A Uniform (Bi-)Simulation Based Framework for Reducing Tree Automata

Parosh A. Abdulla, Lisa Kaati

Uppsala University, Department of Information Technology

Lukáš Holík, Tomáš Vojnar

FIT, Brno University of Technology

Tree Automata

- A bottom-up tree automaton $A = (Q, \Sigma, F, \Delta)$ where:
  - Q is a finite set of states
  - $F \subseteq Q$ is a set of final states
  - Σ a ranked alphabet with a rank function $\# : \Sigma \to \mathbb{N}$.
  - Δ is a set tree transition rules of the form as in the following example:

$$Q = \{q, r, s\}, F = \{s\}, \Sigma = \{a, b, c\}$$

$$\Delta = \{ \quad \#(a) = 0$$
$$(r, q, r) \xrightarrow{c} s \quad \#(b) = 2$$
$$(q, q) \xrightarrow{b} r \quad \#(c) = 3$$
$$\xrightarrow{a} q$$
$$\}$$

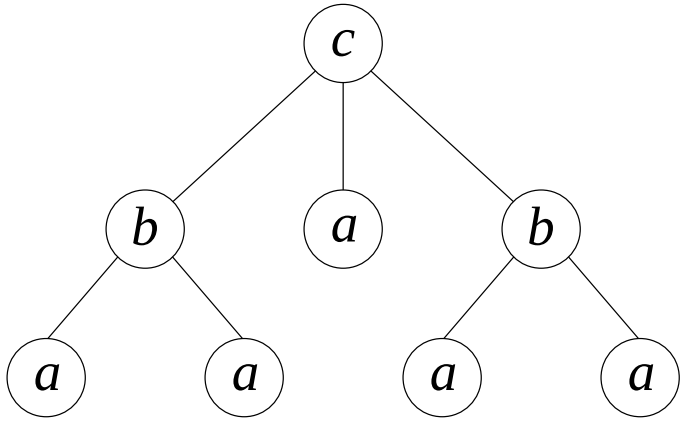
Tree Automata

- A bottom-up tree automaton $A = (Q, \Sigma, F, \Delta)$ where:
  - Q is a finite set of states
  - $F \subseteq Q$ is a set of final states
  - Σ a ranked alphabet with a rank function $\# : \Sigma \to \mathbb{N}$.
  - Δ is a set tree transition rules of the form as in the following example:

$$Q = \{q, r, s\}, F = \{s\}, \Sigma = \{a, b, c\}$$

$$\Delta = \{ \quad (r, q, r) \xrightarrow{c} s \quad (q, q) \xrightarrow{b} r \quad \xrightarrow{a} q \quad \}$$

$$\#(a) = 0 \quad \#(b) = 2 \quad \#(c) = 3$$



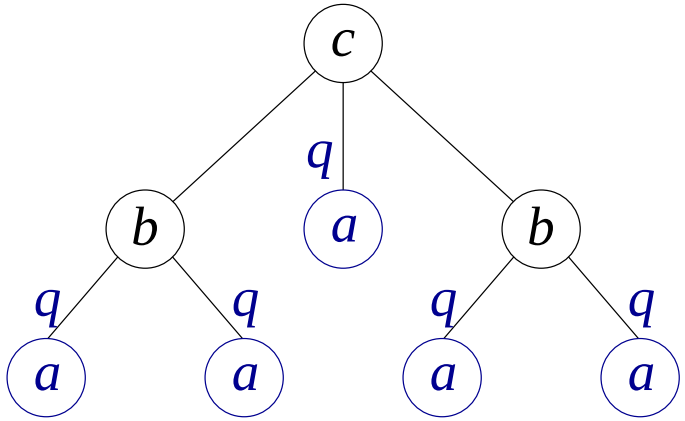
Tree Automata

- A bottom-up tree automaton $A = (Q, \Sigma, F, \Delta)$ where:
  - Q is a finite set of states
  - $F \subseteq Q$ is a set of final states
  - Σ a ranked alphabet with a rank function $\# : \Sigma \to \mathbb{N}$.
  - Δ is a set tree transition rules of the form as in the following example:

$$Q = \{q, r, s\}, F = \{s\}, \Sigma = \{a, b, c\}$$

$$\Delta = \{ \quad (r, q, r) \xrightarrow{c} s \quad (q, q) \xrightarrow{b} r \quad \xrightarrow{a} q \quad \}$$

$$\#(a) = 0 \quad \#(b) = 2 \quad \#(c) = 3$$



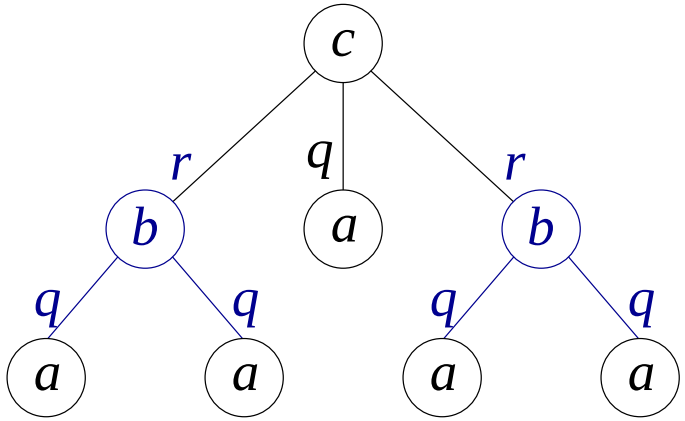
Tree Automata

- A bottom-up tree automaton $A = (Q, \Sigma, F, \Delta)$ where:
  - Q is a finite set of states
  - $F \subseteq Q$ is a set of final states
  - Σ a ranked alphabet with a rank function $\# : \Sigma \to \mathbb{N}$.
  - Δ is a set tree transition rules of the form as in the following example:

$$Q = \{q, r, s\}, F = \{s\}, \Sigma = \{a, b, c\}$$

$$\Delta = \{ \quad (r, q, r) \xrightarrow{c} s \quad (q, q) \xrightarrow{b} r \quad \xrightarrow{a} q \quad \}$$

$$\#(a) = 0 \quad \#(b) = 2 \quad \#(c) = 3$$



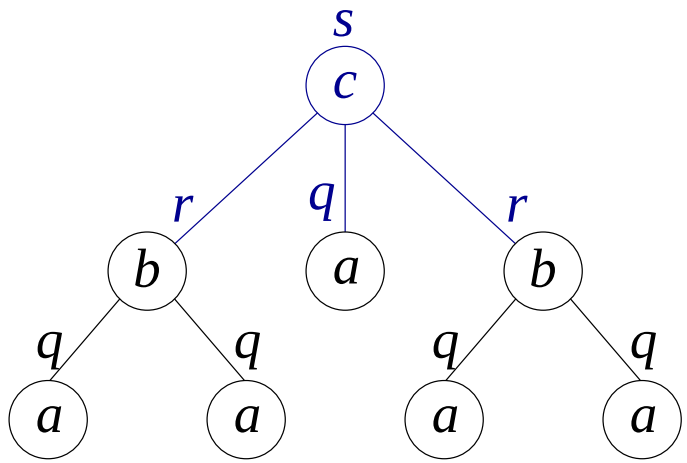
Tree Automata

- A bottom-up tree automaton $A = (Q, \Sigma, F, \Delta)$ where:
  - Q is a finite set of states
  - $F \subseteq Q$ is a set of final states
  - Σ a ranked alphabet with a rank function $\# : \Sigma \to \mathbb{N}$.
  - Δ is a set tree transition rules of the form as in the following example:

$$Q = \{q, r, s\}, F = \{s\}, \Sigma = \{a, b, c\}$$

$$\Delta = \{ \quad (r, q, r) \xrightarrow{c} s \quad (q, q) \xrightarrow{b} r \quad \xrightarrow{a} q \quad \}$$

$$\#(a) = 0 \quad \#(b) = 2 \quad \#(c) = 3$$



Size Reduction of NTA – Motivation

- Tree Automata are used in various domains like:
  - Structured documents (XML)
  - Logics on trees (MSO)
  - Natural language processing
  - Regular Tree Model Checking:
    - verification of systems with unbounded state space
    - sets of states (potentially infinite) represented as tree automata

- A substantial problem is size reduction

Size Reduction of NTA

- Minimal deterministic (canonic) automaton
  - requires determinization
  - can be still bigger that the original nondeterministic one

- Traditional word automata (bi-)simulation based methods:
  - Reduction of an automaton = merging of (bi-)simulation equivalent states
  - A Trade-off: bisimulations are stronger then simulations but easier to compute ($\mathcal{O}(m \log n) \times \mathcal{O}(mn)$ in the word case)

This Paper

❖ We consider simulations and bisimulations over bottom-up tree automata.

❖ This work bulds up on, extends, and unifies the following works

- **Computing Simulations over Tree Automata**: Efficient Techniques for Reducing Tree Automata. P.A. Abdulla, A. Bouajjani, L. Holík, L. Kaati, and T. Vojnar. (TACAS'08).
- **Composed Bisimulations for Tree Automata**. P.A. Abdulla, A. Bouajjani, L. Holík, L. Kaati, and T. Vojnar. (CIAA'08).
- **Backward and Forward Bisimulation Minimisation of Tree Automata**. J. Högberg, A. Maletti, and J. May. (CIAA'07)

❖ We consider four basic types of relations: tree automata downward and upward simulation, and downward and upward bisimulation.

Downward (Bi-)Simulation

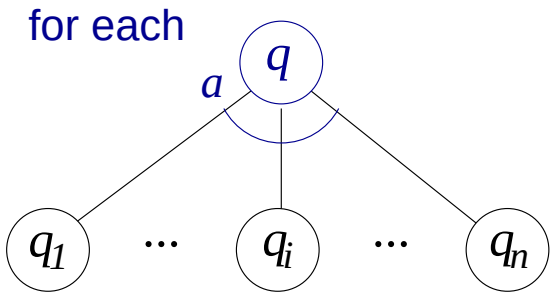
A preorder $D \subseteq Q \times Q$ is a downward simulation if qDr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r$, where $q_i D r_i$ for all $1 \leq i \leq n$.

Downward (Bi-)Simulation

A preorder $D \subseteq Q \times Q$ is a downward simulation if qDr implies that:

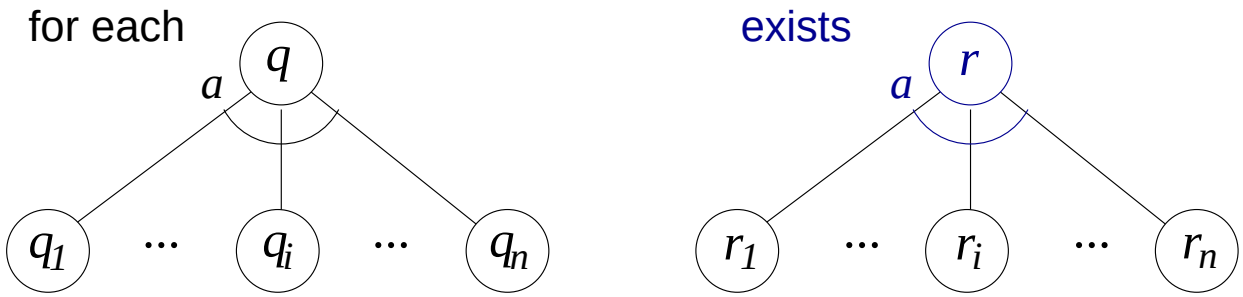
- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r$, where $q_i D r_i$ for all $1 \leq i \leq n$.



Downward (Bi-)Simulation

A preorder $D \subseteq Q \times Q$ is a downward simulation if qDr implies that:

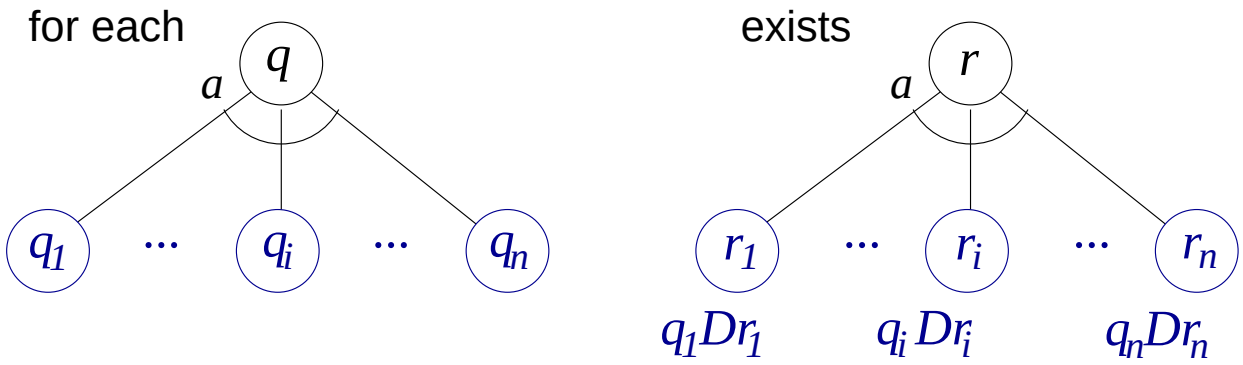
- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r$, where $q_i D r_i$ for all $1 \leq i \leq n$.



Downward (Bi-)Simulation

A preorder $D \subseteq Q \times Q$ is a downward simulation if qDr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r$, where $q_i D r_i$ for all $1 \leq i \leq n$.

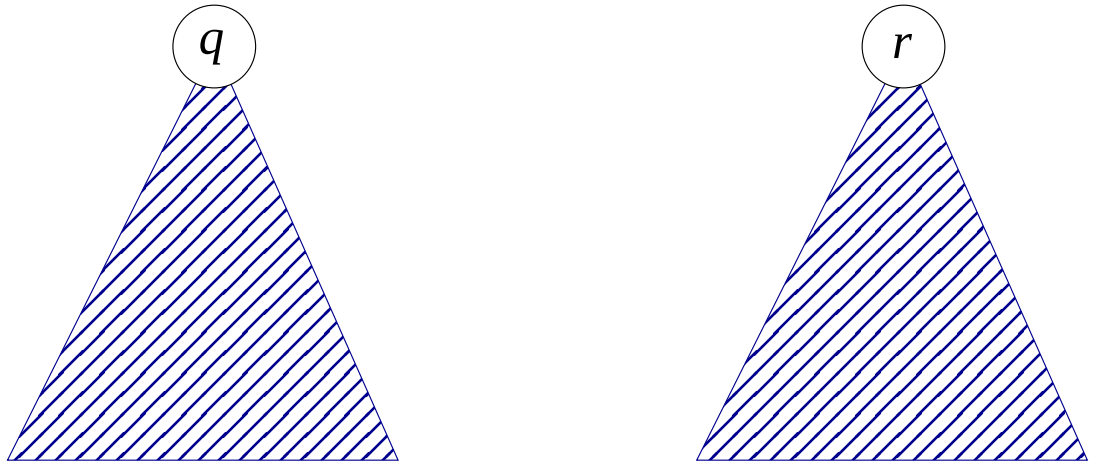


Downward (Bi-)Simulation

A preorder $D \subseteq Q \times Q$ is a downward simulation if qDr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r$, where $q_i D r_i$ for all $1 \leq i \leq n$.

If q accepts a tree then r accepts the same tree too.



Downward (Bi-)Simulation

An equivalence $D \subseteq Q \times Q$ is a downward bisimulation if qDr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r$, where $q_i D r_i$ for all $1 \leq i \leq n$,

and, conversely,

- for each rule $(r_1, \ldots, r_n) \xrightarrow{a} r$ exists a rule $(q_1, \ldots, q_n) \xrightarrow{a} q$, where $r_i D q_i$ for all $1 \leq i \leq n$.

Induced Upward (Bi-)Simulation

Let D be a downward simulation.

A preorder $U \subseteq Q \times Q$ is an upward simulation induced by D if qUr implies that:

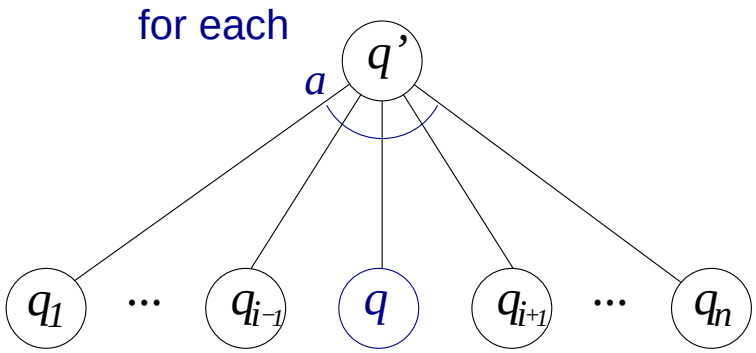
- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$ with $q_i = q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$, $r_i = r$, where $q'Ur'$, and $q_j D r_j$ for all $1 \leq i \neq j \leq n$.
- Moreover, $q \in F \iff r \in F$.

Induced Upward (Bi-)Simulation

Let D be a downward simulation.

A preorder $U \subseteq Q \times Q$ is an upward simulation induced by D if qUr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$ with $q_i = q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$, $r_i = r$, where $q'Ur'$, and $q_j D r_j$ for all $1 \leq i \neq j \leq n$.
- Moreover, $q \in F \iff r \in F$.

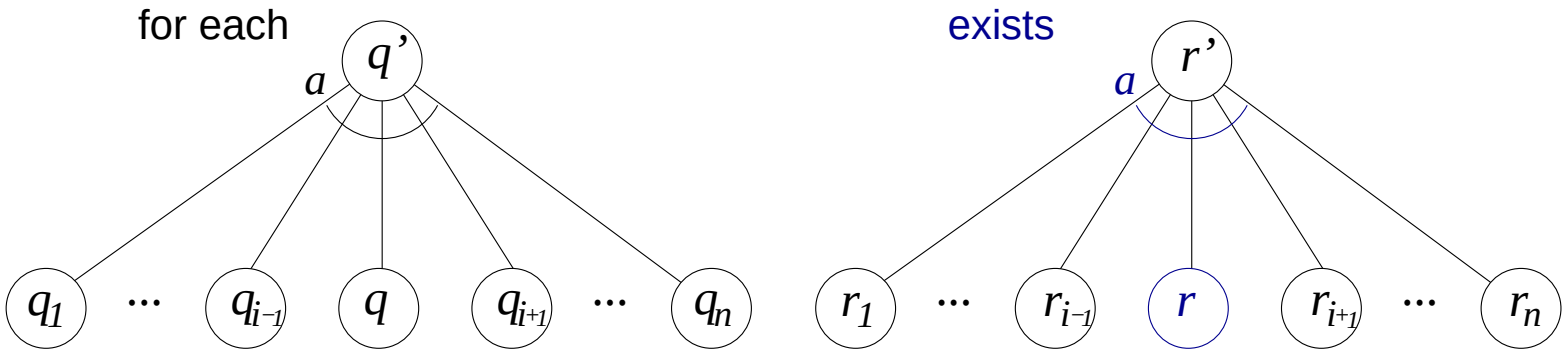


Induced Upward (Bi-)Simulation

Let D be a downward simulation.

A preorder $U \subseteq Q \times Q$ is an upward simulation induced by D if qUr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$ with $q_i = q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$, $r_i = r$, where $q'Ur'$, and $q_j D r_j$ for all $1 \leq i \neq j \leq n$.
- Moreover, $q \in F \iff r \in F$.

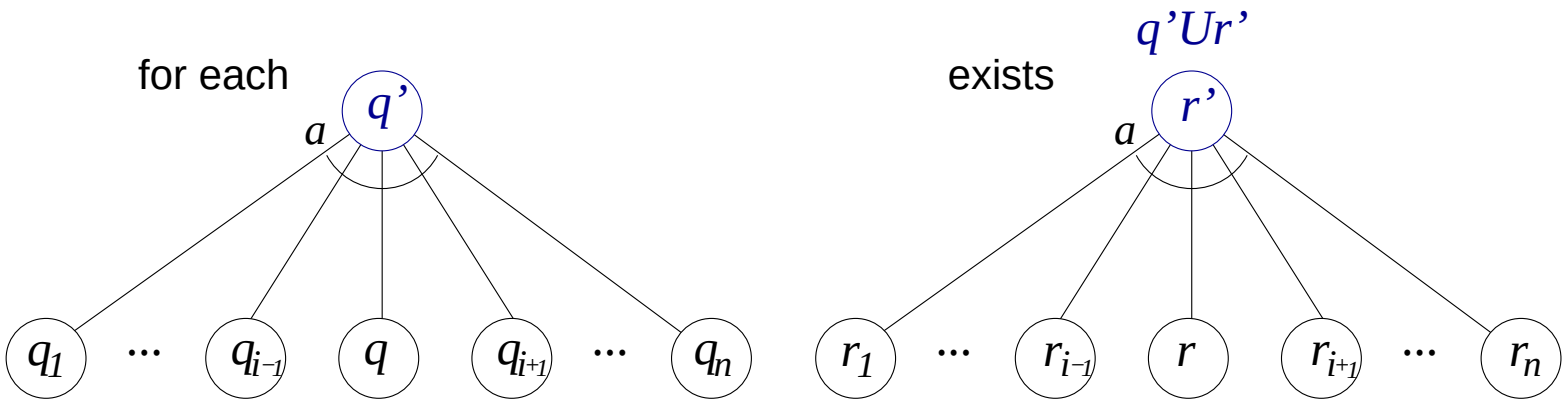


Induced Upward (Bi-)Simulation

Let D be a downward simulation.

A preorder $U \subseteq Q \times Q$ is an upward simulation induced by D if qUr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$ with $q_i = q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$, $r_i = r$, where $q'Ur'$, and $q_j D r_j$ for all $1 \leq i \neq j \leq n$.
- Moreover, $q \in F \iff r \in F$.

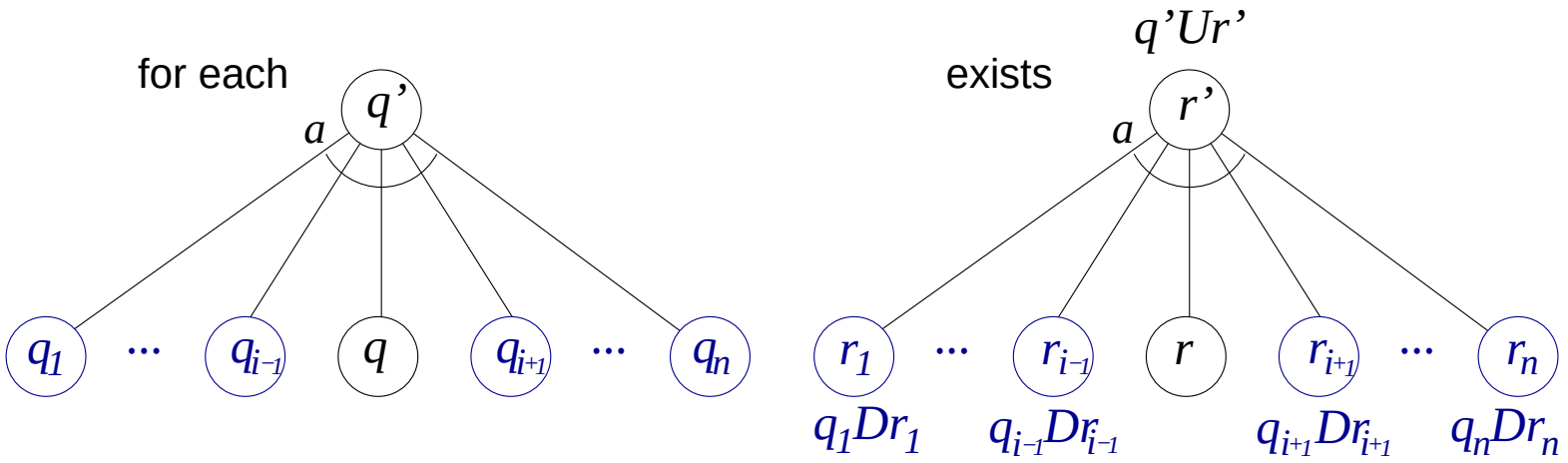


Induced Upward (Bi-)Simulation

Let D be a downward simulation.

A preorder $U \subseteq Q \times Q$ is an upward simulation induced by D if qUr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$ with $q_i = q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$, $r_i = r$, where $q'Ur'$, and $q_j D r_j$ for all $1 \leq i \neq j \leq n$.
- Moreover, $q \in F \iff r \in F$.



Induced Upward (Bi-)Simulation

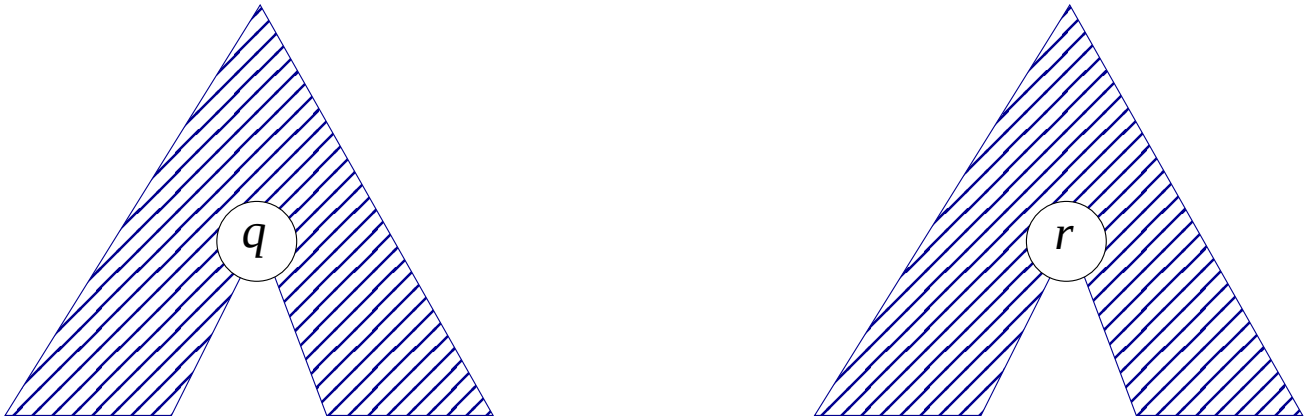
Let D be a downward simulation.

A preorder $U \subseteq Q \times Q$ is an upward simulation induced by D if qUr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$ with $q_i = q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$, $r_i = r$, where $q'Ur'$, and $q_j D r_j$ for all $1 \leq i \neq j \leq n$.
- Moreover, $q \in F \iff r \in F$.

If q appears in a context, then r appears in the same context too.

(Context of a state is that part of a tree that can be accepted "from" the state.)



Induced Upward (Bi-)Simulation

Let D be a downward simulation and let $\equiv_D = D \cap D^{-1}$ be the corresponding simulation equivalence.

An equivalence $U \subseteq Q \times Q$ is an upward bisimulation induced by D if qUr implies that:

- for each rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$ with $q_i = q$ exists a rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$, $r_i = r$, where $q'Ur'$, and $q_j \equiv_D r_j$ for all $1 \leq i \neq j \leq n$,

and, conversely,

- for each rule $(r_1, \ldots, r_n) \xrightarrow{a} r'$ with $r_i = r$ exists a rule $(q_1, \ldots, q_n) \xrightarrow{a} q'$, $q_i = q$, where $r'Uq'$, and $r_j \equiv_D q_j$ for all $1 \leq i \neq j \leq n$.
- Moreover, $q \in F \iff r \in F$.

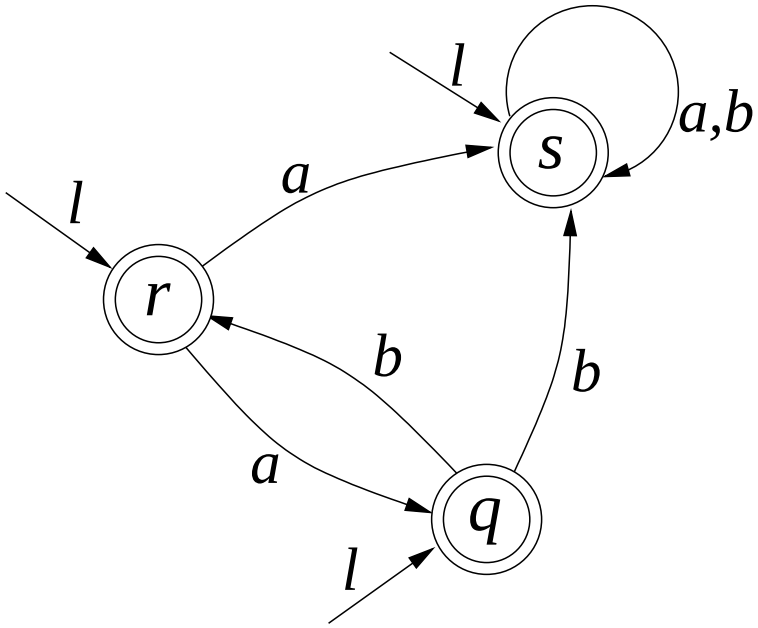
Basic Properties of the (Bi-)Simulations

- For a given type of relation, there always exists a unique maximal relation of the type.
- A bisimulation of a given type is always also a simulation of the type.
- Bisimulations are stronger but can be computed significantly faster than simulations.
  - The difference very roughly is $|\Delta|^2$ contra $|\Delta| \log |Q|$.
- In the following,
  - let DS be the maximal downward simulation and
  - let US be the maximal upward simulation induced by DS.

Reducing Automata by Combination of Upward and Downward Simulation

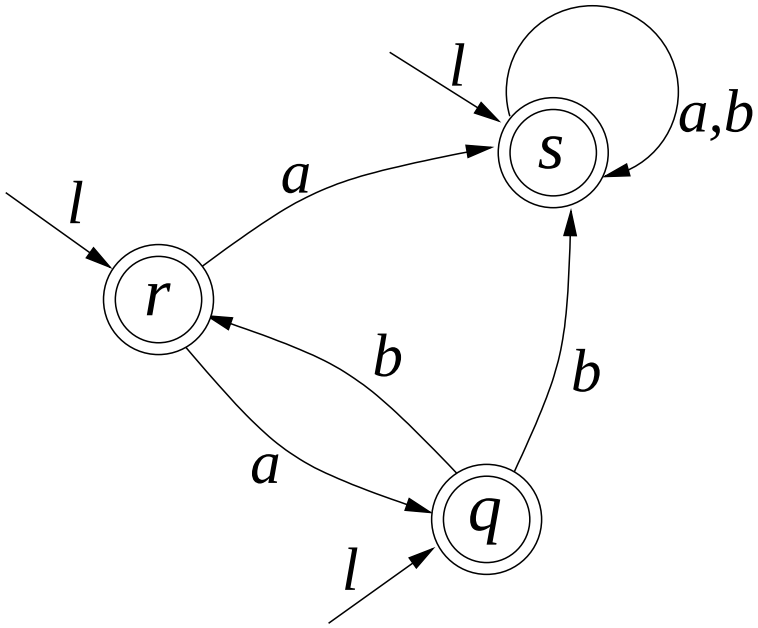
Combination of Upward and Downward

An example

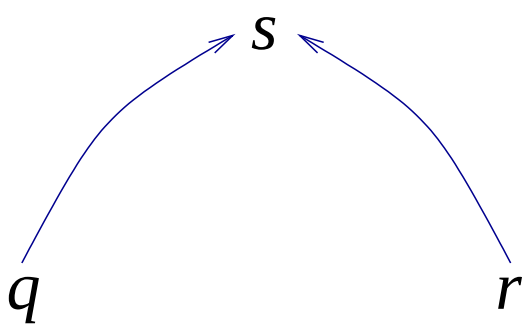


Combination of Upward and Downward

An example

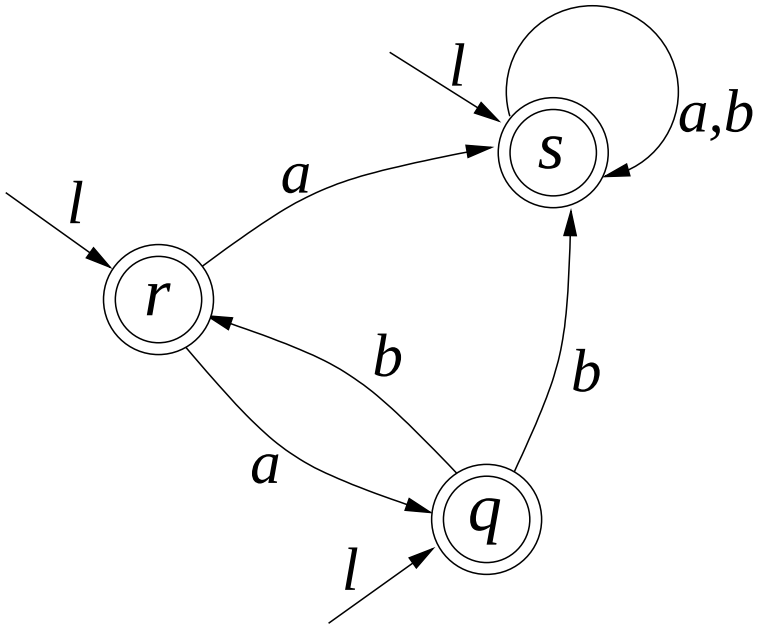


$DS = id \cup \{(q,s),(r,s)\}$



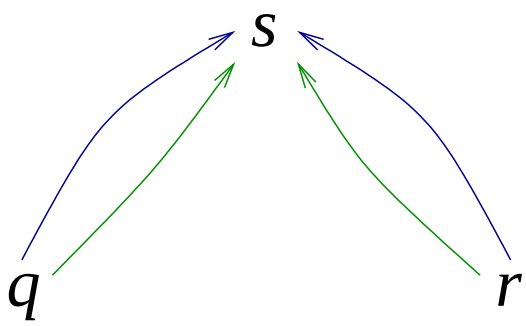
Combination of Upward and Downward

An example



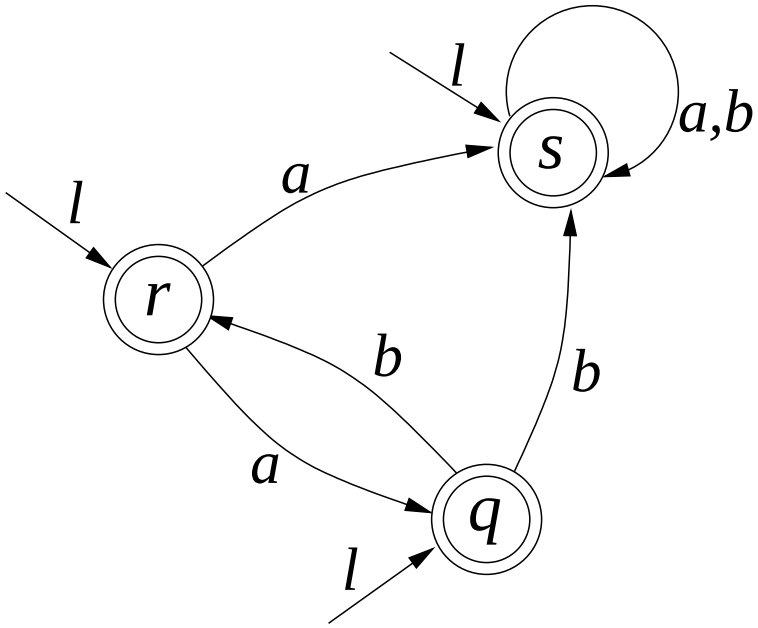
$DS = id \cup \{(q,s),(r,s)\}$

$US = id \cup \{(q,s),(r,s)\}$



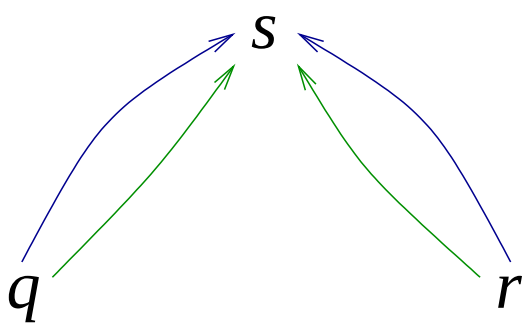
Combination of Upward and Downward

An example



$DS = id \cup \{(q,s),(r,s)\}$

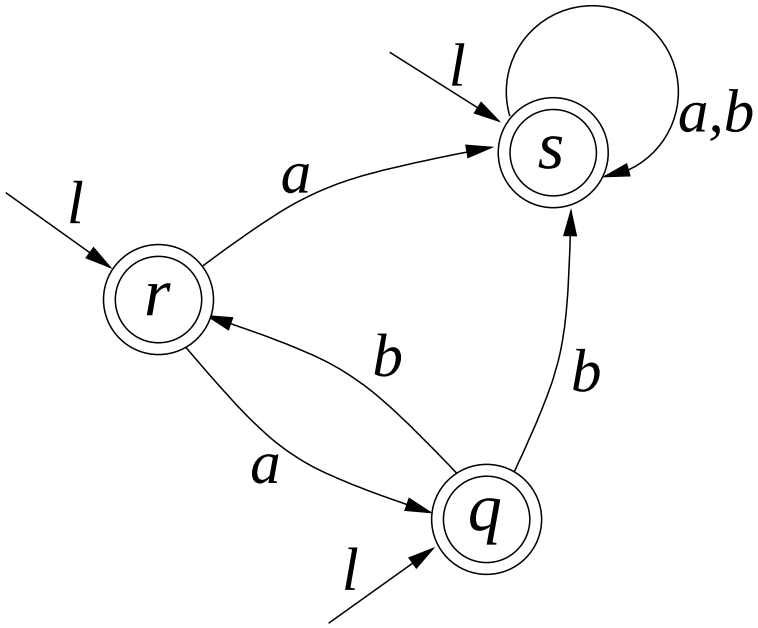
$US = id \cup \{(q,s),(r,s)\}$



- A relation where all q, r, s are equivalent?

Combination of Upward and Downward

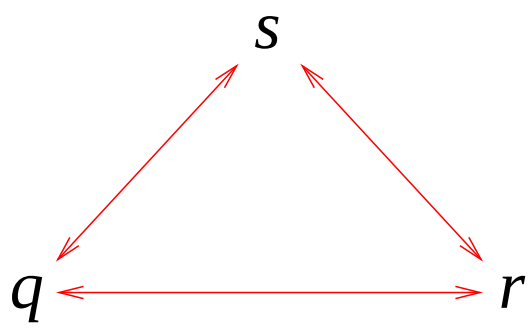
An example



$DS = id \cup \{(q,s),(r,s)\}$

$US = id \cup \{(q,s),(r,s)\}$

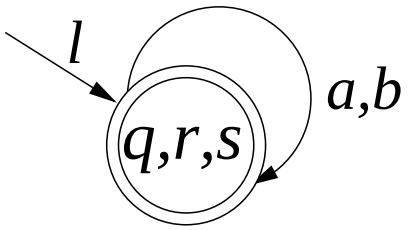
$DS \circ US^{-1} = \{q,r,s\}^2$



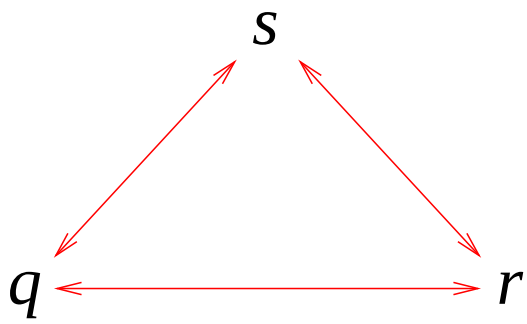
- A relation where all q, r, s are equivalent? The first guess: $C = DS \circ US^{-1}$

Combination of Upward and Downward

An example



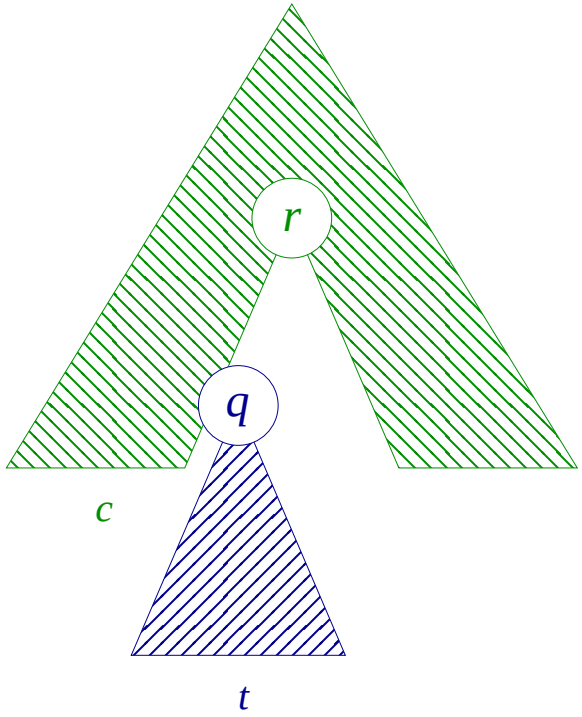
$DS = id \cup \{(q,s),(r,s)\}$
$US = id \cup \{(q,s),(r,s)\}$
$DS \circ US^{-1} = \{q,r,s\}^2$



- A relation where all q, r, s are equivalent? The first guess: $C = DS \circ US^{-1}$

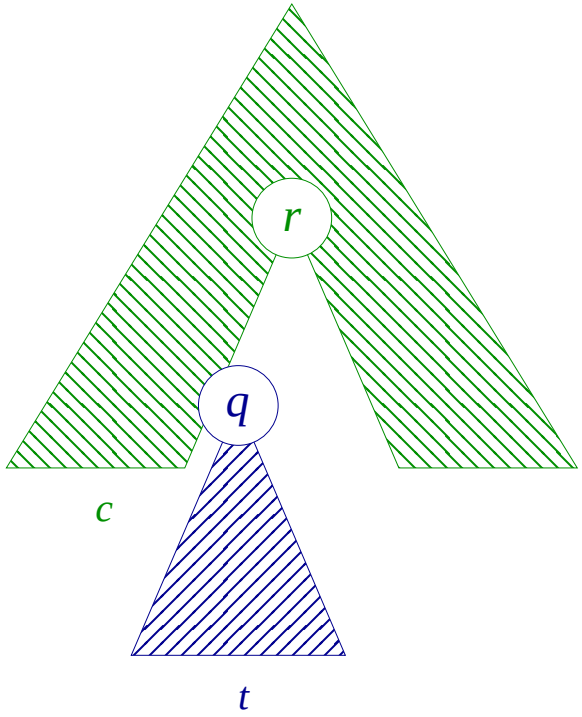
Why does composing simulations make sense?

Consider relation $C = DS \circ US^{-1}$. Let qCr for two states q and r such that q accepts a tree t and r appears in a contexts c.

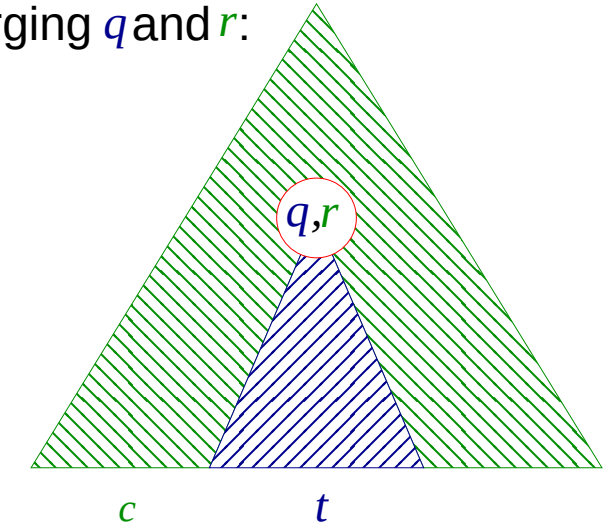


Why does composing simulations make sense?

Consider relation $C = DS \circ US^{-1}$. Let qCr for two states q and r such that q accepts a tree t and r appears in a contexts c.

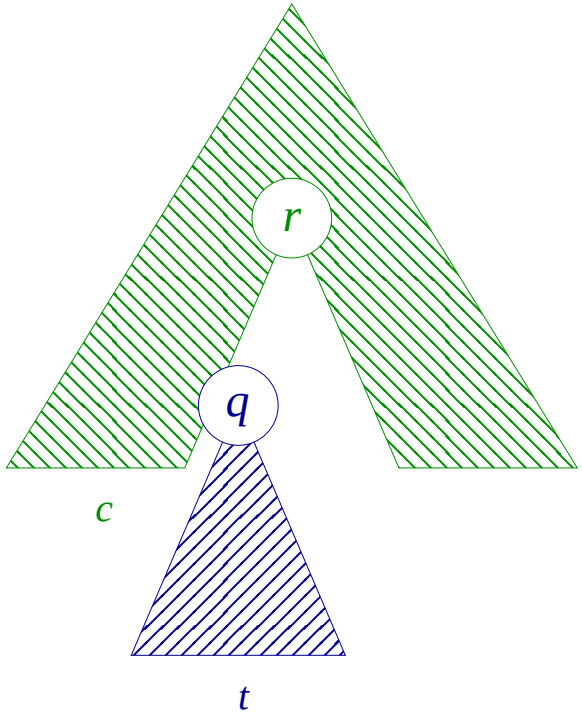


After merging q and r:

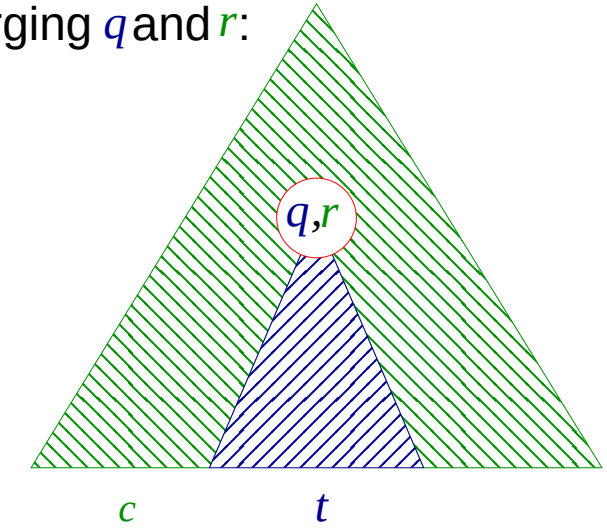


Why does composing simulations make sense?

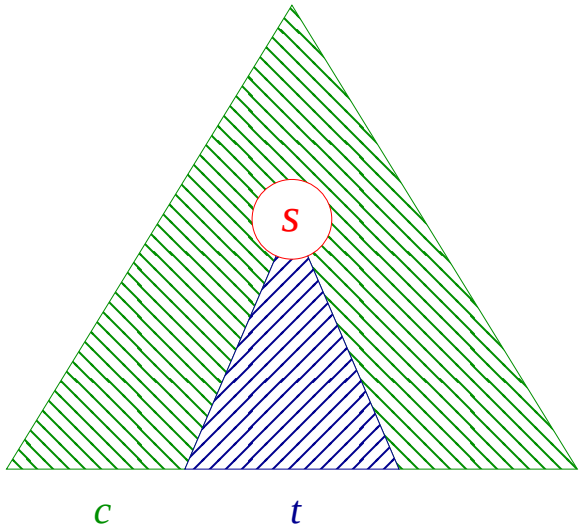
Consider relation $C = DS \circ US^{-1}$. Let qCr for two states q and r such that q accepts a tree t and r appears in a contexts c.



After merging q and r:



$$qCr \Rightarrow \exists s : qDSs \wedge rUSs$$



The Combined Simulation

- Merging states according to $D \circ U^{-1}$ is not always possible, but it still works if we take certain fragment of it,
- namely, we take the unique maximal transitive fragment F of $D \circ U^{-1}$ that contains D, and we merge automata according the corresponding equivalence $F \cup F^{-1}$, called combined equivalence $D \oplus U$.
- **Theorem**: Merging states according to the combined equivalence $D \oplus U$ preserves language.

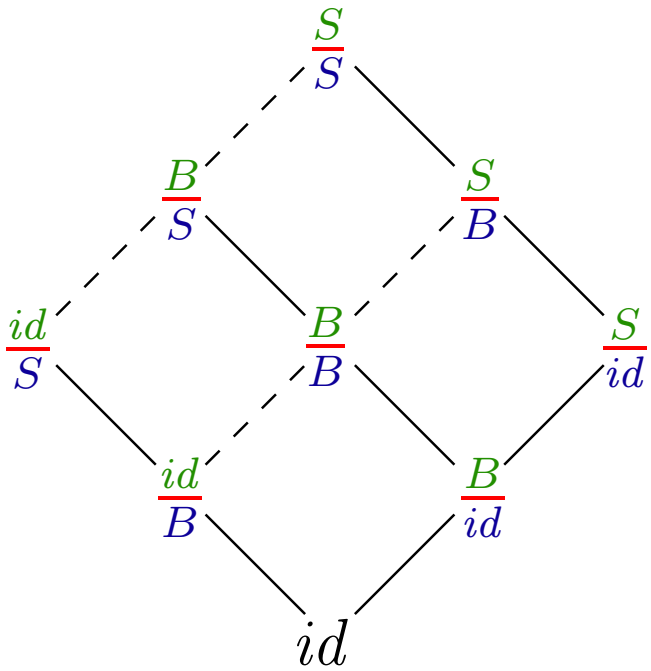
Combining Simulations and Bisimulations

Variants of the Combined Equivalence

❖ Let B stand for **bisimulation** and S for **simulation**. We use $\frac{Up}{Down}$, where $Up, Down \in \{id, B, S\}$, to denote the combined equivalence $D \oplus U$, where D is the maximal downward relation of type $Down$ and U is the maximal upward relation of type Up induced by D.

❖ We have obtained a hierarchy of combined preoreders (equivalences)

- Coarser (more reducing) relations are above.



Experiments with the Relations

❖ We experimented with tree automata obtained from runs of our RTMC tool (verification of certain communication protocols). We monitored computation time and reduction of the number of states and rules: The trade-off between computational cost and reduction capabilities is visible.

TA		$\frac{id}{S}$		$\frac{S}{id}$		$\frac{S}{B}$		$\frac{S}{S}$	
origin	size	red.	time	red.	time	red.	time	red.	time
ARTMC	195	18%	0.5 s	2%	0.5 s	23%	0.5 s	61%	1.0 s
RTMC	613	27%	3.5 s	19%	2.0 s	19 %	2.5 s	88%	5.1 s
RTMC	909	52%	3.6 s	72%	3.1 s	82%	3.4 s	89%	35.1 s
ARTMC	2029	10%	27.0 s	37%	26.0 s	33%	29.0 s	93%	39.0 s
RTMC	2403	26%	31.0 s	0%	25.0 s	0%	34.0 s	82%	37.1 s

TA		$\frac{id}{B}$		$\frac{B}{id}$		$\frac{B}{B}$		$\frac{B}{S}$	
origin	size	red.	time	red.	time	red.	time	red.	time
ARTMC	195	18%	0.1 s	2%	0.5 s	23%	0.2 s	23%	0.6 s
RTMC	613	0%	0.3 s	0%	0.4 s	0%	0.8 s	27%	3.7 s
RTMC	909	14%	0.6 s	72%	0.4 s	82%	0.8 s	83%	4.1 s
ARTMC	2029	10%	1.7 s	14%	1.4 s	19%	3.1 s	44%	29.0 s
RTMC	2403	0%	0.3 s	0%	0.6 s	0%	0.7 s	27%	31.0 s

Conclusion

- In this work:
  - We have improved the way of combining tree automata (bi-)simulations them,
  - and proposed a parametric framework that explains in a uniform way several previous results and allows a fine choice between computational cost and reduction capabilities.

- Future directions:
  - Investigating more the combination principle (even for word automata), for instance, a possibility of defining an infinite hierarchy of relations such that the inducing relation is the result of a combination in the previous step.
  - Applying the principles on more advanced types of automata such as hedge automata.

Thank you.