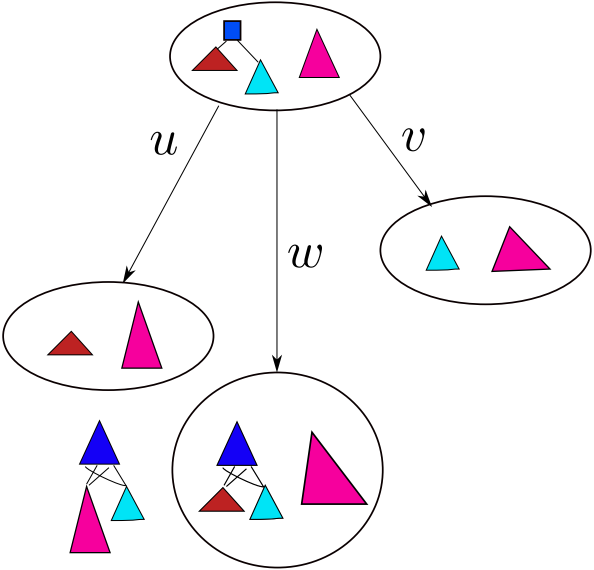
Balancing (the crucial tool for “synchronizing”)



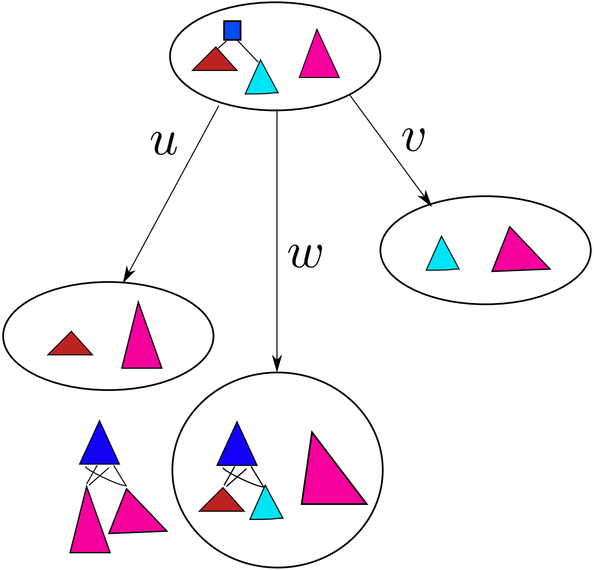
Balancing (the crucial tool for "synchronizing")



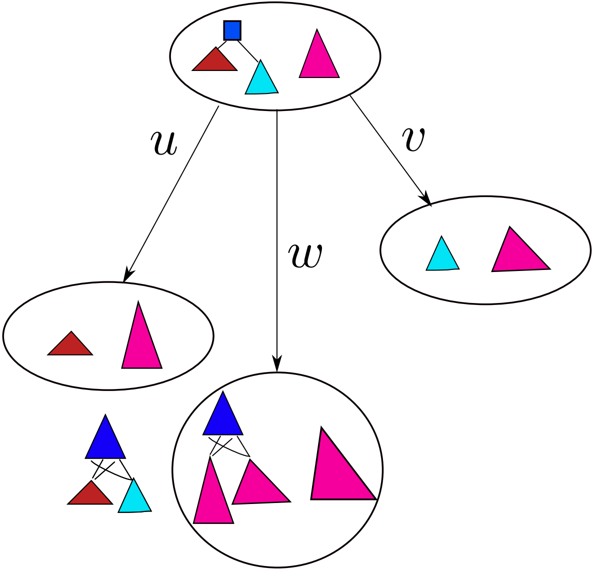
Balancing (the crucial tool for “synchronizing”)



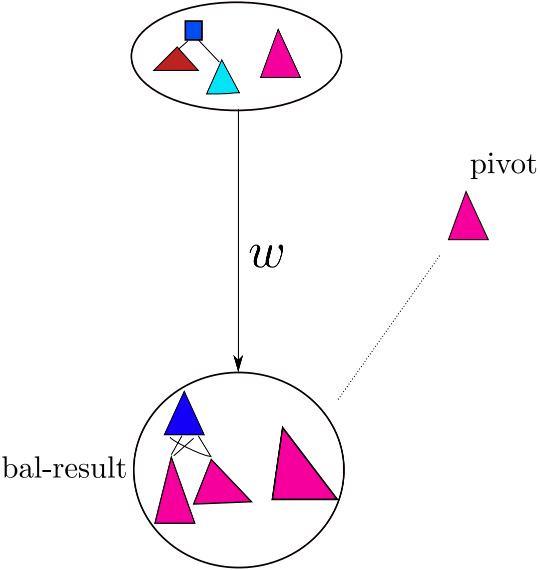
Balancing (the crucial tool for "synchronizing")



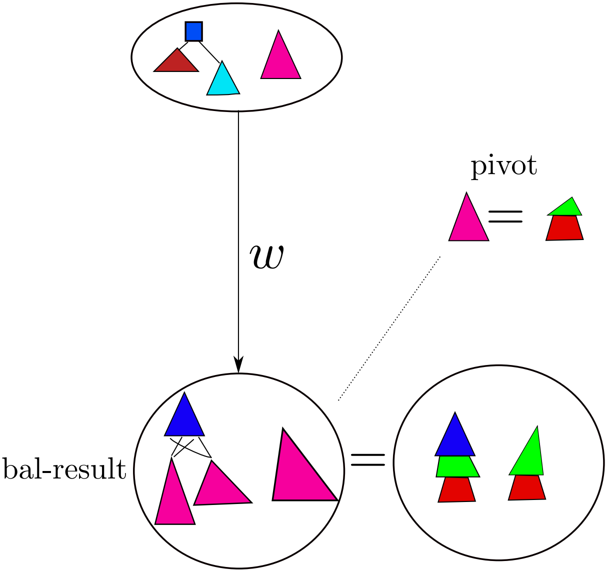
Balancing (the crucial tool for "synchronizing")



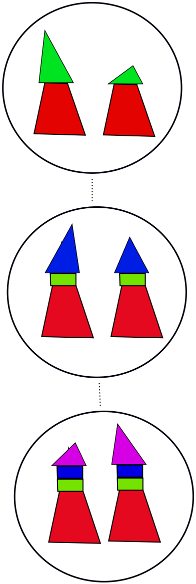
Balancing (the crucial tool for "synchronizing")



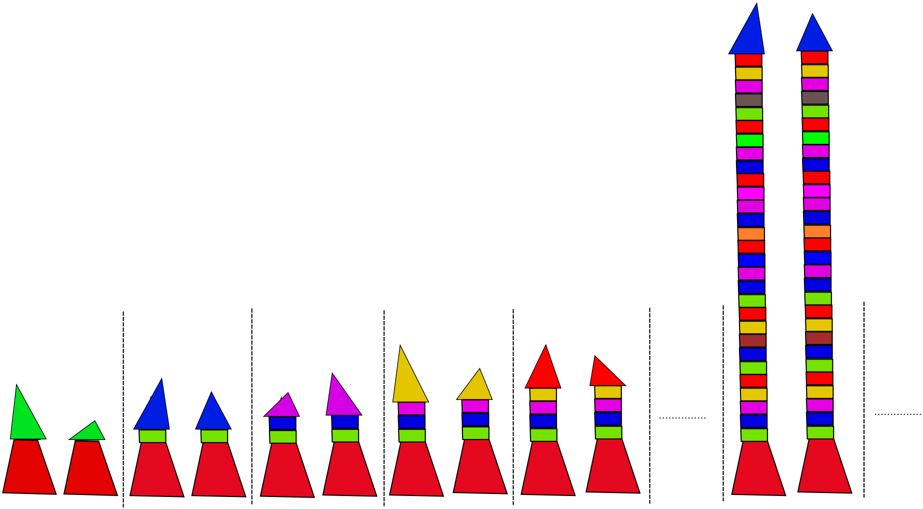
Balancing (the crucial tool for “synchronizing”)



“Stair subsequence” of pairs (on balanced witness path)



Stair subsequence of pairs (written horizontally)



(ℓ, n)-(sub)sequences, with 2^ℓ pairs

(ℓ, n)-(sub)sequences, with 2^ℓ pairs

$(1, n)$-sequence

2^1 pairs

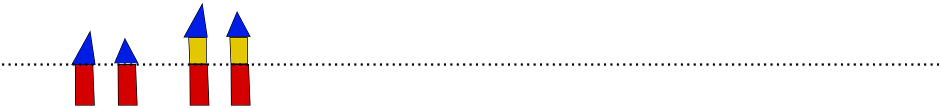
n ... thickness

There is no EL-decreasing $(1,0)$-sequence.

$(1,n)$-sequence

2^1 pairs

n ... thickness



(ℓ, n)-(sub)sequences, with 2^ℓ pairs

There is no EL-decreasing $(1, 0)$-sequence.

$(1, n)$-sequence

q ... cardinality of "alphabet"

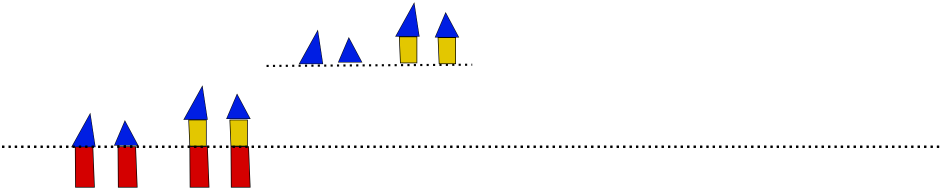
2^1 pairs

In $\boxed{h(1) = 1 + q}$ pairs (of thickness n)

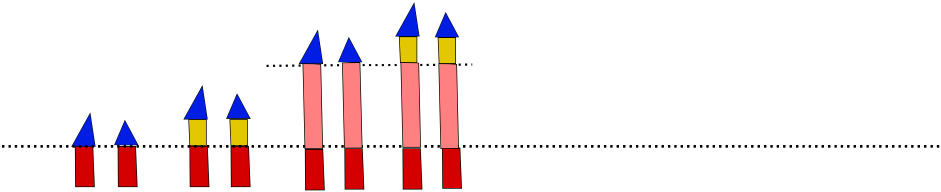
n ... thickness

there is some $(1, n)$-sequence.

(ℓ, n)-(sub)sequences, with 2^ℓ pairs



(ℓ, n)-(sub)sequences, with 2^ℓ pairs

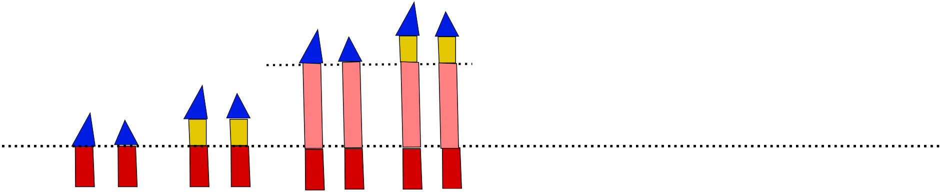


(ℓ, n)-(sub)sequences, with 2^ℓ pairs

$(2, n)$-sequence

$2^2 = 4$ pairs

n ... thickness



(ℓ, n)-(sub)sequences, with 2^ℓ pairs

q ... cardinality of "alphabet"

$(2,n)$-sequence

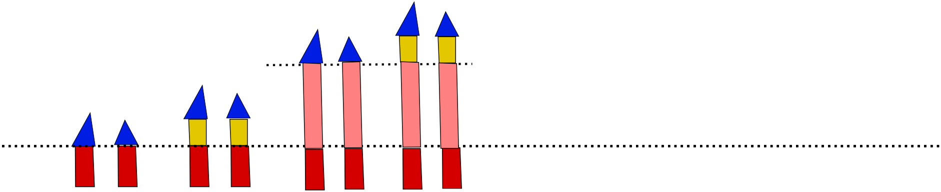
$h(1) = 1 + q$ ... $(1,n)$-sequence

$2^2 = 4$ pairs

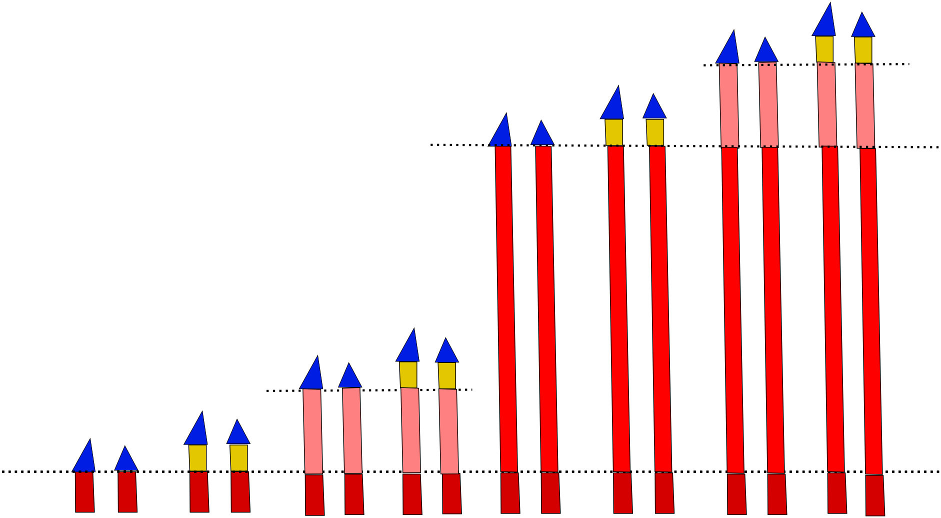
In $\boxed{h(2) = h(1) \cdot (1 + q^{h(1)})}$ pairs

n ... thickness

there is some $(2,n)$-sequence.



(ℓ, n)-(sub)sequences, with 2^ℓ pairs

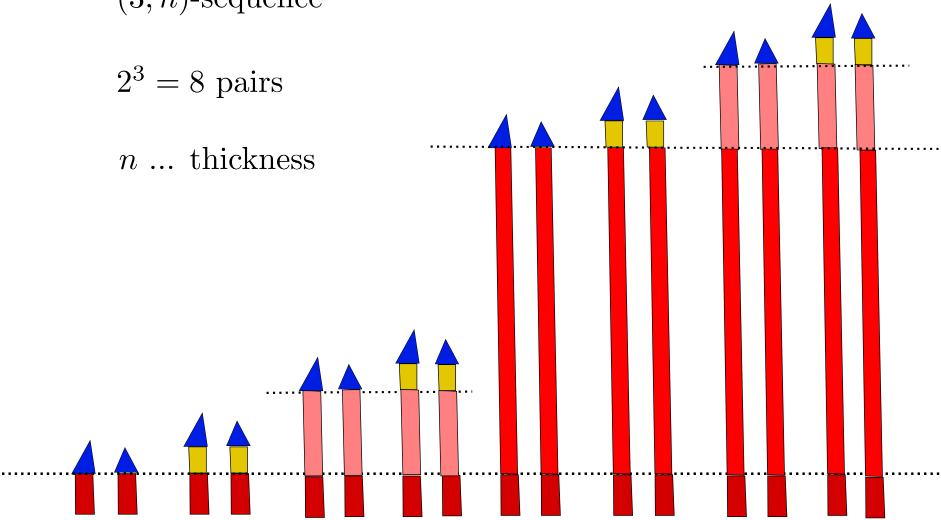


(ℓ, n)-(sub)sequences, with 2^ℓ pairs

$(3, n)$-sequence

$2^3 = 8$ pairs

n ... thickness



(ℓ, n)-(sub)sequences, with 2^ℓ pairs

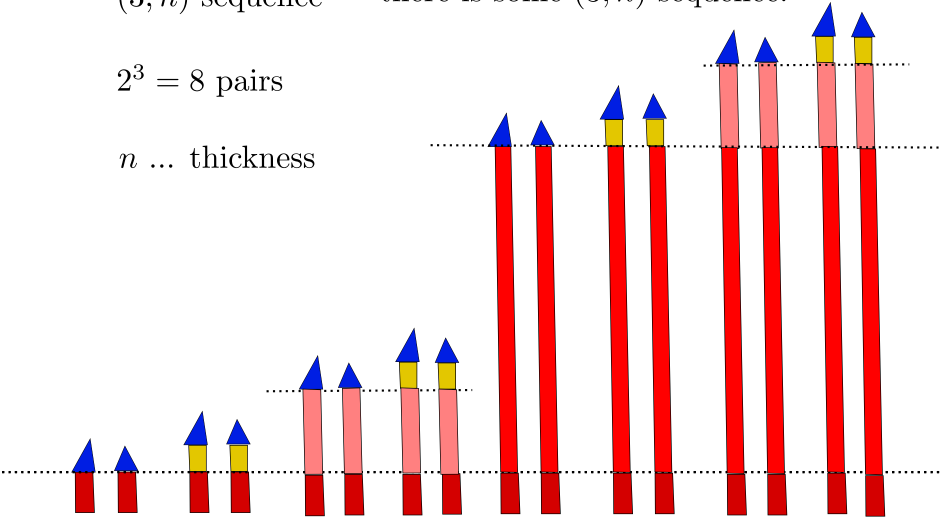
In $\boxed{h(3) = h(2) \cdot (1 + q^{h(2)})}$ pairs

there is some $(3, n)$-sequence.

$(3, n)$-sequence

$2^3 = 8$ pairs

n ... thickness



Final (conditional) step of the "TOWER-proof"

Recall: There is no EL-decreasing $(1, 0)$-sequence.

Final (conditional) step of the “TOWER-proof”

Recall: There is no EL-decreasing $(1, 0)$-sequence.

Claim. Any EL-decreasing $(\ell+1, n+1)$-sequence gives rise to an EL-decreasing (ℓ, n)-sequence.

Final (conditional) step of the "TOWER-proof"

Recall: There is no EL-decreasing $(1, 0)$-sequence.

Claim. Any EL-decreasing $(\ell+1, n+1)$-sequence gives rise to an EL-decreasing (ℓ, n)-sequence.

Corollary. There is no EL-decreasing $(n+1, n)$-sequence.

Final (conditional) step of the "TOWER-proof"

Recall: There is no EL-decreasing $(1,0)$-sequence.

Claim. Any EL-decreasing $(\ell{+}1, n{+}1)$-sequence gives rise to an EL-decreasing (ℓ, n)-sequence.

Corollary. There is no EL-decreasing $(n{+}1, n)$-sequence.

Recall that

$$h(1) = 1 + q,$$
$$h(j{+}1) = h(j) \cdot (1 + q^{h(j)})$$

and that $h(j)$ "stairs" gives rise to (j, n)-sequence (n being the "small" thickness).

Final (conditional) step of the “TOWER-proof”

Recall: There is no EL-decreasing $(1,0)$-sequence.

Claim. Any EL-decreasing $(\ell{+}1, n{+}1)$-sequence gives rise to an EL-decreasing (ℓ, n)-sequence.

Corollary. There is no EL-decreasing $(n{+}1, n)$-sequence.

Recall that

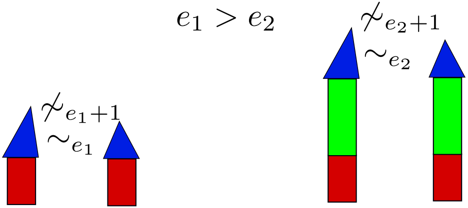
$$h(1) = 1 + q,$$
$$h(j{+}1) = h(j) \cdot (1 + q^{h(j)})$$

and that $h(j)$ “stairs” gives rise to (j, n)-sequence (n being the “small” thickness).

Corollary. There are less than $h(n{+}1)$ stairs, and $h(n{+}1) \leq \mathit{tower}(g(n))$.

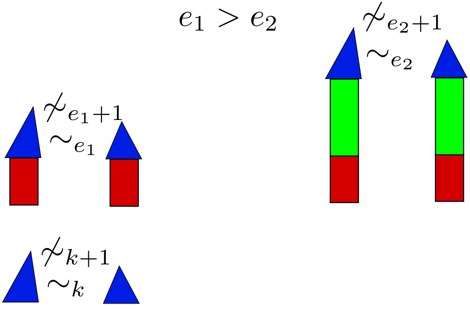
Repeating heads yield an "equation"

$e_1 > e_2$



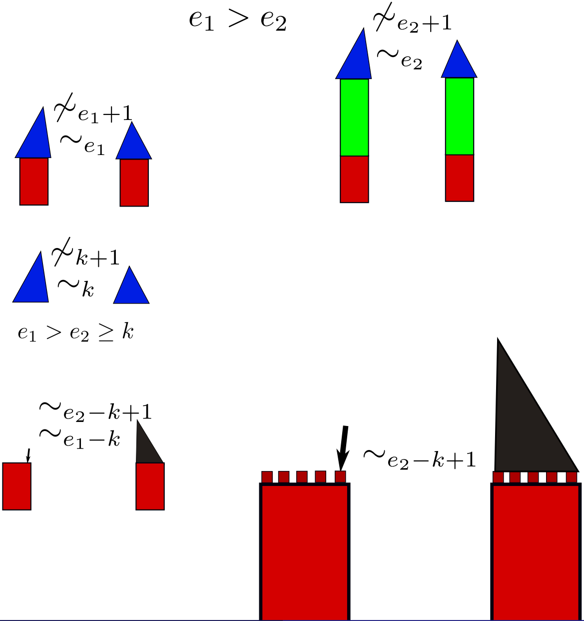
Repeating heads yield an "equation"

$e_1 > e_2$

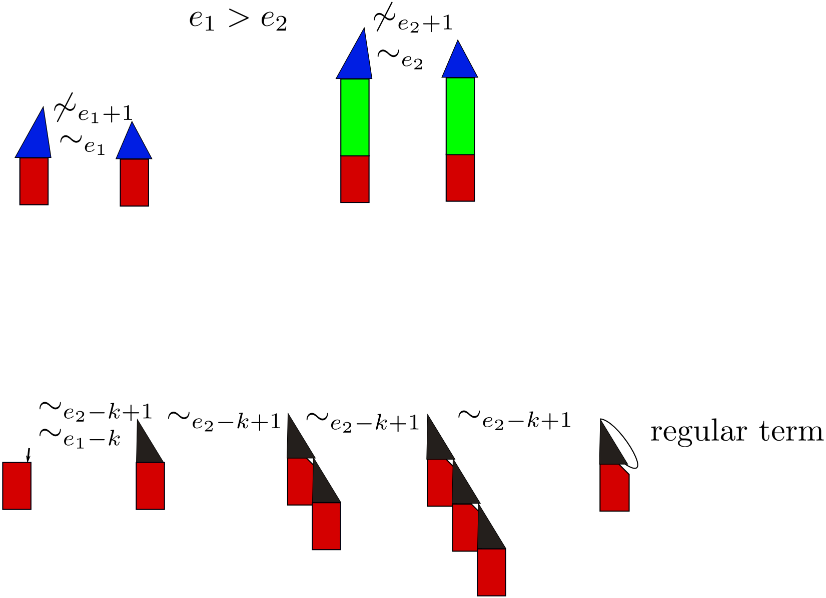


$e_1 > e_2 \geq k$

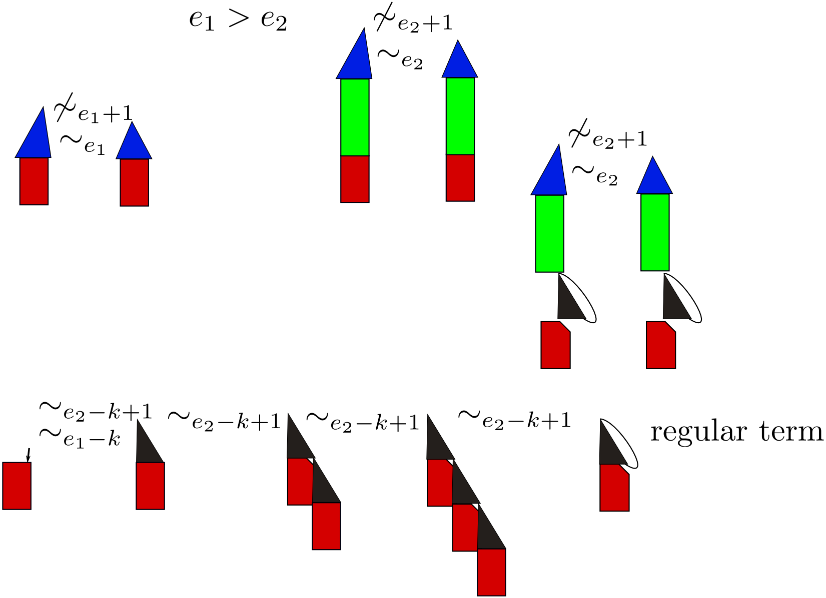
Repeating heads yield an "equation"



Repeating heads yield an "equation"

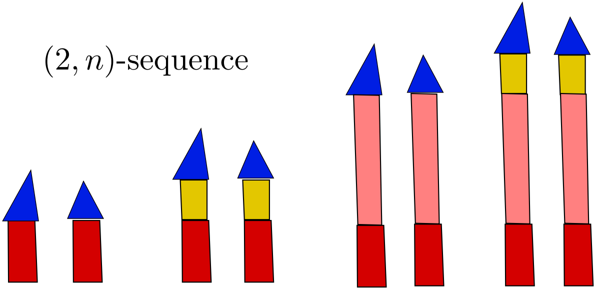


Repeating heads yield an "equation"

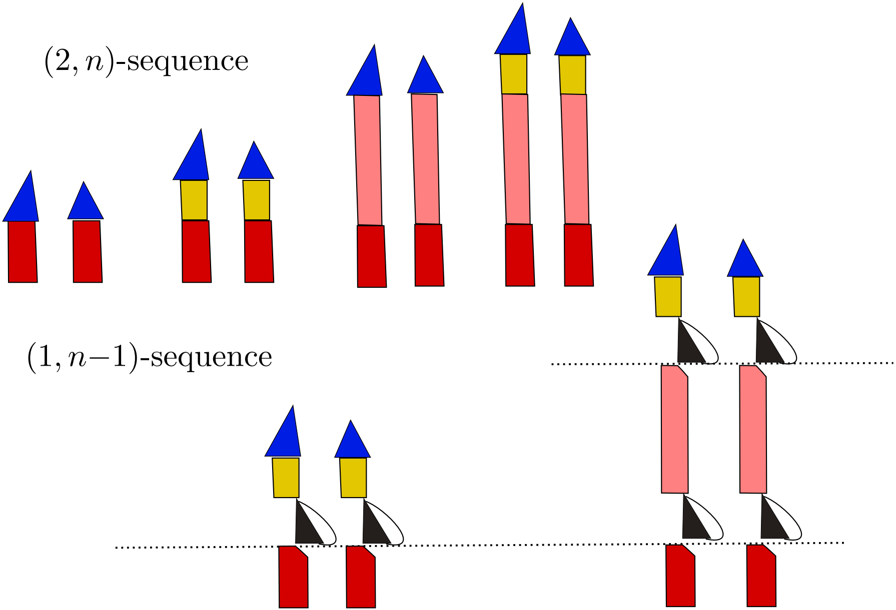


From (ℓ, n) to $(\ell-1, n-1)$ ... decreasing thickness

$(2, n)$-sequence

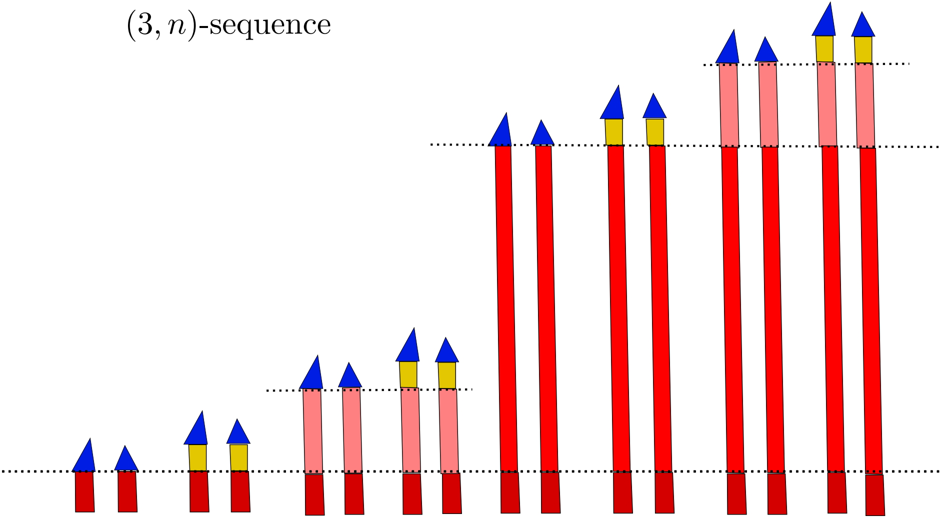


From (ℓ, n) to $(\ell-1, n-1)$ ... decreasing thickness

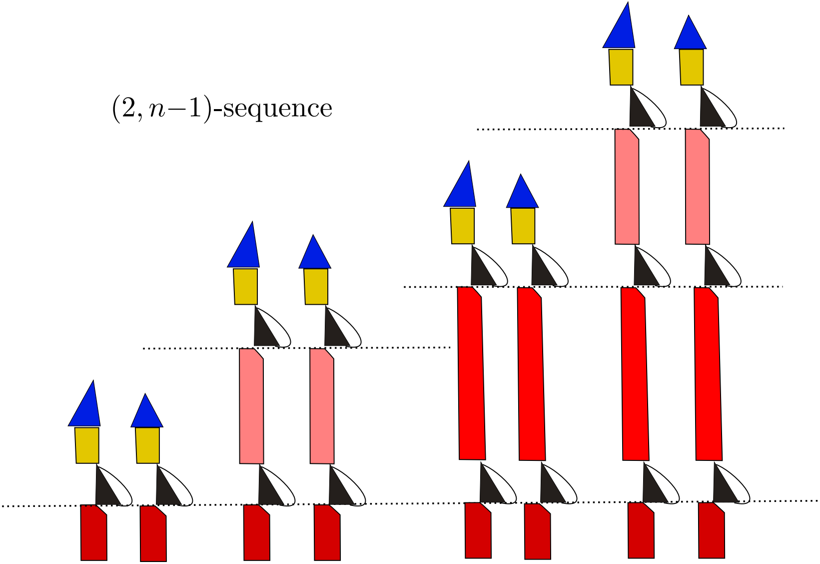


From (ℓ, n) to $(\ell-1, n-1)$ ... decreasing thickness

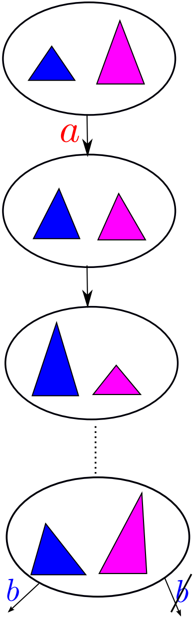
$(3, n)$-sequence



From (ℓ, n) to $(\ell-1, n-1)$ ... decreasing thickness



Bounding lengths of witnesses in the deterministic case



Theorem.

There is an elementary function g such that for any det-FO grammar $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ and $T \not\sim U$ of size n we have

$$EL(T,U) \leq tower(g(n)).$$

Proof is based on two ideas:

1. "Synchronize" the growth of lhs-terms and rhs-terms *while not changing the respective eq-levels*. (Hence no repeat.)
2. Derive a tower-bound on the size of terms in the (modified) sequence.

A lower bound

Bisimulation equivalence for FO-grammars is Ackermann-hard.

Note:
Benedikt M., Göller S., Kiefer S., Murawski A.S.:
Bisimilarity of Pushdown Automata is Nonelementary. LICS 2013
(no ε-transitions)

Ackermann function, class ACK, ACK-completeness

Family $f_0, f_1, f_2, \ldots$ of functions:

$$f_0(n) = n{+}1$$
$$f_{k+1}(n) = f_k(f_k(\ldots f_k(n)\ldots)) = f_k^{(n+1)}(n)$$

Ackermann function f_A: $f_A(n) = f_n(n)$.

Ackermann function, class ACK, ACK-completeness

Family $f_0, f_1, f_2, \ldots$ of functions:

$$f_0(n) = n+1$$
$$f_{k+1}(n) = f_k(f_k(\ldots f_k(n)\ldots)) = f_k^{(n+1)}(n)$$

Ackermann function f_A: $f_A(n) = f_n(n)$.

ACK ... class of problems solvable in time $f_A(g(n))$ where g is a primitive recursive function.

Ackermann function, class ACK, ACK-completeness

Family $f_0, f_1, f_2, \ldots$ of functions:

$$f_0(n) = n{+}1$$
$$f_{k+1}(n) = f_k(f_k(\ldots f_k(n)\ldots)) = f_k^{(n+1)}(n)$$

Ackermann function f_A: $f_A(n) = f_n(n)$.

ACK ... class of problems solvable in time $f_A(g(n))$
where g is a primitive recursive function.

Ackermann-budget halting problem (AB-HP):

Instance: Minsky counter machine M.
Question: does M halt from the zero initial configuration within $f_A(size(M))$ steps ?

Fact. AB-HP is ACK-complete.

Control state reachability in reset counter machines

Reset counter machines (RCMs).

nonnegative counters $c_1, c_2, \ldots, c_d$,

control states $1, 2, \ldots, r$,

configuration $(\ell, (n_1, n_2, \ldots, n_d))$, initial conf. $(1, (0, 0, \ldots, 0))$,

(nondeterministic) instructions of the types

$\ell \xrightarrow{inc(c_i)} \ell'$ *(increment c_i),*

$\ell \xrightarrow{dec(c_i)} \ell'$ *(decrement c_i, if $c_i > 0$),*

$\ell \xrightarrow{reset(c_i)} \ell'$ *(reset c_i, i.e., put $c_i = 0$).*

Control state reachability in reset counter machines

Reset counter machines (RCMs).

nonnegative counters $c_1, c_2, \ldots, c_d$,

control states $1, 2, \ldots, r$,

configuration $(\ell, (n_1, n_2, \ldots, n_d))$, initial conf. $(1, (0, 0, \ldots, 0))$,

(nondeterministic) instructions of the types

$\ell \xrightarrow{inc(c_i)} \ell'$ *(increment c_i),*

$\ell \xrightarrow{dec(c_i)} \ell'$ *(decrement c_i, if $c_i > 0$),*

$\ell \xrightarrow{reset(c_i)} \ell'$ *(reset c_i, i.e., put $c_i = 0$).*

CS-reach problem for RCM:

Instance: an RCM M, a control state ℓ_{FIN} .

Question: is $(1, (0, 0, \ldots, 0)) \longrightarrow^* (\ell_{\text{FIN}}, (\ldots))$?

Control state reachability in reset counter machines

Reset counter machines (RCMs).
nonnegative counters $c_1, c_2, \ldots, c_d$,
control states $1, 2, \ldots, r$,
configuration $(\ell, (n_1, n_2, \ldots, n_d))$, initial conf. $(1, (0, 0, \ldots, 0))$,
(nondeterministic) instructions of the types

$\ell \xrightarrow{inc(c_i)} \ell'$ *(increment c_i),*
$\ell \xrightarrow{dec(c_i)} \ell'$ *(decrement c_i, if $c_i > 0$),*
$\ell \xrightarrow{reset(c_i)} \ell'$ *(reset c_i, i.e., put $c_i = 0$).*

CS-reach problem for RCM:

Instance: an RCM M, a control state ℓ_{FIN} .
Question: is $(1, (0, 0, \ldots, 0)) \longrightarrow^* (\ell_{\mathrm{FIN}}, (\ldots))$?

Fact. CS-reach problem for RCM is ACK-complete.
(See [Schnoebelen, MFCS 2010].)

Reduction of CS-reach for RCM to FO-bisimilarity

Given an RCM M, i.e.,

counters $c_1, c_2, \ldots, c_d$,
control states $1, 2, \ldots, r$,

and instructions of the types

$\ell \xrightarrow{inc(c_i)} \ell'$ *(increment* c_i*)*,
$\ell \xrightarrow{dec(c_i)} \ell'$ *(decrement* c_i*, if* $c_i > 0$*)*,
$\ell \xrightarrow{reset(c_i)} \ell'$ *(reset* c_i*, i.e., put* $c_i = 0$*)*,

and ℓ_{FIN},

we construct $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{R})$ and E_0, F_0 so that

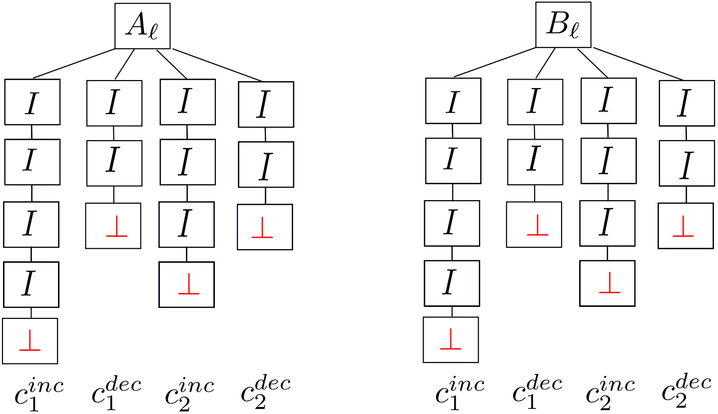
$$(1, (0, 0, \ldots, 0)) \longrightarrow^* (\ell_{\mathrm{FIN}}, (\ldots)) \quad \textit{iff} \quad E_0 \not\sim F_0.$$

CS-reachability as bisimulation game

Example with counters c_1, c_2; we start with the pair

$$(A_1(\perp, \perp, \perp, \perp,), B_1(\perp, \perp, \perp, \perp)).$$

The pair after mimicking $(1, (0, 0)) \longrightarrow^* \boxed{(\ell, (2, 1))}$ might be



Attacker's win

Attacker wins in

$$(A_{\ell_{\mathrm{FIN}}}(\dots), B_{\ell_{\mathrm{FIN}}}(\dots))$$

due to the rule $A_{\ell_{\mathrm{FIN}}}(x_1, x_2, x_3, x_4) \xrightarrow{a} \dots$ (while there is no rule for $B_{\ell_{\mathrm{FIN}}}$).

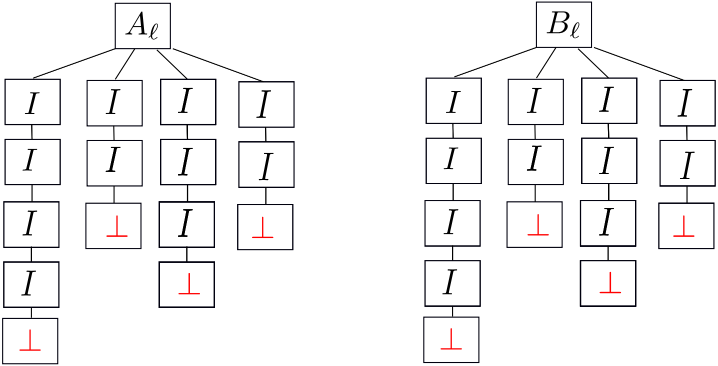
Counter increment

For $\boxed{ins = \ell \xrightarrow{inc(c_2)} \ell'}$

we have rules

$A_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} A_{\ell'}(x_1, x_2, I(x_3), x_4)$,

$B_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} B_{\ell'}(x_1, x_2, I(x_3), x_4)$,



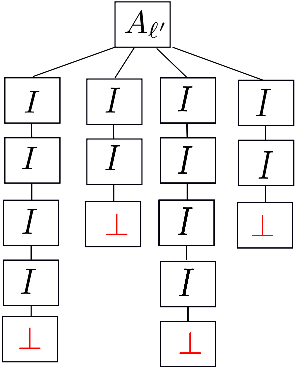
Counter increment

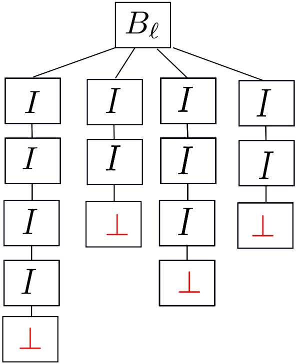
For $\boxed{ins = \ell \stackrel{inc(c_2)}{\longrightarrow} \ell'}$

we have rules

$A_\ell(x_1, x_2, x_3, x_4) \stackrel{ins}{\longrightarrow} A_{\ell'}(x_1, x_2, I(x_3), x_4)$,

$B_\ell(x_1, x_2, x_3, x_4) \stackrel{ins}{\longrightarrow} B_{\ell'}(x_1, x_2, I(x_3), x_4)$,





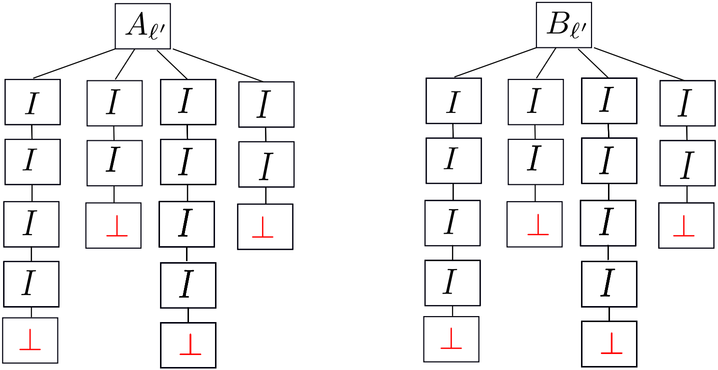
Counter increment

For $\boxed{ins = \ell \xrightarrow{inc(c_2)} \ell'}$

we have rules

$A_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} A_{\ell'}(x_1, x_2, I(x_3), x_4)$,

$B_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} B_{\ell'}(x_1, x_2, I(x_3), x_4)$,



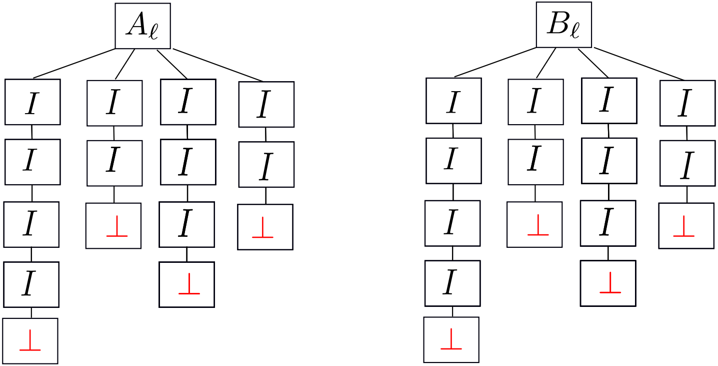
Counter reset

For $\boxed{ins = \ell \xrightarrow{reset(c_2)} \ell'}$

we have rules

$A_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} A_{\ell'}(x_1, x_2, \perp, \perp)$,

$B_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} B_{\ell'}(x_1, x_2, \perp, \perp)$,



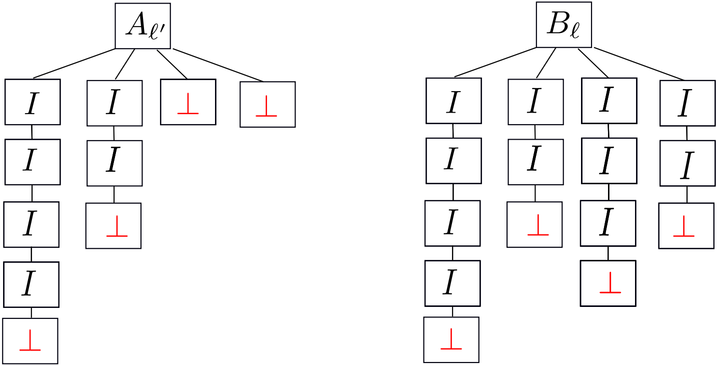
Counter reset

For $\boxed{ins = \ell \xrightarrow{reset(c_2)} \ell'}$

we have rules

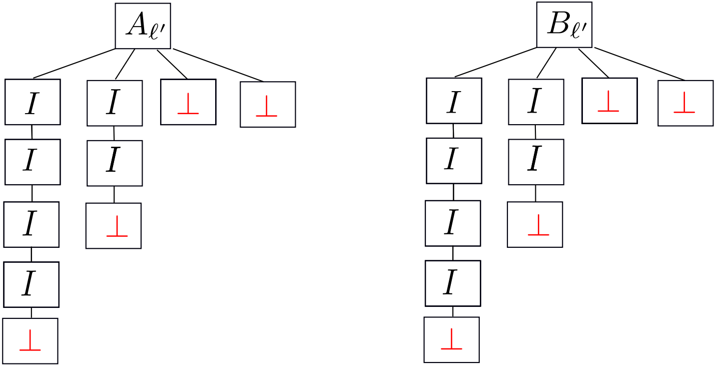
$A_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} A_{\ell'}(x_1, x_2, \perp, \perp)$,

$B_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} B_{\ell'}(x_1, x_2, \perp, \perp)$,



Counter reset

For $\boxed{ins = \ell \xrightarrow{reset(c_2)} \ell'}$
we have rules
$A_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} A_{\ell'}(x_1, x_2, \perp, \perp)$,
$B_\ell(x_1, x_2, x_3, x_4) \xrightarrow{ins} B_{\ell'}(x_1, x_2, \perp, \perp)$,

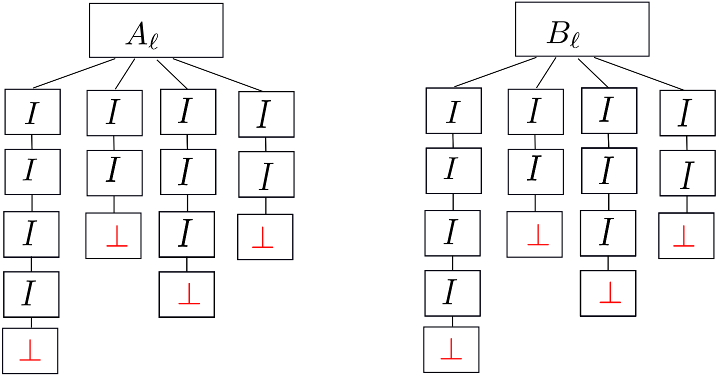


Counter decrement

For $\boxed{ins = \ell \xrightarrow{dec(c_2)} \ell'}$ we have two phases; the first-phase rules are

$$A_\ell \xrightarrow{ins} A_{(\ell',2)},\ A_\ell \xrightarrow{ins} B_{(\ell',2,a)},\ A_\ell \xrightarrow{ins} B_{(\ell',2,b)},$$

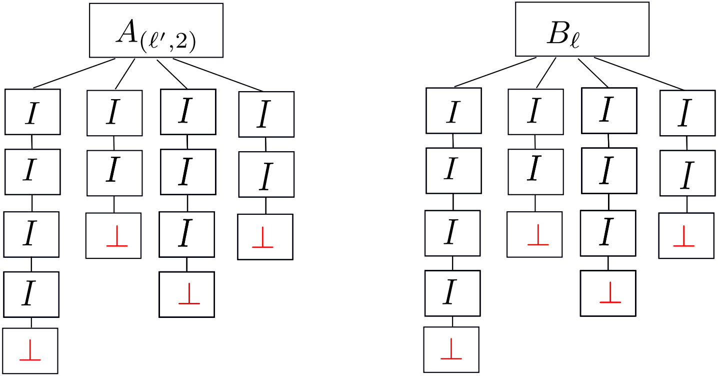
$$B_\ell \xrightarrow{ins} B_{(\ell',2,a)},\ B_\ell \xrightarrow{ins} B_{(\ell',2,b)},$$



Counter decrement

For $\boxed{ins = \ell \xrightarrow{dec(c_2)} \ell'}$ we have two phases; the first-phase rules are
$A_\ell \xrightarrow{ins} A_{(\ell',2)}$, $A_\ell \xrightarrow{ins} B_{(\ell',2,a)}$, $A_\ell \xrightarrow{ins} B_{(\ell',2,b)}$,

$$B_\ell \xrightarrow{ins} B_{(\ell',2,a)},\ B_\ell \xrightarrow{ins} B_{(\ell',2,b)},$$

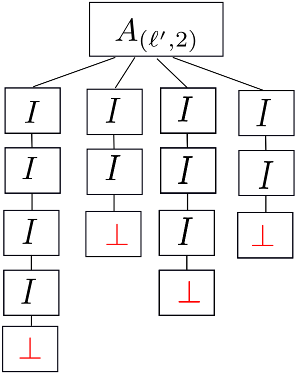


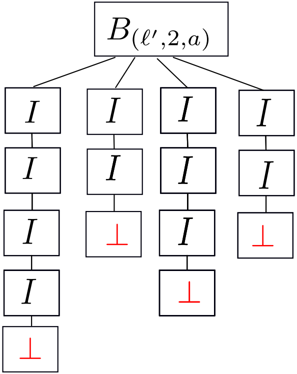
Counter decrement

For $\boxed{ins = \ell \stackrel{dec(c_2)}{\longrightarrow} \ell'}$ we have two phases; the first-phase rules are

$A_\ell \stackrel{ins}{\longrightarrow} A_{(\ell',2)}$, $A_\ell \stackrel{ins}{\longrightarrow} B_{(\ell',2,a)}$, $A_\ell \stackrel{ins}{\longrightarrow} B_{(\ell',2,b)}$,

$B_\ell \stackrel{ins}{\longrightarrow} B_{(\ell',2,a)}$, $B_\ell \stackrel{ins}{\longrightarrow} B_{(\ell',2,b)}$,

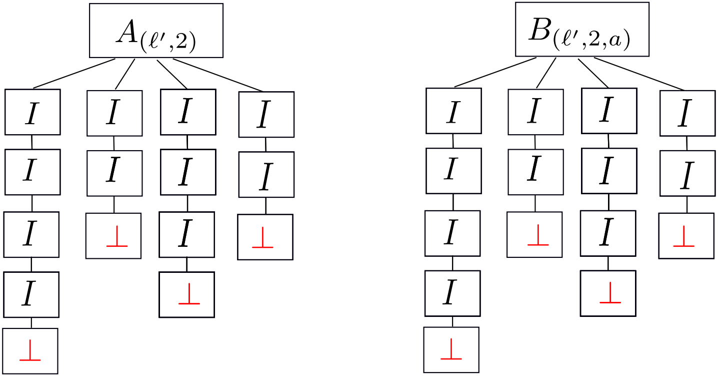




Counter decrement (option a)

$A_{(\ell',2)}(x_1,x_2,x_3,x_4) \xrightarrow{a} A_{\ell'}(x_1,x_2,x_3,I(x_4)),\ A_{\ell',2}(x_1,x_2,x_3,x_4) \xrightarrow{b} x_3,$
$B_{(\ell',2,a)}(x_1,x_2,x_3,x_4) \xrightarrow{a} B_{\ell'}(x_1,x_2,x_3,I(x_4)),$

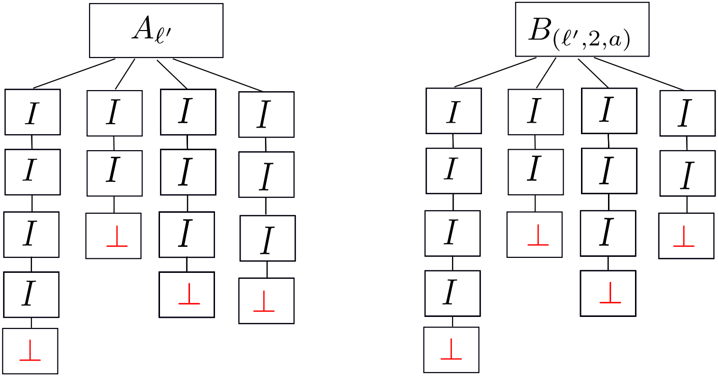
$$B_{(\ell',2,a)}(x_1,x_2,x_3,x_4) \xrightarrow{b} x_3,$$



Counter decrement (option a)

$A_{(\ell',2)}(x_1, x_2, x_3, x_4) \xrightarrow{a} A_{\ell'}(x_1, x_2, x_3, I(x_4))$, $A_{\ell',2}(x_1, x_2, x_3, x_4) \xrightarrow{b} x_3$,
$B_{(\ell',2,a)}(x_1, x_2, x_3, x_4) \xrightarrow{a} B_{\ell'}(x_1, x_2, x_3, I(x_4))$,

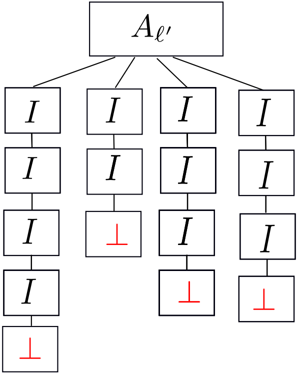
$B_{(\ell',2,a)}(x_1, x_2, x_3, x_4) \xrightarrow{b} x_3$,

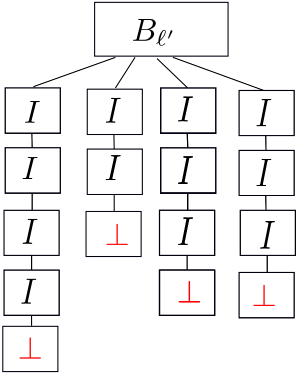


$$A_{(\ell',2)}(x_1,x_2,x_3,x_4) \xrightarrow{a} A_{\ell'}(x_1,x_2,x_3,I(x_4)),\ A_{\ell',2}(x_1,x_2,x_3,x_4) \xrightarrow{b} x_3,$$

$$B_{(\ell',2,a)}(x_1,x_2,x_3,x_4) \xrightarrow{a} B_{\ell'}(x_1,x_2,x_3,I(x_4)),$$

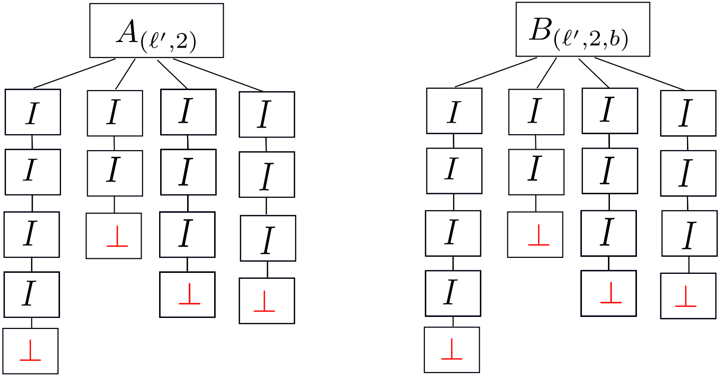
$$B_{(\ell',2,a)}(x_1,x_2,x_3,x_4) \xrightarrow{b} x_3,$$





Counter decrement (option b)

$A_{(\ell',2)}(x_1,x_2,x_3,x_4) \xrightarrow{a} A_{\ell'}(x_1,x_2,x_3,I(x_4))$, $A_{\ell',2}(x_1,x_2,x_3,x_4) \xrightarrow{b} x_3$,
$B_{(\ell',2,b)}(x_1,x_2,x_3,x_4) \xrightarrow{a} A_{\ell'}(x_1,x_2,x_3,I(x_4))$,
$B_{(\ell',2,b)}(x_1,x_2,x_3,x_4) \xrightarrow{b} x_4$,
$I(x_1) \xrightarrow{c} x_1$

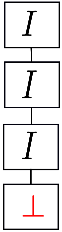


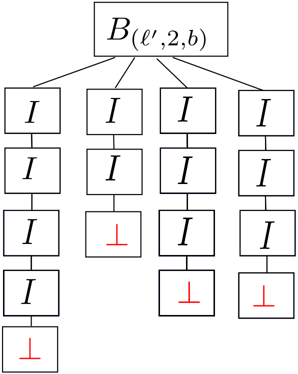
Counter decrement (option b)

$A_{(\ell',2)}(x_1, x_2, x_3, x_4) \xrightarrow{a} A_{\ell'}(x_1, x_2, x_3, I(x_4))$, $A_{\ell',2}(x_1, x_2, x_3, x_4) \xrightarrow{b} x_3$,
$B_{(\ell',2,b)}(x_1, x_2, x_3, x_4) \xrightarrow{a} A_{\ell'}(x_1, x_2, x_3, I(x_4))$,

$B_{(\ell',2,b)}(x_1, x_2, x_3, x_4) \xrightarrow{b} x_4$,

$I(x_1) \xrightarrow{c} x_1$



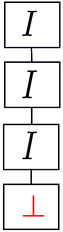


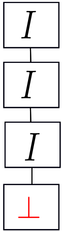
Counter decrement (option b)

$A_{(\ell',2)}(x_1, x_2, x_3, x_4) \xrightarrow{a} A_{\ell'}(x_1, x_2, x_3, I(x_4))$, $A_{\ell',2}(x_1, x_2, x_3, x_4) \xrightarrow{b} x_3$,
$B_{(\ell',2,b)}(x_1, x_2, x_3, x_4) \xrightarrow{a} A_{\ell'}(x_1, x_2, x_3, I(x_4))$,

$B_{(\ell',2,b)}(x_1, x_2, x_3, x_4) \xrightarrow{b} x_4$,

$I(x_1) \xrightarrow{c} x_1$





Final remarks

- (Trace) equivalence of deterministic FO-grammars is P-hard and in TOWER.
- Bisimulation equivalence of FO-grammars is Ackermann-hard and decidable.

Final remarks

- (Trace) equivalence of deterministic FO-grammars is P-hard and in TOWER.
- Bisimulation equivalence of FO-grammars is Ackermann-hard and decidable.

Branching bisimilarity (and weak bisimilarity):

- for normed BPA ExpTime-hard and decidable (Y. Fu, ICALP'13).

Final remarks

- (Trace) equivalence of deterministic FO-grammars is
  P-hard and in TOWER.
- Bisimulation equivalence of FO-grammars is
  Ackermann-hard and decidable.

Branching bisimilarity (and weak bisimilarity):

- for normed BPA
  ExpTime-hard and decidable (Y. Fu, ICALP'13).
- Czerwinski and J.: in NExpTime
  (withdrawn from Concur'14 [not finished then]; on arxiv soon)

Final remarks

- (Trace) equivalence of deterministic FO-grammars is P-hard and in TOWER.
- Bisimulation equivalence of FO-grammars is Ackermann-hard and decidable.

Branching bisimilarity (and weak bisimilarity):

- for normed BPA ExpTime-hard and decidable (Y. Fu, ICALP'13).
- Czerwinski and J.: in NExpTime (withdrawn from Concur'14 [not finished then]; on arxiv soon)
- undecidable for PDA (Fu and others, ICALP'14)

Final remarks

- (Trace) equivalence of deterministic FO-grammars is P-hard and in TOWER.
- Bisimulation equivalence of FO-grammars is Ackermann-hard and decidable.

Branching bisimilarity (and weak bisimilarity):

- for normed BPA ExpTime-hard and decidable (Y. Fu, ICALP'13).
- Czerwinski and J.: in NExpTime (withdrawn from Concur'14 [not finished then]; on arxiv soon)
- undecidable for PDA (Fu and others, ICALP'14)
- decidable for PDA with popping ε-moves
  - Fu and Yin: Dividing Line between Decidable PDA's and Undecidable Ones; arxiv.org/abs/1404.7015 (????)

Final remarks

- (Trace) equivalence of deterministic FO-grammars is P-hard and in TOWER.
- Bisimulation equivalence of FO-grammars is Ackermann-hard and decidable.

Branching bisimilarity (and weak bisimilarity):

- for normed BPA ExpTime-hard and decidable (Y. Fu, ICALP'13).
- Czerwinski and J.: in NExpTime (withdrawn from Concur'14 [not finished then]; on arxiv soon)
- undecidable for PDA (Fu and others, ICALP'14)
- decidable for PDA with popping ε-moves
  - Fu and Yin: Dividing Line between Decidable PDA's and Undecidable Ones; arxiv.org/abs/1404.7015 (????)
  - J.: Bisimulation Equivalence of First-Order Grammars; arxiv.org/abs/1405.7923