Improved Algorithms for the Automata-based Approach to Model-Checking

L. Doyen (EPFL) and J.-F. Raskin (ULB)

TACAS 2007
Braga - Portugal
March 28, 2007

Automata-based approach to model-checking

- Programs **and** properties are formalized as **regular languages** of **infinite words** ;
- Any **regular** language of infinite words is accepted by a **nondeterministic** Büchi automaton (NBW) ;
- The verification problem: given a NBW A (that formalizes Prg) and a NBW B (that formalizes Prop), check if $\mathbf{L(A) \subseteq L(B)}$.

Automata-based approach to model-checking

- The **language inclusion problem** for NBW is **PSpace-Complete** ;
- So, the complexity is rather high but **similar** (or easier than) to the complexity of many other verification problems ;
- Nevertheless, currently there is **no practical** algorithms to solve this language inclusion problem. The usual approach through **explicit complementation** is difficult.

Automata-based approach to model-checking

- Because there are **no practical** algorithms for language inclusion, we need either to:
    - restrict the properties to **LTL** or to languages expressible by **deterministic** Büchi automata (DBW). Those are easier to complement than NBW ;
    - specify the **bad behaviors**. Let C be a NBW that models bad behaviors then the verification problem boils down to check that $L(A) \cap L(C) = \varnothing$.

Plan of the talk

- Complementation of NBW
- Simulation pre-orders and fixed points
- An improved algorithm for emptiness of ABW
- The universality and language inclusion problems

Complementation of NBW

A **forty year** Saga (M. Vardi)

- 1961, Büchi: doubly exponential construction
- 1986, Sistla Vardi Wolper : simply exponential construction $O(2^{n^2})$
- 1988, Michel: lower bound $O(2^{n\log n})$
- 1989, Safra: (nearly) optimal solution $O(2^{n\log n})$ construction using determinization
- 1991, Klarlund: $O(2^{n\log n})$ construction without determinization
- 1997, Kurpferman Vardi : $O(2^{n\log n})$ similar to Klarlund but more modular
- 2004, Yang: slightly better lower bound $(0.76n)^n$
- 2004, Friedgut Kupferman Vardi: slightly better upper bound $(0.97n)^n$

Complementation of NBW

- **Few** attempts to implement the successive procedures:
  - Safra procedure have been implemented by Tasiran et al. (1995) and Thomas et al.(2005): need of **intricate data structures** and **very low scalability** (6 states);
  - KV procedure implemented by Gurumurthy et al. (2003): use **several optimisations** (based on simulation equivalences) but **very low scalability** (6 states);
  - Recently, Tabakov (2006) implemented KV with **BDDs** for checking universality but **very low scalability** (8 states).

KV construction ABW and AcoBW

- The KV construction uses **alternating** Büchi word (ABW) and alternating coBüchi word (AcoBW) automata
- Alternating automata are **generalizations** of nondeterministic Büchi automata
- Let $A=(Q,q_0,\Sigma,\delta,\alpha)$
  - in **nondeterministic** automata: $\delta(q,\sigma)= \{q_1,q_2,..,q_n\}$
  - in **alternating** automata: $\delta(q,\sigma)= \{\{q_1,q_2,..,q_n\},\{r_1,r_2,...,r_m\},...\}$

KV construction ABW and AcoBW

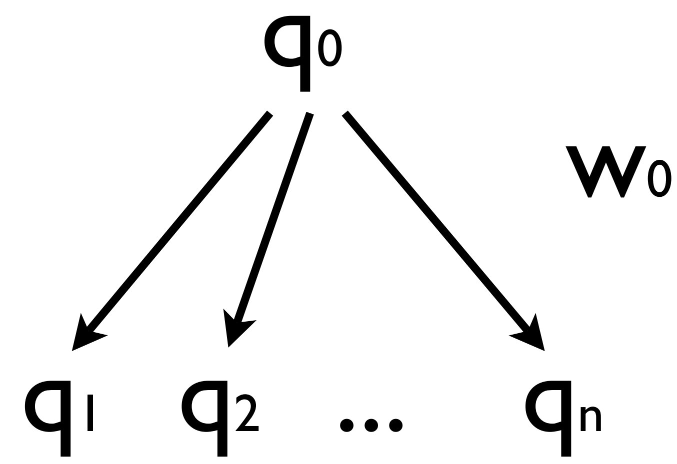
- The KV construction uses **alternating** Büchi word (ABW) and alternating coBüchi word (AcoBW) automata
- Alternating automata are **generalizations** of nondeterministic Büchi automata
- Let $A=(Q,q_0,\Sigma,\delta,\alpha)$
  - in **nondeterministic** automata: $\delta(q,\sigma)=\{q_1,q_2,..,q_n\}$ equivalent to $\{\{q_1\},\{q_2\},..,\{q_n\}\}$
  - in **alternating** automata: $\delta(q,\sigma)=\{\{q_1,q_2,..,q_n\},\{r_1,r_2,...,r_m\},...\}$

Run of an ABW $A=(Q,q_0,\Sigma,\delta,\alpha)$
on a word $w=w_0w_1\ldots w_n\ldots$

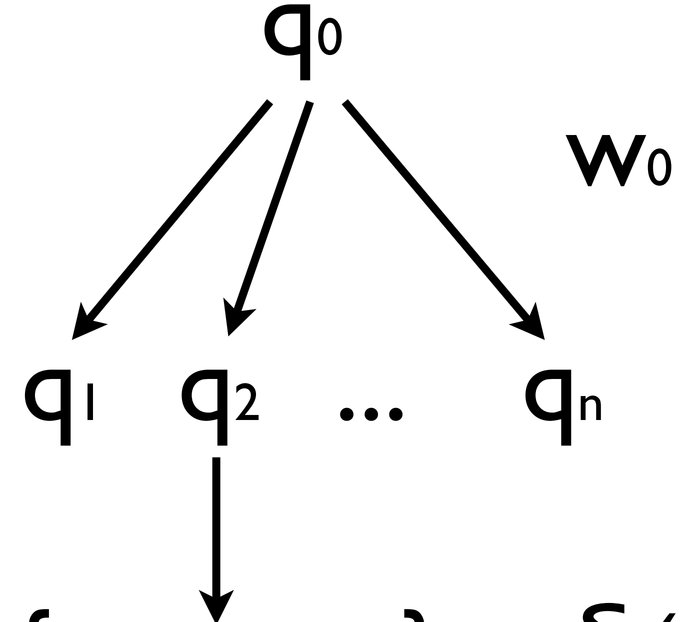
$$q_0 \xrightarrow{w_0}$$

Choose $\{q_1,q_2,..,q_n\} \in \delta(q_0,w_0)$

Run of an ABW $A=(Q,q_0,\Sigma,\delta,\alpha)$ on a word $w=w_0w_1\ldots w_n\ldots$



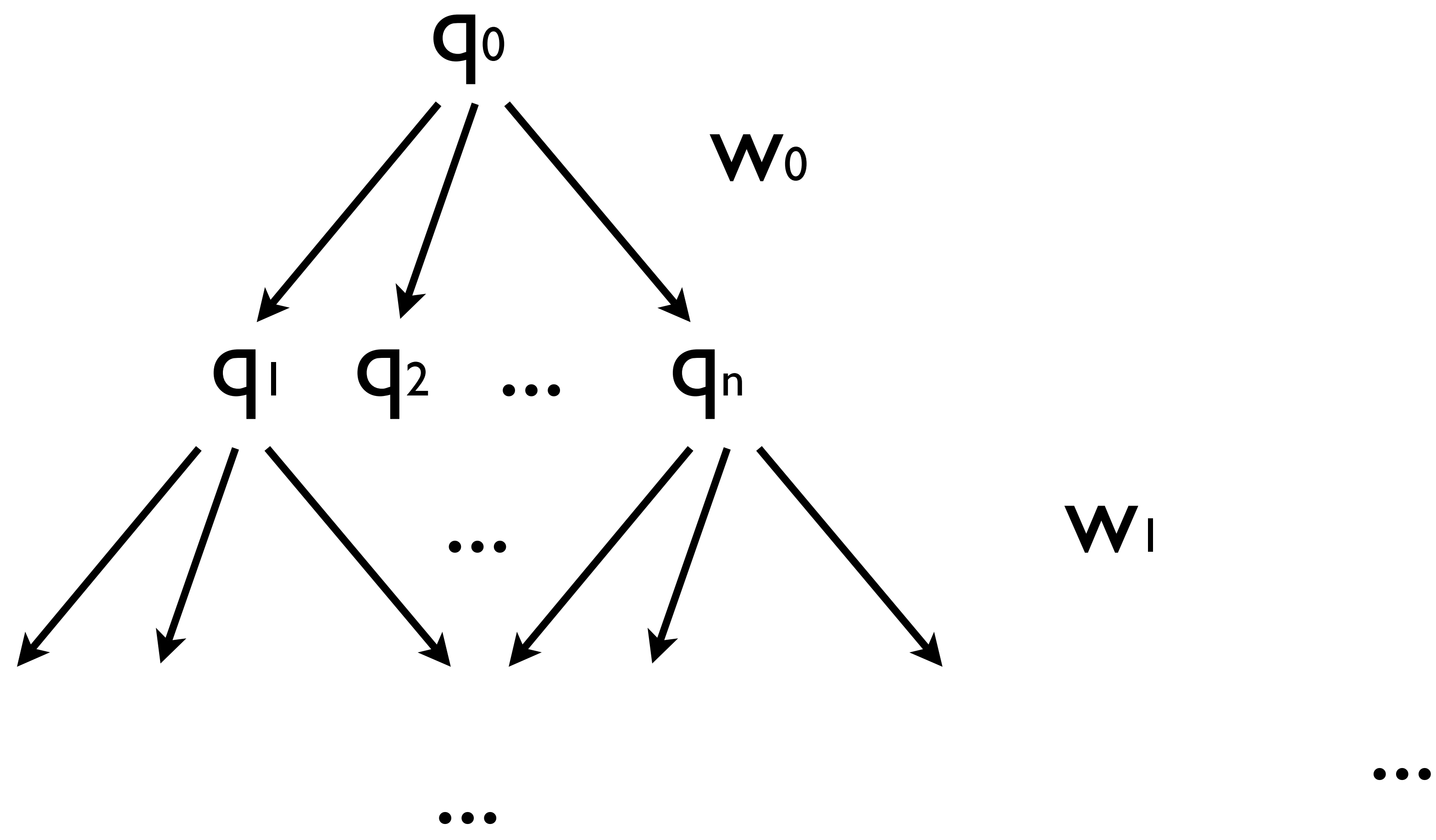
Run of an ABW $A=(Q,q_0,\Sigma,\delta,\alpha)$ on a word $w=w_0w_1\ldots w_n\ldots$



Choose $\{r_1,r_2,\ldots,r_m\} \in \delta(q_2,w_1)$

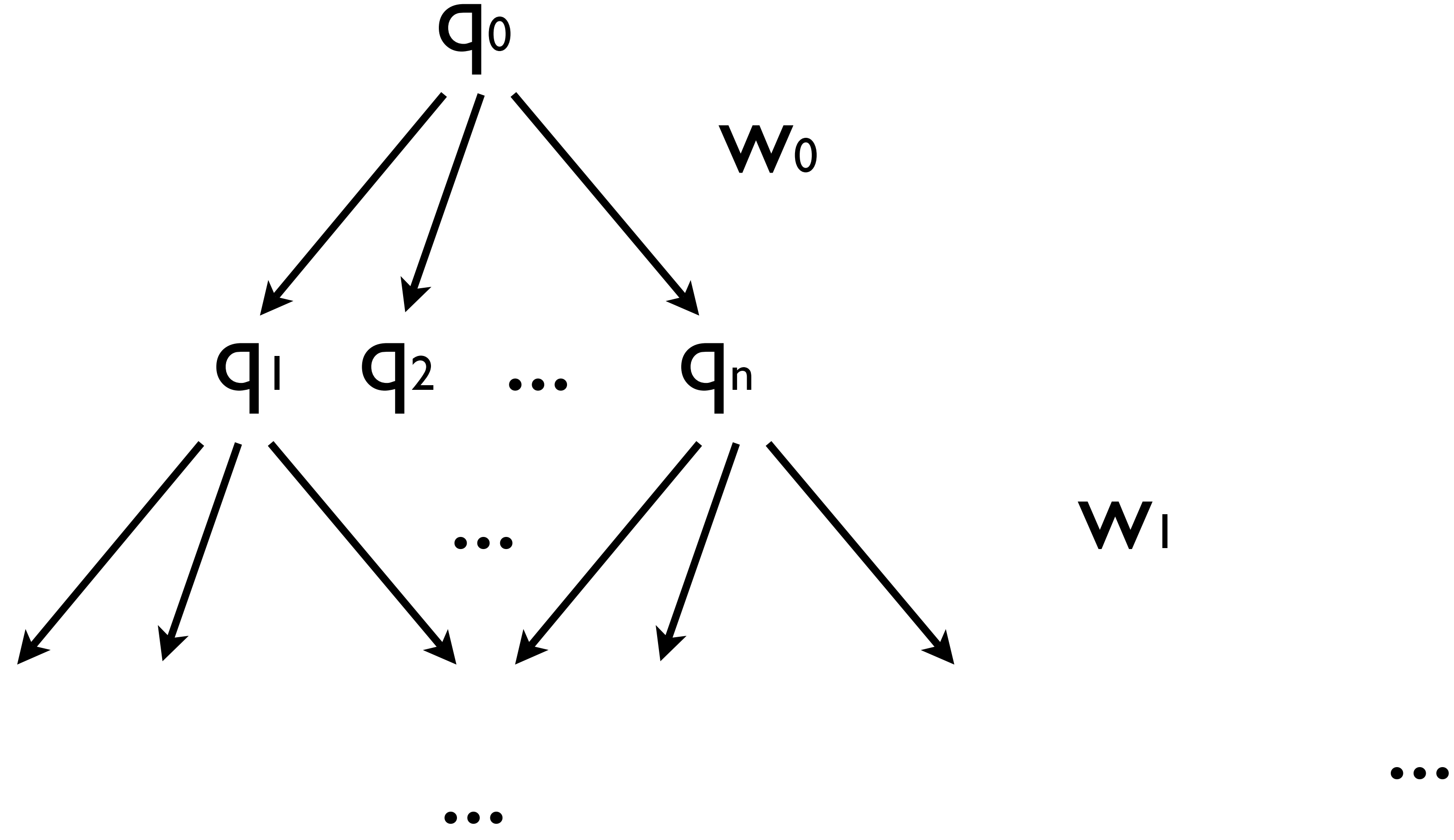
for each q_i of previous layer

Run of an ABW $A=(Q,q_0,\Sigma,\delta,\alpha)$ on a word $w=w_0w_1\ldots w_n\ldots$



The run is **accepting** if every branch intersects **infinitely often** α

Run of an AcoBW $A=(Q,q_0,\Sigma,\delta,\alpha)$ on a word $w=w_0w_1\ldots w_n\ldots$



The run is **accepting** if every branch intersects **only finitely often** α

KV construction

Input: A an NBW

B an AcoBW that accepts the complement of A

C an ABW that accepts the same language as B

Output: D an NBW that accepts the same language as C

KV construction

Input: A an NBW

This step is trivial O(1)

B an AcoBW that accepts the complement of A

C an ABW that accepts the same language as B

Output: D an NBW that accepts the same language as C

KV construction

- Let A be an **NBW** with transition relation δ ;
- Let B be an **AcoBW** identical to A but with transition relation δ' defined as follows: for all $q \in Q$: for all $\sigma \in \Sigma$:

  if $\delta(q,\sigma)=\{\{q_1\},\{q_2\},\ldots,\{q_n\}\}$ **then** $\delta'(q,\sigma) =\{\{q_1,q_2,\ldots,q_n\}\}$;
- So in B, we have **dualized** the transition relation: a run of the AcoBW on a word w is the tree that contains the set of **all** runs of the NBW on w ;
- ... and the accepting condition: B has an accepting run (tree) on w iff all the runs of A are rejecting ;
- So, **B accepts the complement of A**.

KV construction

Input: A an NBW

B an AcoBW that accepts the complement of A

This step is conceptually interesting and costs $O(n^2)$

C an ABW that accepts the same language as B

Output: D an NBW that accepts the same language as C

Accepting runs of AcoBW

- Accepting runs of AcoBW are **memoryless** (Emerson and Jutla, 1991).
- Memoryless runs are structured and that structure can be exploited to transform an AcoBW **into** an **ABW** (Kupferman and Vardi, 1997).

KV construction

Input: A an NBW

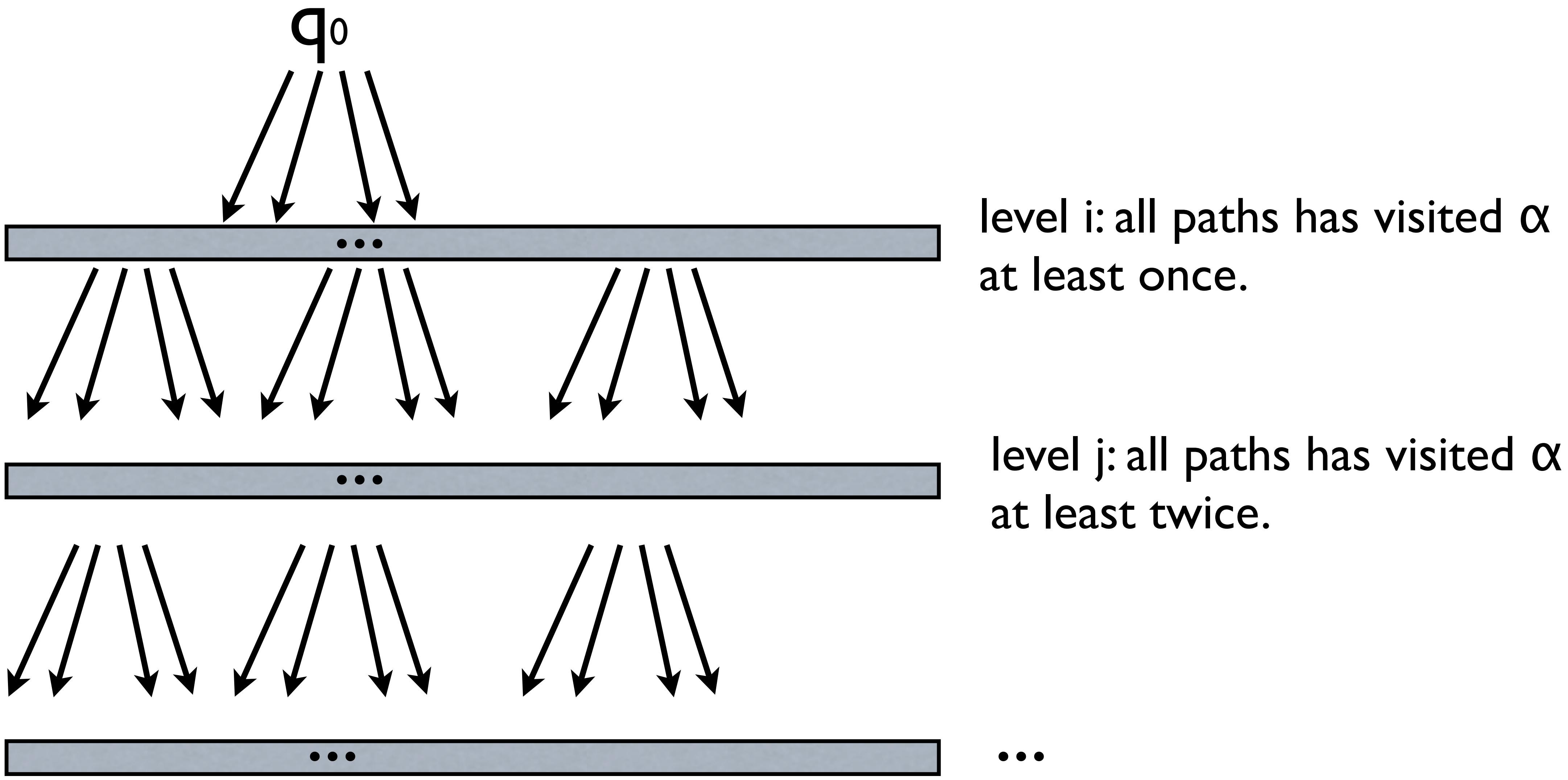
B an AcoBW that accepts the complement of A

C an ABW that accepts the same language as B

This step is conceptually simple but costs $O(2^n)$

Output: D an NBW that accepts the same language as C

Accepting runs of ABW



A NBW can guess a run by maintaining pairs (S,O):
S states of a level and O⊆S states that need a visit to α.

Miyano-Hayashi construction

- Given an ABW $C=(Q,q_0,\Sigma,\delta,\alpha)$, the NBW that accepts the same language is given by $D=(2^Q x 2^Q,(\{q_0\},\emptyset), \Sigma,\delta',\alpha')$ where:
  - for any $(S,0)\in 2^Q x 2^Q$, for any $\sigma\in\Sigma$:
    - if $O\neq\emptyset$ then $\delta'((S,O),\sigma)$ is the set of elements $\{(S',O'\backslash\alpha)\}$ s.t.

      $O'\subseteq S'$, $\forall q\in S$: $\exists T\in\delta(q, \sigma):T\subseteq S'$, and $\forall q\in O$: $\exists T\in\delta(q, \sigma):T\subseteq O'$.
    - if $O=\emptyset$ then $\delta'((S,O),\sigma)$ is the set of elements $\{(S',O'\backslash\alpha)\}$ s.t.

      $O'=S'$, $\forall q\in S$: $\exists T\in\delta(q, \sigma):T\subseteq S'$.
    - $\alpha'=2^Q x\{\emptyset\}$

Miyano-Hayashi construction

- Given an ABW $C=(Q,q_0,\Sigma,\delta,\alpha)$, the NBW that accepts the same language is given by $D=(2^Q x 2^Q,(\{q_0\},\varnothing), \Sigma,\delta',\alpha')$ where:
  - for any $(S,O)\in 2^Q x 2^Q$, for any $\sigma\in\Sigma$:

Unfortunately, this automaton is (usually) **huge** as it is constructed on the set of locations
$2^Q x 2^Q$

Miyano-Hayashi construction

- Given an ABW C=(Q
  given by D=(2^Qx2^Q,({
  - for any (S,0)∈ 2^Qx

Unfortun

(usually) **hu**

the set of locations

2^Qx2^Q

This explains the **poor** performances reported for current implementations of the construction

But, we do not need explicit complementation ...

- To check **universality** of A, we do **not** need to construct D explicitely;
- ... we only need to check if D is **empty** or not;
- ... similarly to check inclusion, i.e. $L(A) \subseteq L(B)$, we

  do **not** need to construct the complement of B but we need to check that $L(A) \cap L^c(B)$ is **empty**.

But, we do not need explicit complementation ...

- To check **universality** of A, we do **not** need to construct D explicitely;
- ... we only need to check if D is **empty** or not;

How can we check **efficiently** the **emptiness** of D ?

Emptiness of NBW

To evaluate **emptiness** of $A=(Q,q_0,\Sigma,\delta,\alpha)$

Check if

$$q_0 \in \boldsymbol{\nu y . \mu x . (\mathrm{Pre}(x) \cup (\mathrm{Pre}(y) \cap \alpha))}$$

Simulation pre-orders and fixed points

Let A= be a NBW,

$\leq \subseteq Q \times Q$ is a **simulation pre-order** iff

for any $q_1, q_2, q_3 \in Q$, for any $\sigma \in \Sigma$,

1)

$$\begin{array}{lll} & q_3 & \\ \textbf{if} & \leq & \\ & q_1 & \xrightarrow{\quad\sigma\quad} q_2 \end{array}$$

Simulation pre-orders and fixed points

Let A= be a NBW,

$\leq \subseteq Q \times Q$ is a **simulation pre-order** iff

for any $q_1, q_2, q_3 \in Q$, for any $\sigma \in \Sigma$,

then there exists $q_4 \in Q$ s.t.:

1)

$$\begin{array}{cccc} & q_3 & \xrightarrow{\quad\sigma\quad} & q_4 \\ \textbf{if} & \leq & & \leq \\ & q_1 & \xrightarrow{\quad\sigma\quad} & q_2 \end{array}$$

Simulation pre-orders and fixed points

Let A= be a NBW,

$\leq \subseteq Q \times Q$ is a **simulation pre-order** iff

for any $q_1, q_2, q_3 \in Q$, for any $\sigma \in \Sigma$,

then there exists $q_4 \in Q$ s.t.:

1) **if**

$$\begin{tikzcd} q_3 \arrow[r, "\sigma"] & q_4 \\ q_1 \arrow[u, phantom, "\leq" rotate=90] \arrow[r, "\sigma"] & q_2 \arrow[u, phantom, "\leq" rotate=90] \end{tikzcd}$$

2) and, for any $q_1, q_2 \in Q$: **if** $q_1 \leq q_2$ and $q_2 \in \alpha$ **then** $q_1 \in \alpha$

Simulation pre-orders and fixed points

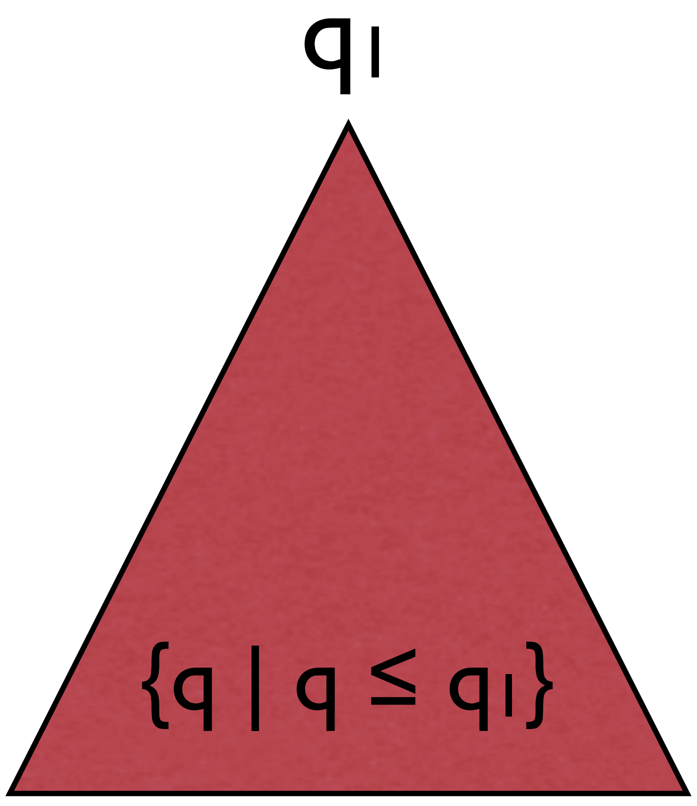
- Lemma: for any NBW $A=(Q,q_0,\Sigma,\delta,\alpha)$, for any **simulation pre-order** $\leq$, for any **$\leq$-closed** $S,T \subseteq Q$:

  (1) for all $\sigma \in \Sigma$: $Pre(\sigma)(S)$ is **$\leq$-closed**;

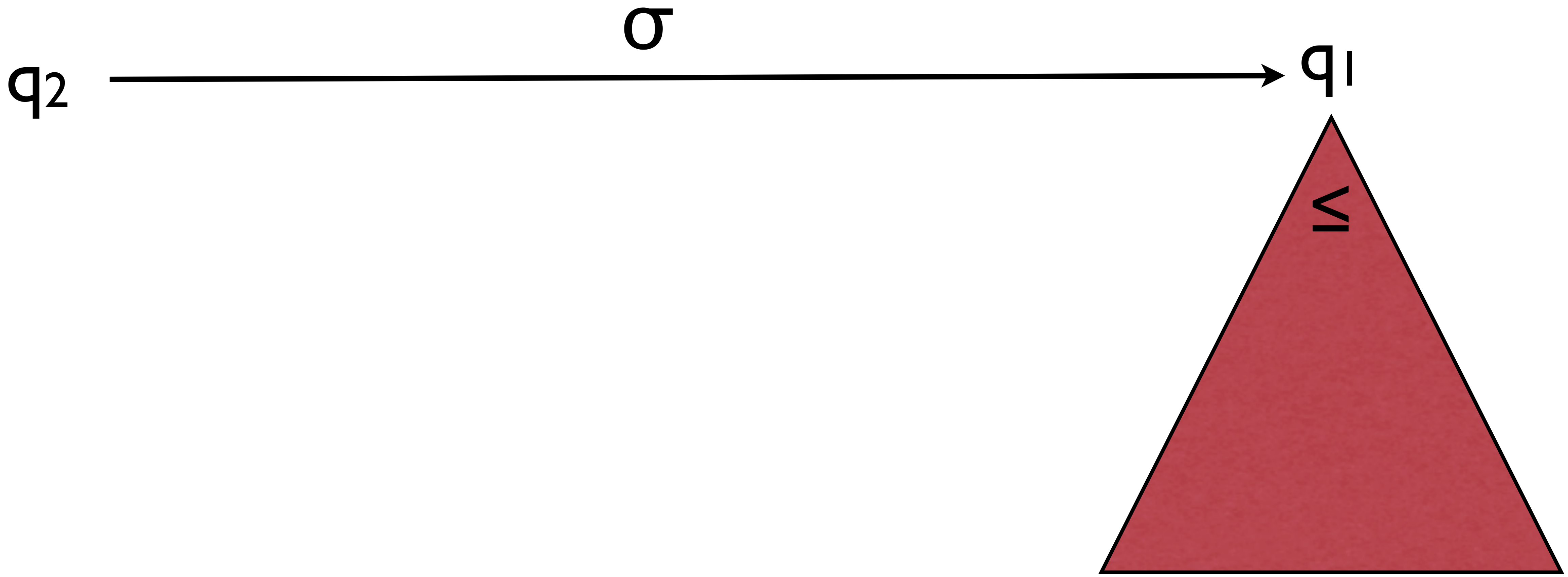
  (2) $S \cup T$ and $S \cap T$ are **$\leq$-closed**;

  (3) α is **$\leq$-closed**;

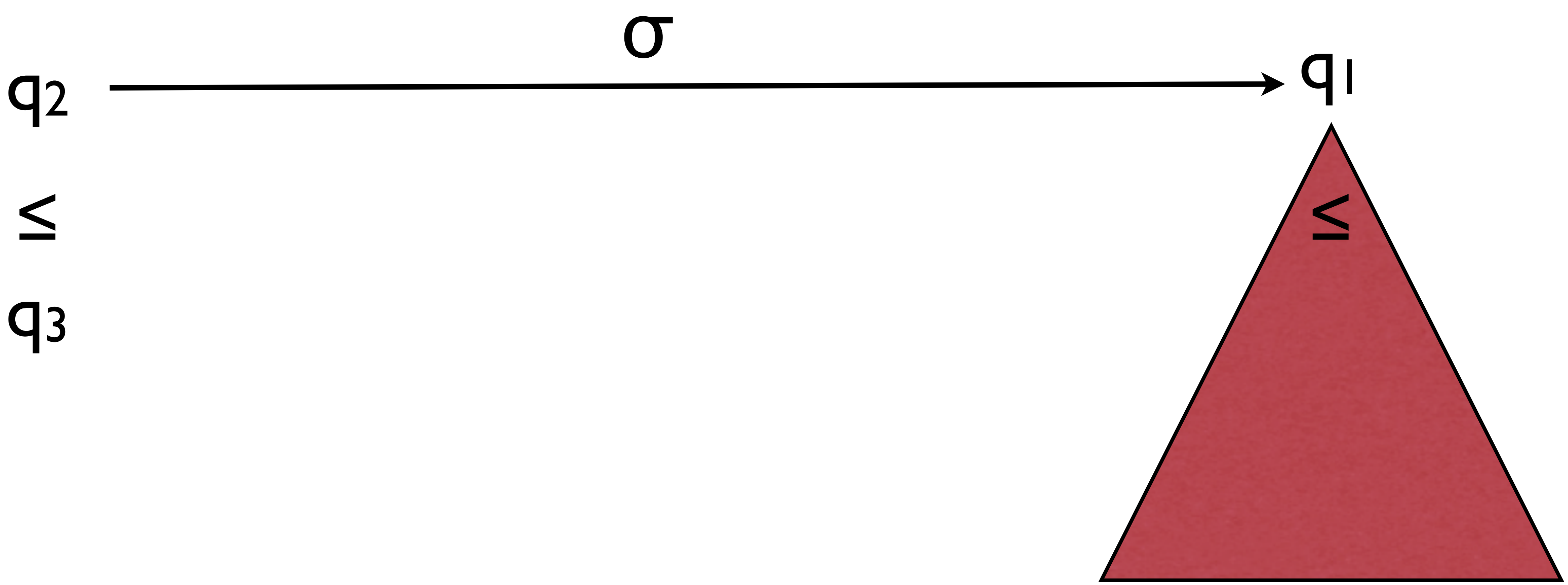
Simulation pre-orders and fixed points



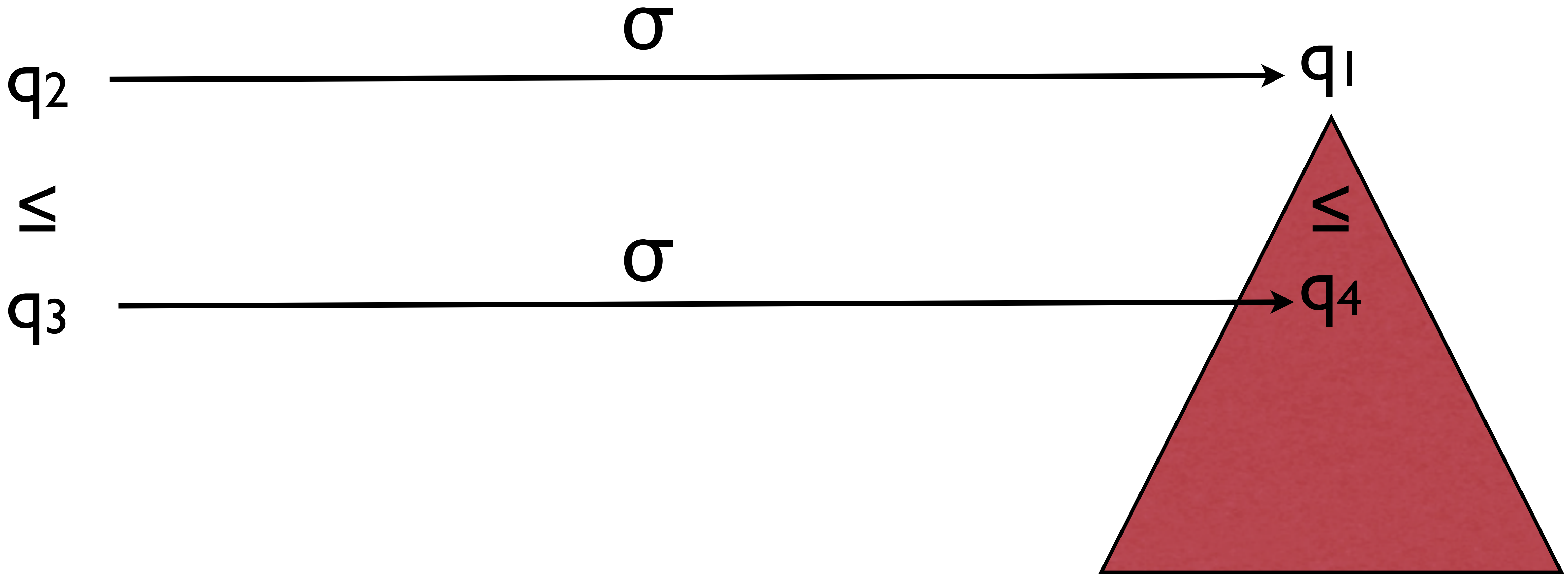
Simulation pre-orders and fixed points



Simulation pre-orders and fixed points

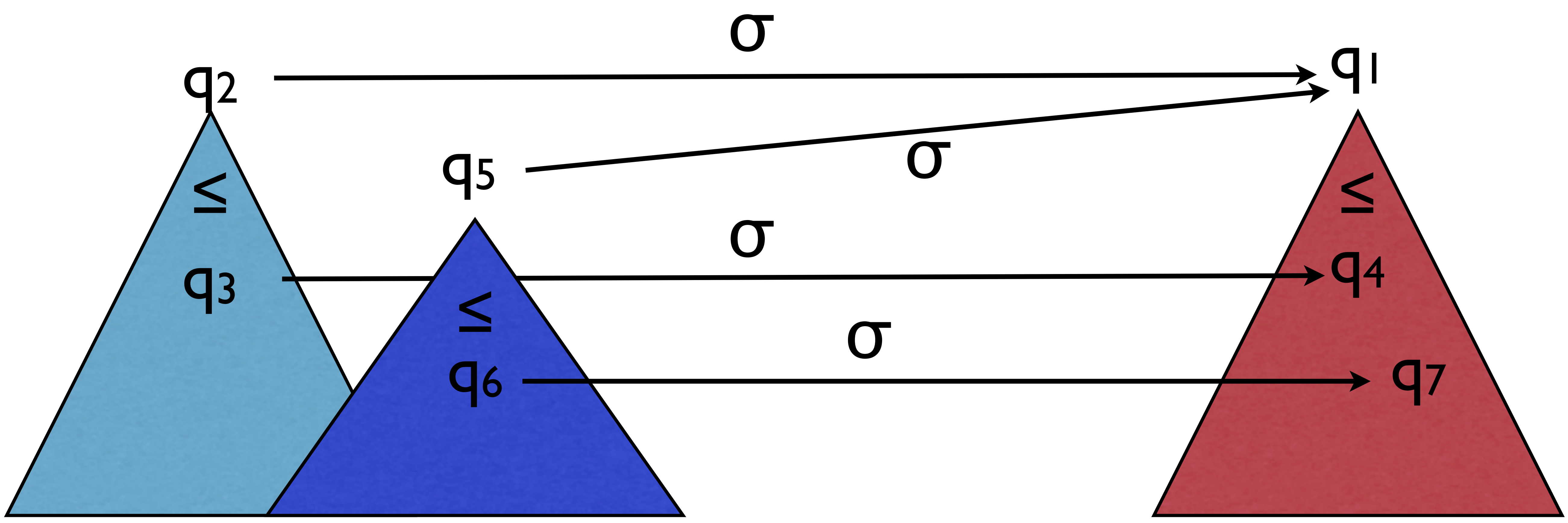


Simulation pre-orders and fixed points



Simulation pre-orders and fixed points

Simulation pre-orders and fixed points



Simulation pre-orders and fixed points

- Lemma: for any NBW $A=(Q,q_0,\Sigma,\delta,\alpha)$, for any **simulation pre-order** $\leq$, for any **$\leq$-closed** $S,T \subseteq Q$:

  (1) for all $\sigma \in \Sigma$: $Pre(\sigma)(S)$ is **$\leq$-closed**;

  ed;

So, all the sets that we manipulate in
$\nu y \,.\, \mu x \,.\, (Pre(x) \cup (Pre(y) \cap \alpha))$

are $\leq$-closed.

Simulation pre-orders and fixed points

- Lemma: for any NBW $A=(Q,q_0,\Sigma,\delta,\alpha)$, for any **simulation pre-order** $\leq$, for any **$\leq$-closed** $S,T\subseteq Q$:

(1) for

ed;

$\leq$-closed can be represented symbolically by their **maximal elements only**

So, all the set

$\nu y \,.\, \mu x \,.\, (\, Pre(x) \cup (\, Pre(y) \cap \alpha \,)\,)$

are $\leq$-closed.

Simulation pre-orders and fixed points

- Lemma: fo
  any **simul**
  ≤-closed

  (1) for

So, all the set
$\nu y . \mu x . (Pre(x) \cup (Pre(y) \cap \alpha))$

are ≤-closed.

≤-

symbolically by their
maximal elements only

We can potentially compute
$\nu y . \mu x . (Pre(x) \cup (Pre(y) \cap \alpha))$

more efficiently by working on maximal elements **only**.

Good news !

The NBW that results from the KV procedure
is equipped **by construction**
with a **simulation** pre-order ≤.

Idea: do not construct the huge NBW but
check emptiness **directly** and evaluate the
fixed point **efficiently**
by exploiting the ≤-**pre-order**.

Illustration: emptiness of ABW

- Remember that given an ABW $A=(Q,q_0,\Sigma,\delta,\alpha)$, the Miano-Hayashi construction specifies an NBW $B= (2^Q \times 2^Q,(\{q_0\},\varnothing\}),\Sigma,\delta',\alpha')$.

- The following relation $\leq \subseteq 2^Q \times 2^Q$ defined by

  $(S,O) \leq (S',O')$ iff (1) ($O=\varnothing$ iff $O'=\varnothing$) and (2) $S \subseteq S'$ and $O \subseteq O'$

  is a **simulation pre-order** on B.

- Note that the $\leq$-closure of a pair (S,O) contains an **exponential number** of elements in the size of S and O!

Illustration: emptiness of ABW

- Remember that given an ABW $A=(Q,q_0,\Sigma,\delta,\alpha)$, the Miano-Hayashi construction specifies an NBW $B=(2^Q \times 2^Q,(\{q_0\},\varnothing),\Sigma,\delta',\alpha')$.

- The following relation $\leq \subseteq 2^Q \times 2^Q$ defined by

  $(S,O) \leq (S',O')$ iff (1) ($O=\varnothing$ iff $O'=\varnothing$) and (2) $S \subseteq S'$ and $O \subseteq O'$

  is a **simulation p**

- Note that the ≤-cl
  exponential nu

We can check emptiness of B by manipulating **≤-closed sets** represented by their **maximal elements only**.

Illustration: emptiness of ABW

- Remember that given an ABW $A=(Q,q_0,\Sigma,\delta,\alpha)$, the Miano-Hayashi construction specifies an

ed by

nd (2) S⊆S' and O⊆O'

exponential nu

This potentially saves us an **exponential** !

emptiness of B by
≤-closed sets
represented by their **maximal elements only**.

Illustration: emptiness of ABW

- Remember that gi
Hayashi constructi

exponential nu

This potentiall
expone

We have a **polynomial** time algorithm that given (S,O) and $\sigma \in \Sigma$, compute a compact representation of **Pre(σ)(↓(S,O))**

emptiness of B by
≤-closed sets
represented by their **maximal elements only**.

Practical evaluation

- We have implemented our new algorithm to check universality of NBW;
- Evaluation on a **randomized model** proposed by Tabakov and Vardi (2005) that generates random NBW (two parameters: r,f);
- On that randomized model Tabakov's BDD implementation can handle **6 states** on the most difficult instances with median time **<20s**.

Practical evaluation
Universality

Input: A an NBW

Implicit — B an AcoBW that accepts the complement of A

Implicit — C an ABW that accepts the same language as B

Implicit — **Output**: D an NBW that accepts the same language as C

Practical evaluation Universality

Input: A an NBW

Implicit

We evaluate the fixed point for emptiness directly, that is, **without** constructing the automaton specified by the construction.

Implicit

We evaluate this fixed point by manipulating **≤-closed** sets through their **maximal elements only**.

Implicit

Practical evaluation Universality

Table 1. Automata size for which the median execution time for checking universality is less than 20 seconds. The symbol $\propto$ means *more than 1500*.

f \ r	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
0.1	$\propto$	$\propto$	$\propto$	550	200	120	60	40	30	40	50	50	70	90	100
0.3	$\propto$	$\propto$	$\propto$	500	200	100	40	30	40	70	100	120	160	180	200
0.5	$\propto$	$\propto$	$\propto$	500	200	120	60	60	90	120	120	120	140	260	500
0.7	$\propto$	$\propto$	$\propto$	500	200	120	70	80	100	200	440	1000	$\propto$	$\propto$	$\propto$
0.9	$\propto$	$\propto$	$\propto$	500	180	100	80	200	600	$\propto$	$\propto$	$\propto$	$\propto$	$\propto$	$\propto$

For r=2, f=0.5, Tabakov can handle **8** states while our algorithm handles **120** states in less than 20s.

Practical evaluation Universality

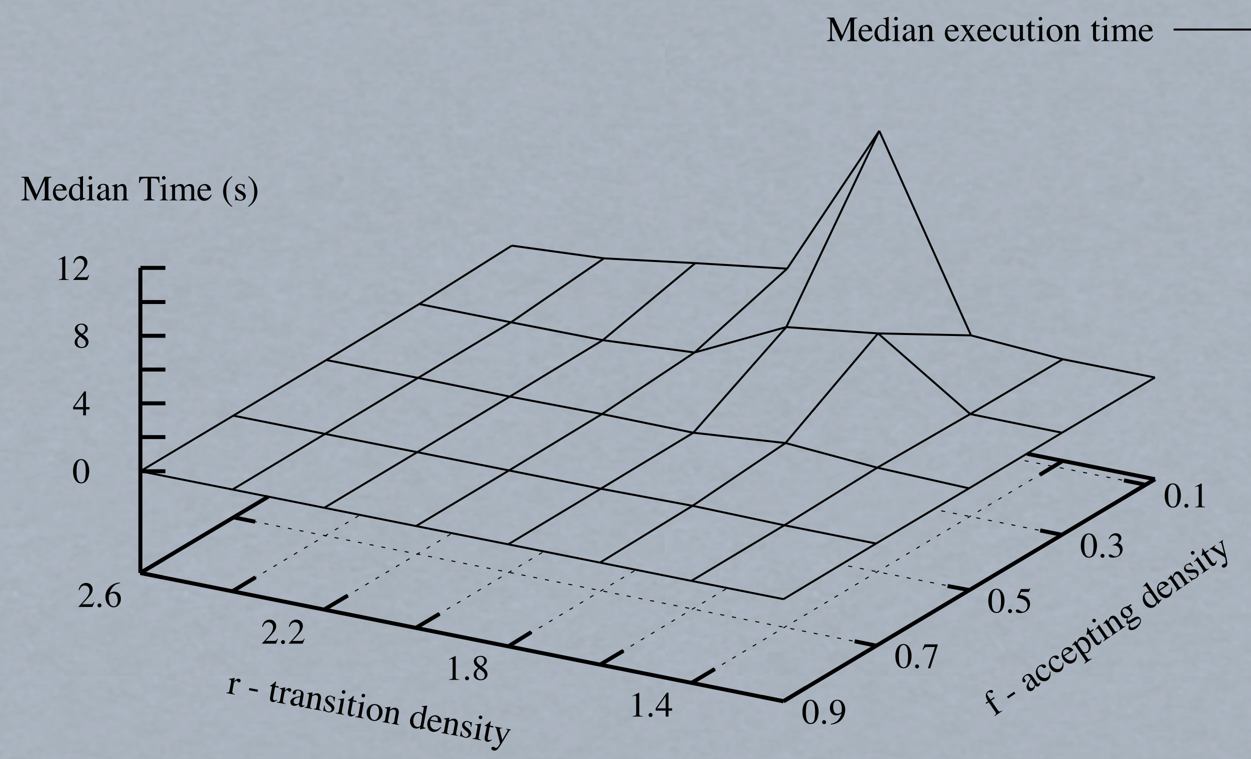


Fig. 1. Median time to check universality of 100 automata of size 30 for each sample point.

To compare, Tabakov's BDD implementation was able to handle automata of size **6** on the entire state space (within 20s as in our expermients).

Conclusions

- In the automata-based approach to model-checking: keep **implicit** the complementation step and check for emptiness efficiently by exploiting **simulation pre-orders** that exists by **construction** ;
- Implementation for **universality** problem shows promising results: **several orders of magnitude** on the randomized model !

Future Works

- Implement and evaluate the new **language inclusion** algorithm ;
- Evaluate beyond the randomized model ;
- Revisit the **LTL** model-checking problem: do not construct the **NBW** of the negation of the formula but use **ABW** and check directly for **emptiness**.