Automata for Boundedness Problems on Trees

Christof Löding

RWTH Aachen

Joint work (in progress) with
Thomas Colcombet

CNRS/LIAFA, Paris

Workshop on Distance Automata
November 2009, Paris

1. How it started: infinite trees and the parity index problem

2. Basics on cost functions

3. Automata on finite trees
   - Decision problems
   - Transformations

4. Conclusion

Parity index problem: background

Analyze or decide complexity of languages according to certain parameters in automata theory and logic:

- number of states/transitions (minimization)
- quantifier alternations in logical definitions
- nesting-depth of operations such as Kleene-star in regular expressions
- ...

Goal: Better understanding and optimization of these parameters

Parity index problem: background

Analyze or decide complexity of languages according to certain parameters in automata theory and logic:

- number of states/transitions (minimization)
- quantifier alternations in logical definitions
- nesting-depth of operations such as Kleene-star in regular expressions
- ...

Goal: Better understanding and optimization of these parameters

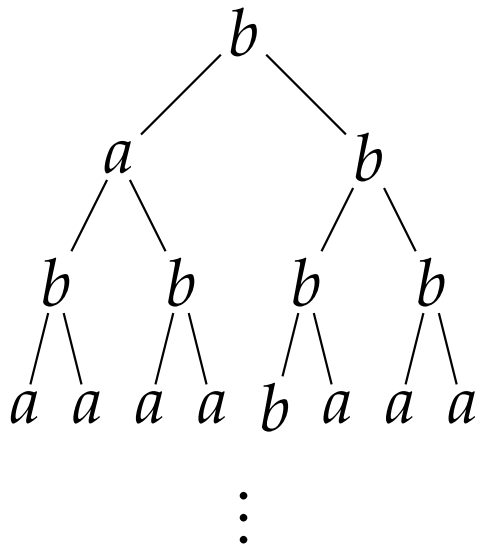
Parity Index Problem:

- Languages of infinite trees and parity tree automata
- Parameter to optimize: the number of priorities

Parity Tree Automata

- Automata $\mathcal{A} = (Q, \Sigma, q_0, \Delta, pri)$ on infinite binary trees
- Transitions of the form (q, a, q', q'')

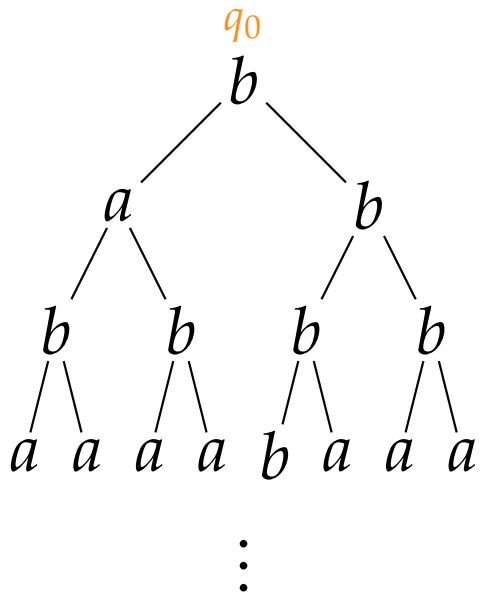
Run of $\mathcal{A}$:



Parity Tree Automata

- Automata $\mathcal{A} = (Q, \Sigma, q_0, \Delta, pri)$ on infinite binary trees
- Transitions of the form (q, a, q', q'')

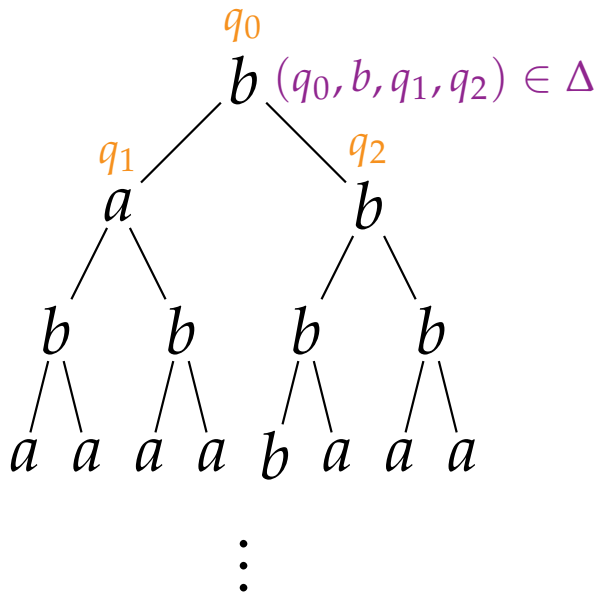
Run of $\mathcal{A}$:



Parity Tree Automata

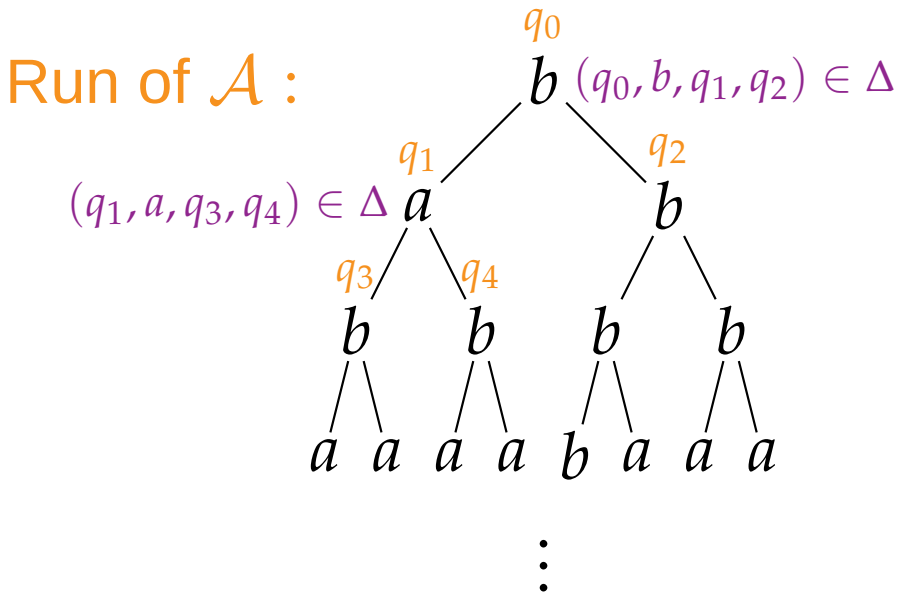
- Automata $\mathcal{A} = (Q, \Sigma, q_0, \Delta, pri)$ on infinite binary trees
- Transitions of the form (q, a, q', q'')

Run of $\mathcal{A}$:



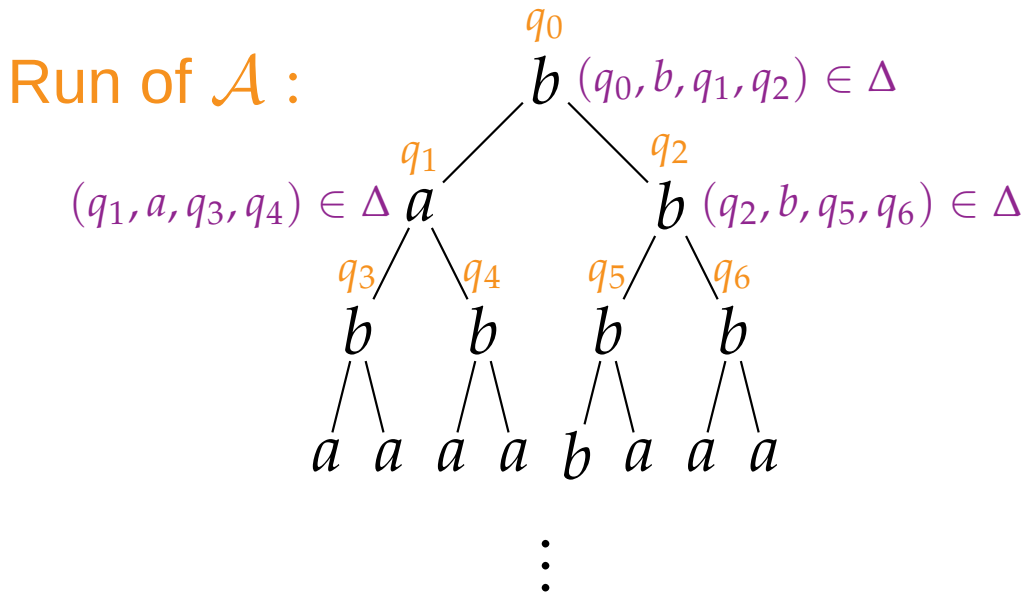
Parity Tree Automata

- Automata $\mathcal{A} = (Q, \Sigma, q_0, \Delta, pri)$ on infinite binary trees
- Transitions of the form (q, a, q', q'')



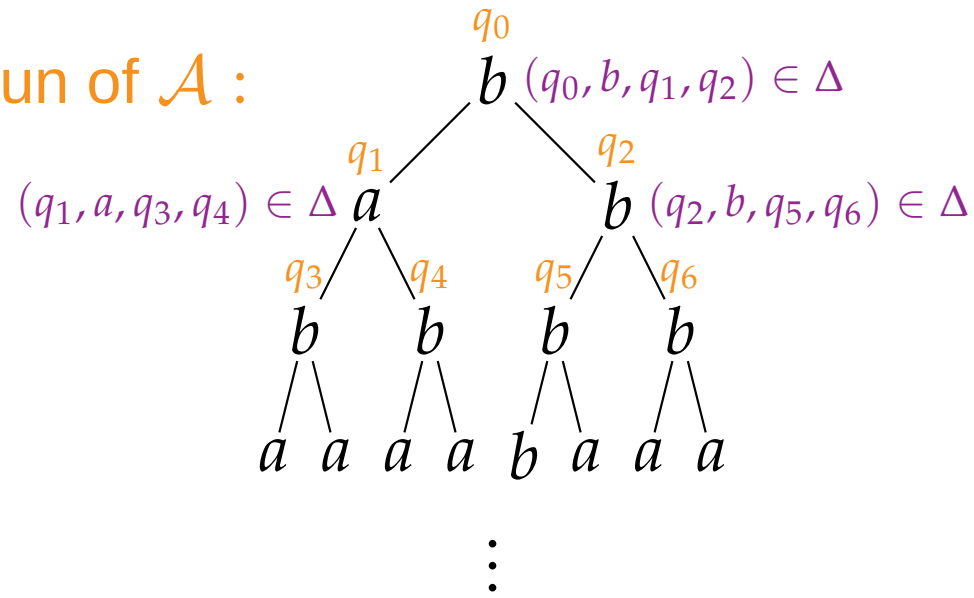
Parity Tree Automata

- Automata $\mathcal{A} = (Q, \Sigma, q_0, \Delta, pri)$ on infinite binary trees
- Transitions of the form (q, a, q', q'')



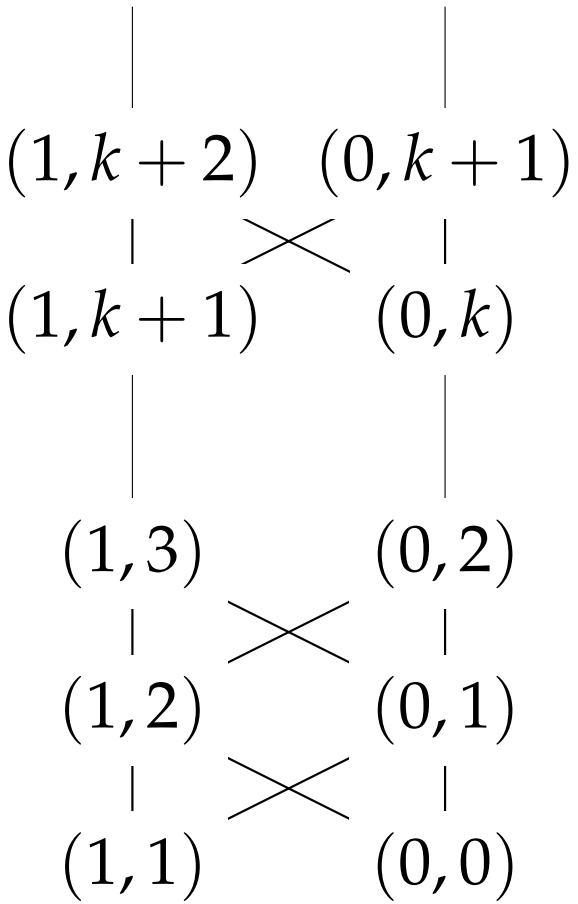
Parity Tree Automata

- Automata $\mathcal{A} = (Q, \Sigma, q_0, \Delta, pri)$ on infinite binary trees
- Transitions of the form (q, a, q', q'')



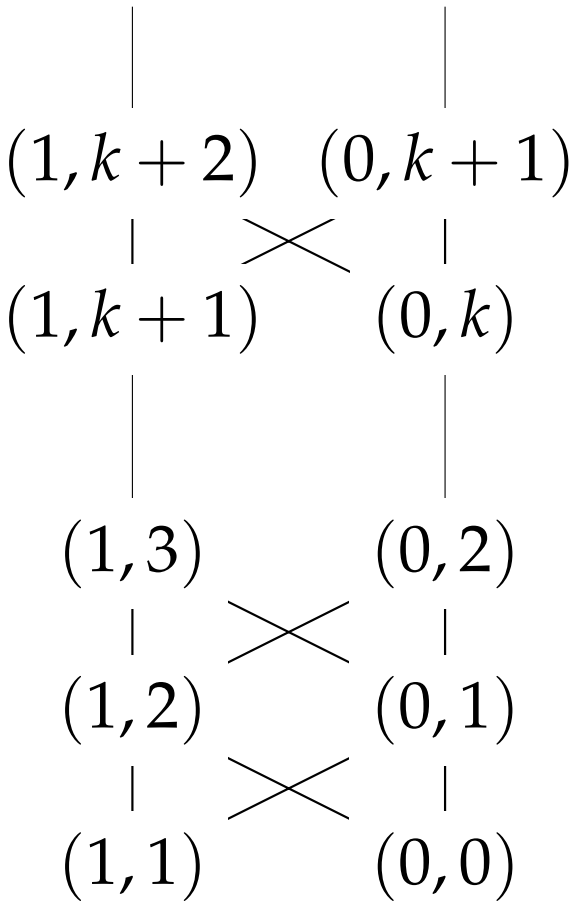
- Priority function $pri : Q \to \mathbb{N}$
- Run accepting if on each path the maximal priority appearing infinitely often is even.

Mostowski Hierarchy



The parity index of $\mathcal{A}$ is the pair (i, j) of highest and lowest priority it uses

Mostowski Hierarchy



The parity index of $\mathcal{A}$ is the pair (i,j) of highest and lowest priority it uses

Deciding the levels of the hierarchy:

Given: Language T and $i < j$

Question: Is there an automaton of index (i,j) for T?

Relation to Fixpoint Alternation

Alternating parity tree automata:

parity index $\leftrightarrow$ alternation depth in μ-calculus

Nondeterministic parity tree automata:

parity index $\leftrightarrow$ alternation depth in a disjunctive version of the μ-calculus

Results on Mostowski hierarchy

- The hierarchy is strict (Niwiński'86)
- Decidability
  - Decidable if the input language is deterministic (Niwinski/Walukiewicz'05)
  - General case unknown

Results on Mostowski hierarchy

- The hierarchy is strict (Niwiński'86)
- Decidability
  - Decidable if the input language is deterministic (Niwinski/Walukiewicz'05)
  - General case unknown

Our result (Colcombet/L.'08):

Theorem. The parity index problem can be effectively reduced to the *uniform universality problem for distance-parity automata*.

Distance-parity automata: general idea

Enrich finite automata such that an accepting run reveals some more information on the way the input was accepted:
a value is assigned to each accepted input

- Augment automaton by a finite number of counters
- Transitions of the automaton can be associated with operations on the counters: increment or reset
- Counters are nested/ordered: manipulating one counter resets all smaller counters
- Counters do not influence the control structure

Distance-parity automata: general idea

Enrich finite automata such that an accepting run reveals some more information on the way the input was accepted:
a value is assigned to each accepted input

- Augment automaton by a finite number of counters
- Transitions of the automaton can be associated with operations on the counters: increment or reset
- Counters are nested/ordered: manipulating one counter resets all smaller counters
- Counters do not influence the control structure

On finite words: used to solve the restricted star-height problem (Kirsten'05)

Computing the cost of a sequence

Fix $Op = \{I_1 < R_1 < I_2 < \cdots < R_n\}$.

- I_k increments counter k, and reset counters $1, \ldots, k-1$,
- R_k resets counters $1, \ldots, k$.

Given $u \in Op^\infty$, $cost(u)$ is the supremum over all counter values seen when executing the sequence.

Example

	I_1	I_1	I_2	I_1	I_2	I_1	R_2	I_1	I_1	I_1	I_2	I_1	I_2	I_1
0	0	0	1	1	2	2	0	0	0	0	1	1	2	2
0	1	2	0	1	0	1	0	1	2	3	0	1	0	1

Computing the cost of a sequence

Fix $Op = \{I_1 < R_1 < I_2 < \cdots < R_n\}$.

- I_k increments counter k, and reset counters $1, \ldots, k-1$,
- R_k resets counters $1, \ldots, k$.

Given $u \in Op^\infty$, $cost(u)$ is the supremum over all counter values seen when executing the sequence.

Example

	I_1	I_1	I_2	I_1	I_2	I_1	R_2	I_1	I_1	I_1	I_2	I_1	I_2	I_1
0	0	0	1	1	2	2	0	0	0	0	1	1	2	2
0	1	2	0	1	0	1	0	1	2	3	0	1	0	1

Computing the cost of a sequence

Fix $Op = \{I_1 < R_1 < I_2 < \cdots < R_n\}$.

- I_k increments counter k, and reset counters $1, \ldots, k-1$,
- R_k resets counters $1, \ldots, k$.

Given $u \in Op^\infty$, $cost(u)$ is the supremum over all counter values seen when executing the sequence.

Example

	I_1	I_1	I_2	I_1	I_2	I_1	R_2	I_1	I_1	I_1	I_2	I_1	I_2	I_1
0	0	0	1	1	2	2	0	0	0	0	1	1	2	2
0	1	2	0	1	0	1	0	1	2	3	0	1	0	1

Can be ω for infinite sequences: $cost(I_1^1 R_1 I_1^2 R_1 I_1^3 \ldots) = \omega$

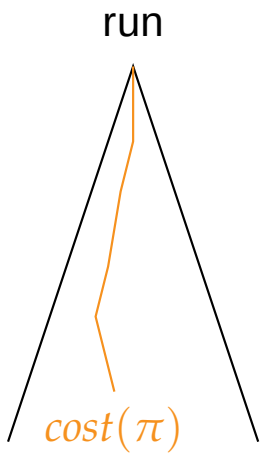
The automaton model

A distance-parity automaton $\mathcal{A}$ is a parity automaton augmented with $d : \Delta \to Op$.

The automaton model

A distance-parity automaton $\mathcal{A}$ is a parity automaton augmented with $d : \Delta \to Op$.

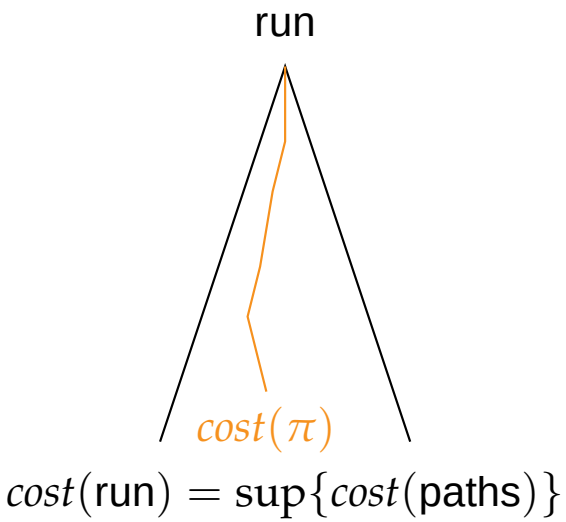
- Each path through a run induces a cost



The automaton model

A distance-parity automaton $\mathcal{A}$ is a parity automaton augmented with $d : \Delta \longrightarrow Op$.

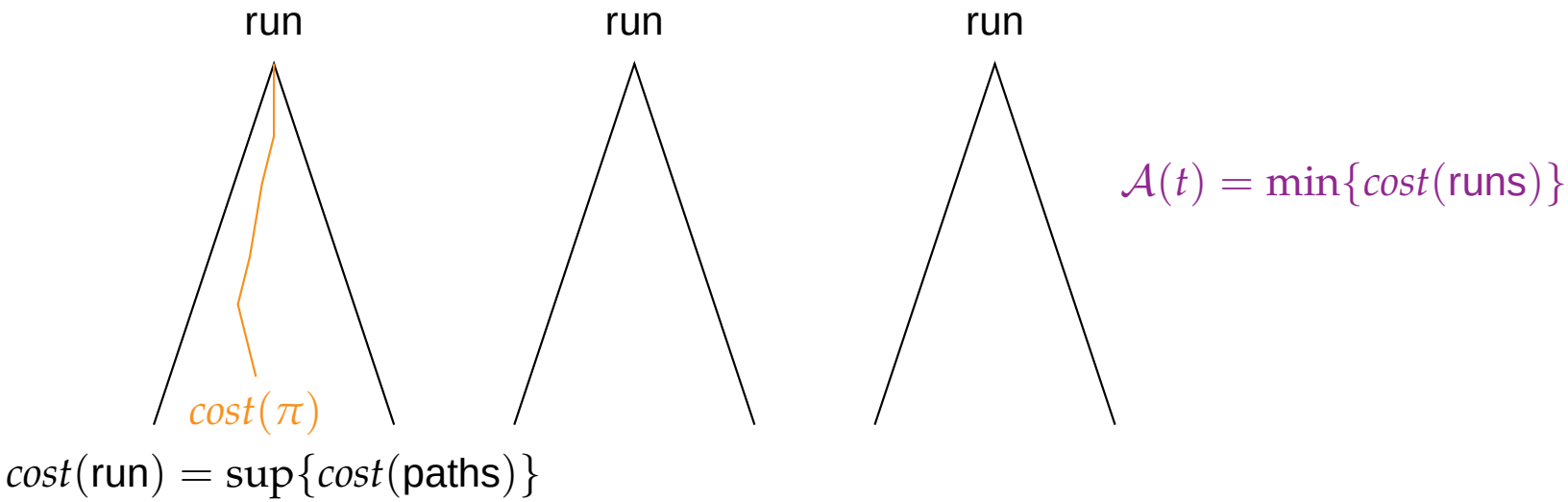
- Each path through a run induces a cost
- The cost of an accepting run is the supremum of cost of the paths



The automaton model

A distance-parity automaton $\mathcal{A}$ is a parity automaton augmented with $d : \Delta \to Op$.

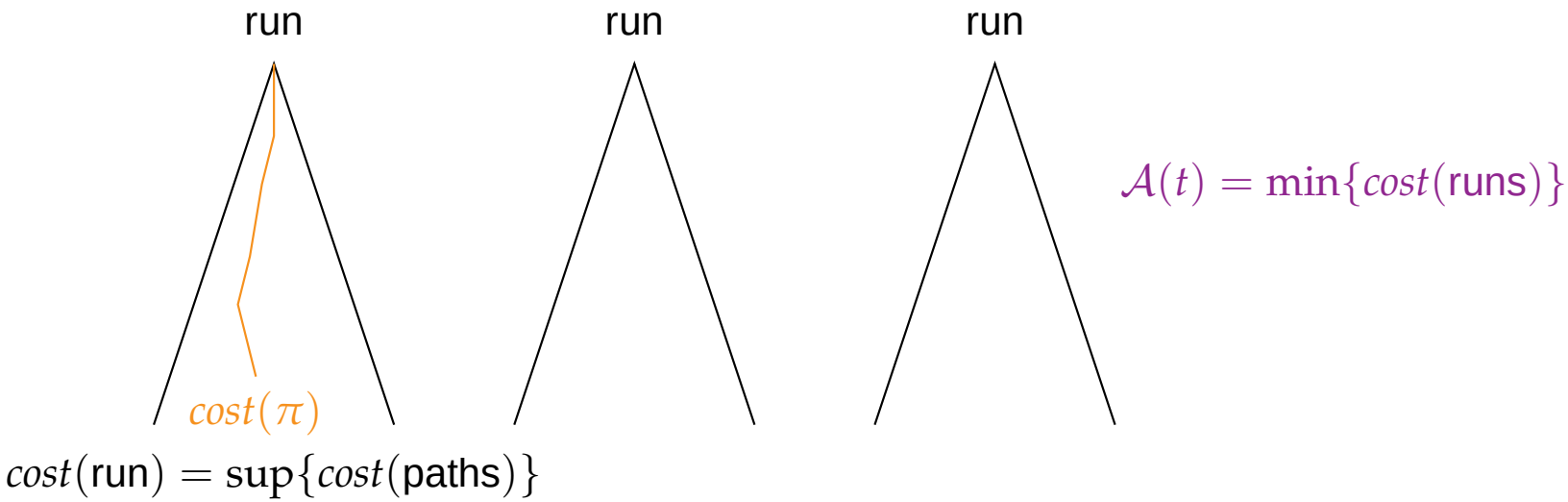
- Each path through a run induces a cost
- The cost of an accepting run is the supremum of cost of the paths
- The cost of a tree is the minimum cost over all accepting runs (ω if there are no accepting runs)



The automaton model

A distance-parity automaton $\mathcal{A}$ is a parity automaton augmented with $d : \Delta \to Op$.

- Each path through a run induces a cost
- The cost of an accepting run is the supremum of cost of the paths
- The cost of a tree is the minimum cost over all accepting runs (ω if there are no accepting runs)



- $\mathcal{A}$ computes a cost function over the domain of infinite trees.

Boundedness problem

Uniform universality: Is the function computed by $\mathcal{A}$ bounded?

Boundedness problem

Uniform universality: Is the function computed by $\mathcal{A}$ bounded?

Theorem. The parity index problem can be effectively reduced to the uniform universality problem for distance-parity automata.

Boundedness problem

Uniform universality: Is the function computed by $\mathcal{A}$ bounded?

Theorem. The parity index problem can be effectively reduced to the uniform universality problem for distance-parity automata.

But: We are not (yet) able to solve the uniform universality problem for distance-parity automata.

Main problem: Algorithms for automata on infinite trees are tightly connected to infinite games and finite memory determinacy results. We are lacking such results.

$\rightsquigarrow$ Since infinite trees are too difficult for the moment, we first develop tools and a general theory for cost functions in the setting of finite trees.

Cost functions and counters

We call a function $f : D \to \mathbb{N} \cup \{\omega\}$ a cost function.

We use counters to compute cost functions:

- Several counters with operations

  increment, reset, and check

- Can be nested (manipulating counter i resets counters $1, \ldots, i-1$) or non-nested.

Cost functions and counters

We call a function $f : D \to \mathbb{N} \cup \{\omega\}$ a cost function.

We use counters to compute cost functions:

- Several counters with operations

  increment, reset, and check

- Can be nested (manipulating counter i resets counters $1, \ldots, i-1$) or non-nested.
- Value of an instruction sequence w:
  - B-semantics

    $$cost^{\mathsf{B}}(w) = \max\{\text{counter values on check positions in } w\}$$

  - S-semantics

    $$cost^{\mathsf{S}}(w) = \min\{\text{counter values on check positions in } w\}$$

  (convention $\max \emptyset = 0$ and $\min \emptyset = \infty$)

Equivalence of cost functions

Let $f, g : D \to \mathbb{N} \cup \{\infty\}$ be two cost functions over the same domain.

$$f \preceq g \quad \Leftrightarrow \quad \forall X \subseteq D : \ g \text{ bounded on } X \ \to f \text{ bounded on } X$$
$$f \approx g \quad \Leftrightarrow \quad f \preceq g \text{ and } g \preceq f$$

Equivalence of cost functions

Let $f, g : D \to \mathbb{N} \cup \{\infty\}$ be two cost functions over the same domain.

$$f \preceq g \quad \Leftrightarrow \quad \forall X \subseteq D : \ g \text{ bounded on } X \ \to f \text{ bounded on } X$$
$$f \approx g \quad \Leftrightarrow \quad f \preceq g \text{ and } g \preceq f$$

Example: $f, g : \{a, b\}^* \to \mathbb{N} \cup \{\infty\}$ with

$$\begin{aligned} f(w) &= 2 \cdot |w|_a \\ g(w) &= |w| \end{aligned}$$

Then $f \preceq g$ but not $g \preceq f$ because f is bounded on b^* but g is not.

Equivalence of cost functions

Let $f, g : D \to \mathbb{N} \cup \{\infty\}$ be two cost functions over the same domain.

$$f \preceq g \;\Leftrightarrow\; \forall X \subseteq D :\; g \text{ bounded on } X \;\to f \text{ bounded on } X$$
$$f \approx g \;\Leftrightarrow\; f \preceq g \text{ and } g \preceq f$$

Example: $f, g : \{a, b\}^* \to \mathbb{N} \cup \{\infty\}$ with

$$\begin{aligned} f(w) &= 2 \cdot |w|_a \\ g(w) &= |w| \end{aligned}$$

Then $f \preceq g$ but not $g \preceq f$ because f is bounded on b^* but g is not.

$\rightsquigarrow$ equivalence and "inclusion" of cost functions

The Inclusion Problem

Given two automata computing cost functions f_1 and f_2, decide if

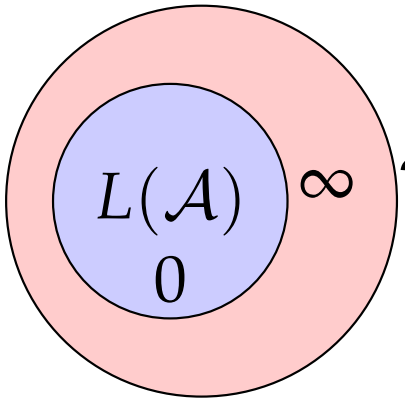
$$f_1 \preceq f_2$$

The Inclusion Problem

Given two automata computing cost functions f_1 and f_2, decide if

$$f_1 \preceq f_2$$

Special cases for a single automaton $\mathcal{A}$ computing f:

- Uniform universality: $f \preceq 0$
  Is f bounded on the full domain?
- Emptiness: $\infty \preceq f$
  Is f unbounded on all subsets of the domain?
- Limitedness: $f \preceq$ 

?

1 How it started: infinite trees and the parity index problem

2 Basics on cost functions

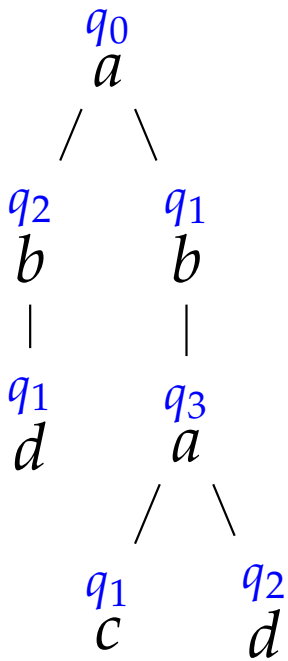
3 Automata on finite trees

- Decision problems
- Transformations

4 Conclusion

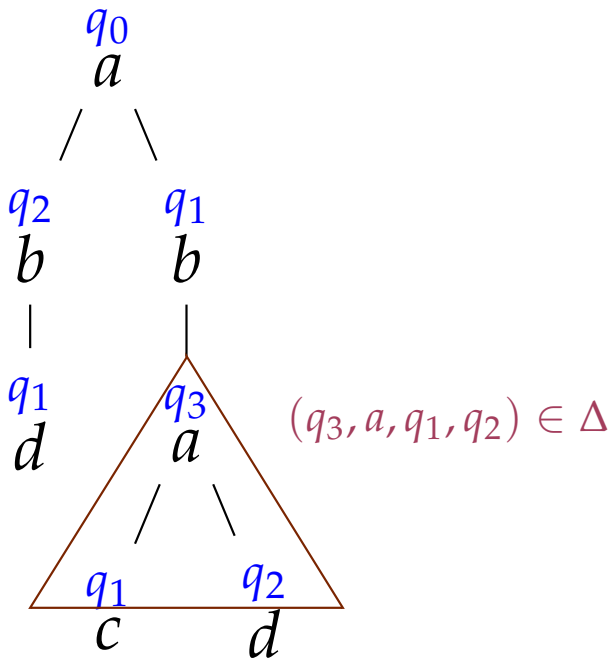
Tree Automata

Tree automaton: $\mathcal{A} = (Q, \Sigma, Q_0, \Delta)$ with transitions
$\Delta \subseteq \bigcup Q \times \Sigma_i \times Q^i$



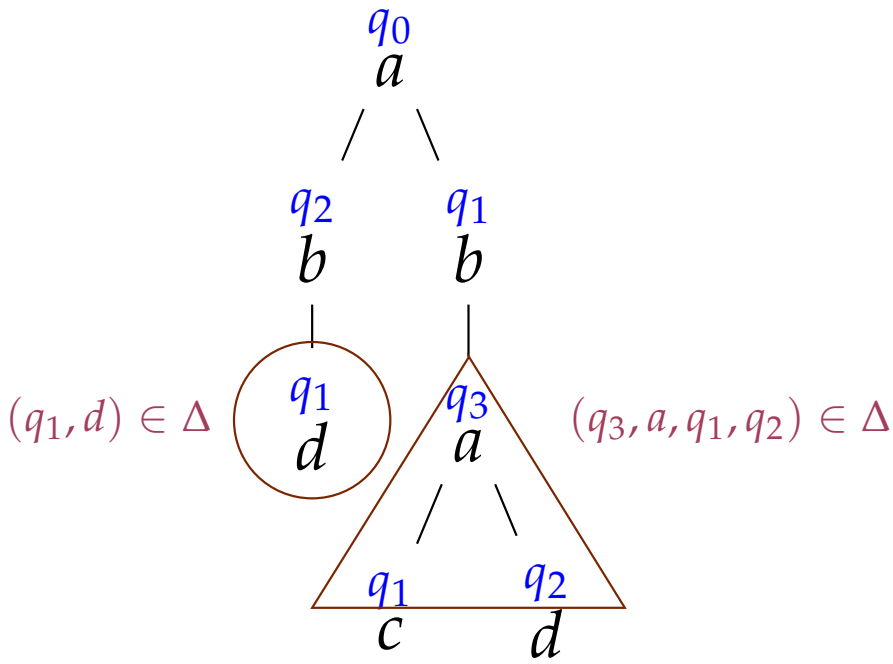
Tree Automata

Tree automaton: $\mathcal{A} = (Q, \Sigma, Q_0, \Delta)$ with transitions $\Delta \subseteq \bigcup Q \times \Sigma_i \times Q^i$



Tree Automata

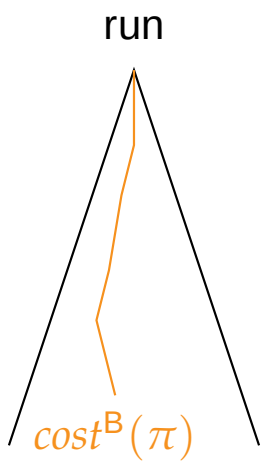
Tree automaton: $\mathcal{A} = (Q, \Sigma, Q_0, \Delta)$ with transitions $\Delta \subseteq \bigcup Q \times \Sigma_i \times Q^i$



Adding costs – B-semantics

Counter operations on transitions. Counters evolve independently along each path (from root towards leafs):

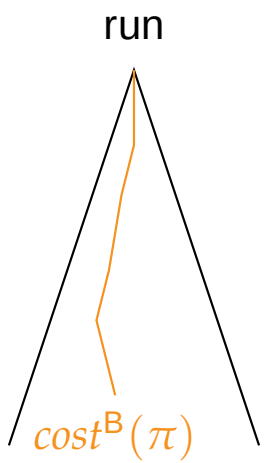
$$\mathcal{A}^{\mathsf{B}}(t) = \min_{\rho \text{ acc. run on } t} \max_{\pi \text{ path in } \rho} \underbrace{\max\{\text{check values on } \pi\}}_{cost^{\mathsf{B}}(\pi)}$$



Adding costs – B-semantics

Counter operations on transitions. Counters evolve independently along each path (from root towards leafs):

$$\mathcal{A}^{\mathsf{B}}(t) = \min_{\rho \text{ acc. run on } t} \max_{\pi \text{ path in } \rho} \underbrace{\max\{\text{check values on } \pi\}}_{cost^{\mathsf{B}}(\pi)}$$

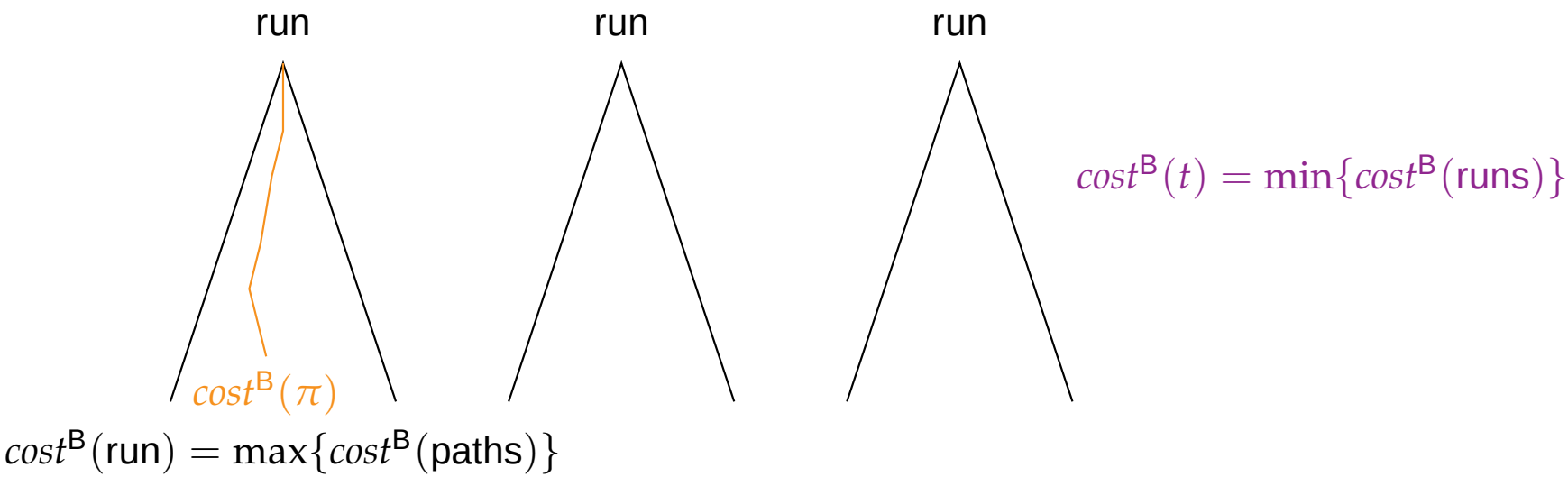


$cost^{\mathsf{B}}(\text{run}) = \max\{cost^{\mathsf{B}}(\text{paths})\}$

Adding costs – B-semantics

Counter operations on transitions. Counters evolve independently along each path (from root towards leafs):

$$\mathcal{A}^{\mathsf{B}}(t) = \min_{\rho \text{ acc. run on } t} \max_{\pi \text{ path in } \rho} \underbrace{\max\{\text{check values on } \pi\}}_{cost^{\mathsf{B}}(\pi)}$$

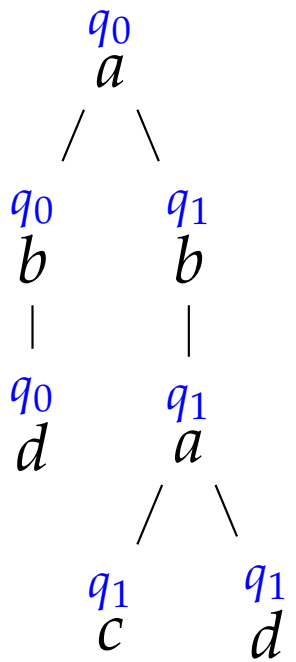


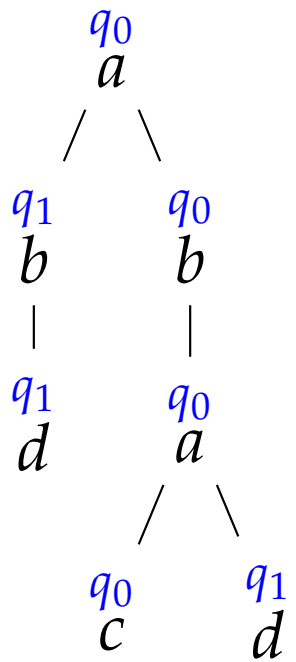
Example

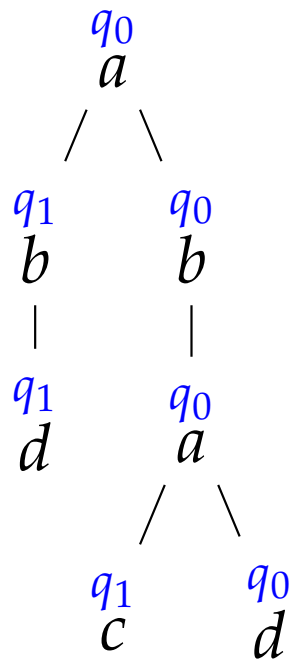
B-automaton associating to each tree the length of its shortest path:

q_0 = guess path
q_1 = dummy state

Transitions from q_0 increment and check the counter





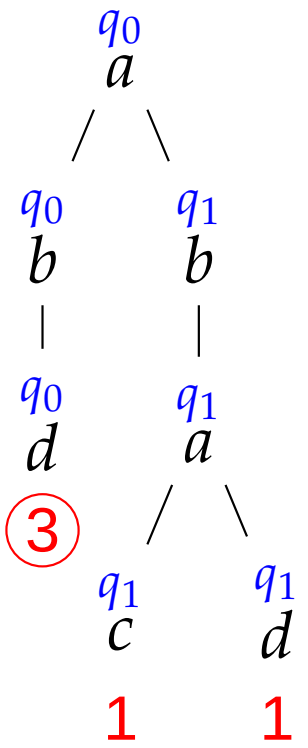


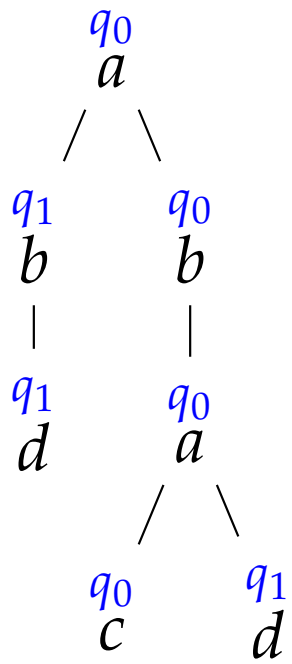
Example

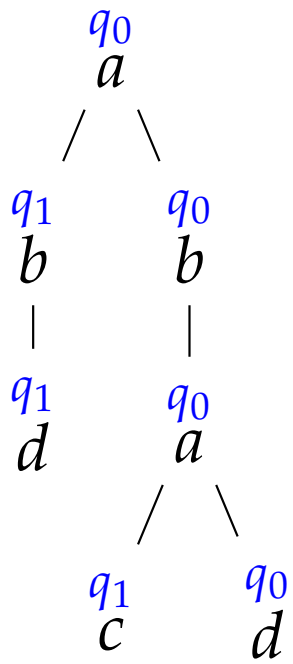
B-automaton associating to each tree the length of its shortest path:

q_0 = guess path
q_1 = dummy state

Transitions from q_0 increment and check the counter





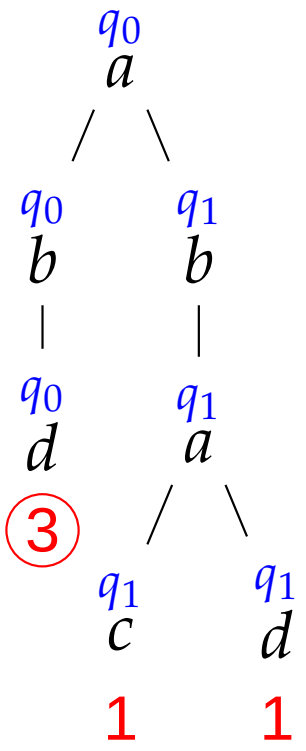


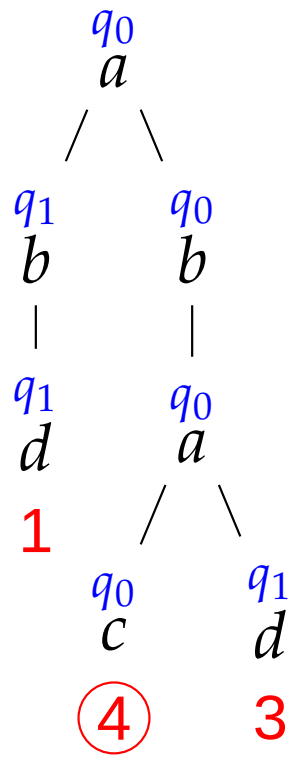
Example

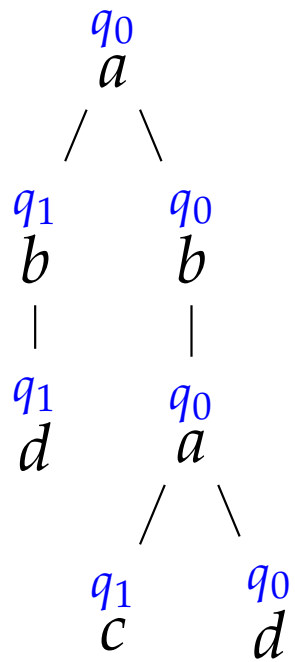
B-automaton associating to each tree the length of its shortest path:

q_0 = guess path
q_1 = dummy state

Transitions from q_0 increment and check the counter





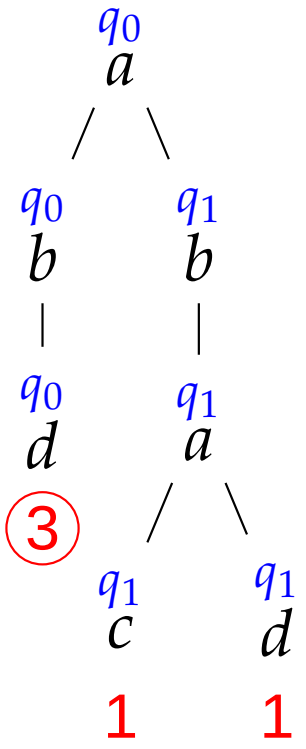


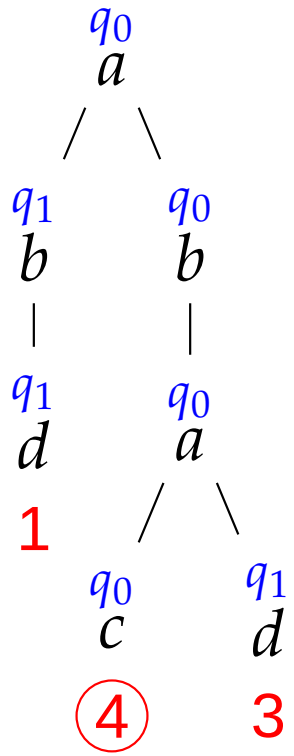
Example

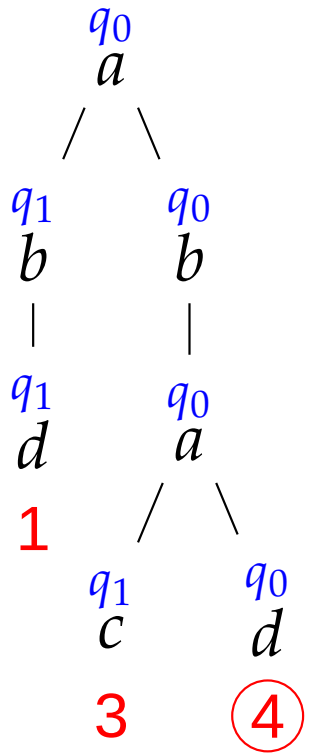
B-automaton associating to each tree the length of its shortest path:

q_0 = guess path
q_1 = dummy state

Transitions from q_0 increment and check the counter





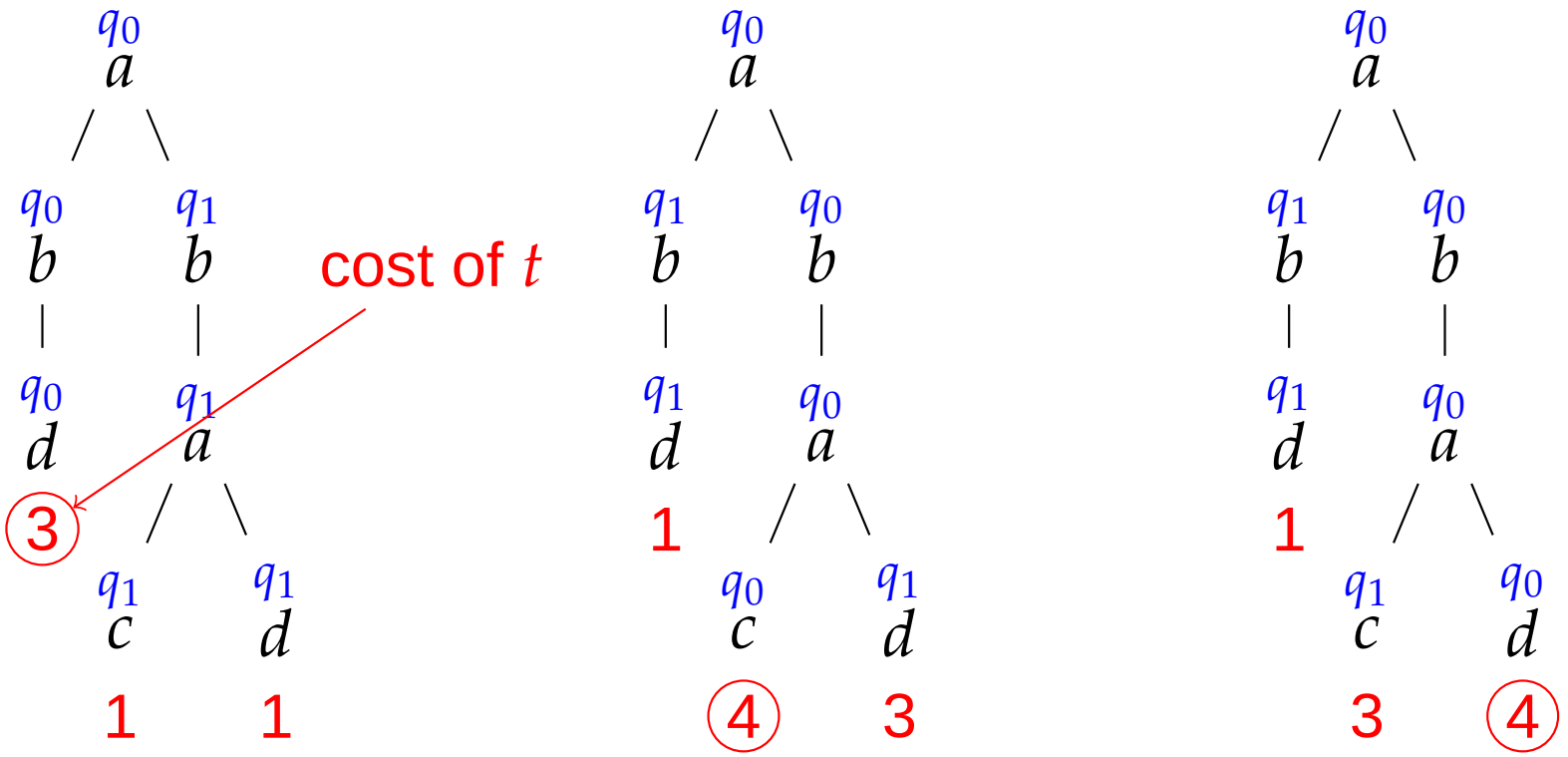


Example

B-automaton associating to each tree the length of its shortest path:

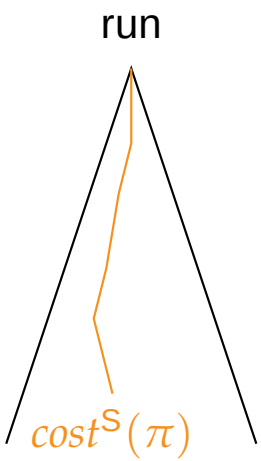
q_0 = guess path
q_1 = dummy state

Transitions from q_0 increment and check the counter



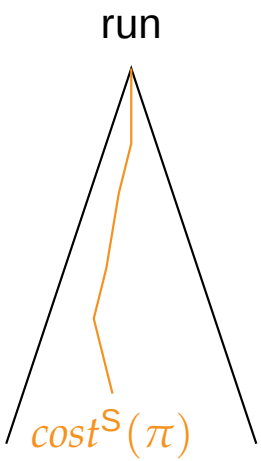
S-semantics

$$\mathcal{A}^{\mathsf{S}}(t) = \max_{\rho \text{ acc. run on } t} \min_{\pi \text{ path in } \rho} \underbrace{\min\{\text{check values on } \pi\}}_{cost^{\mathsf{S}}(\pi)}$$



S-semantics

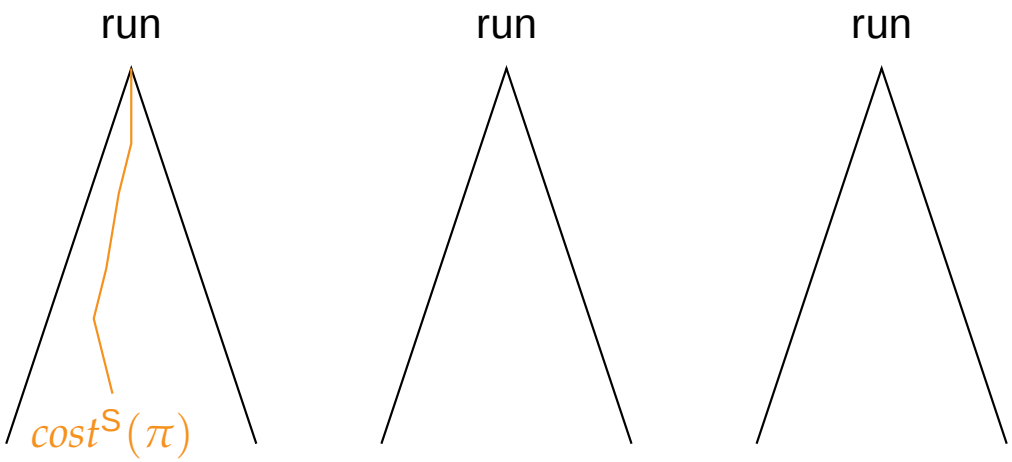
$$\mathcal{A}^{\mathsf{S}}(t) = \max_{\rho \text{ acc. run on } t} \min_{\pi \text{ path in } \rho} \underbrace{\min\{\text{check values on } \pi\}}_{cost^{\mathsf{S}}(\pi)}$$



$$cost^{\mathsf{S}}(\text{run}) = \min\{cost^{\mathsf{S}}(\text{paths})\}$$

S-semantics

$$\mathcal{A}^{\mathsf{S}}(t) = \max_{\rho \text{ acc. run on } t} \min_{\pi \text{ path in } \rho} \underbrace{\min\{\text{check values on } \pi\}}_{cost^{\mathsf{S}}(\pi)}$$



$cost^{\mathsf{S}}(t) = \max\{cost^{\mathsf{S}}(\text{runs})\}$

$cost^{\mathsf{S}}(\text{run}) = \min\{cost^{\mathsf{S}}(\text{paths})\}$

Goals and Techniques

Goals:

- Decide the inclusion problem for regular cost functions over trees.

$$\mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{S}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{S}}$$

Goals and Techniques

Goals:

- Decide the inclusion problem for regular cost functions over trees.

$$\mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{S}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{S}}$$

Simplest version: $\mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}$

Goals and Techniques

Goals:

- Decide the inclusion problem for regular cost functions over trees.

$$\mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{S}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{S}}$$

  Simplest version: $\mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}$

- Show equivalence of the models: transform between different types of automata (B/S, nested/non-nested, ...)

Goals and Techniques

Goals:

- Decide the inclusion problem for regular cost functions over trees.

$$\mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{S}}, \quad \mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}, \quad \mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{S}}$$

  Simplest version: $\mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}$
- Show equivalence of the models: transform between different types of automata (B/S, nested/non-nested, ...)

Classical techniques in new setting:

- Games to capture membership and emptiness problems for tree automata
- Checking path properties using word automata: run a word automaton along all paths

Decision problems

Uniform universality

- B-semantics: $\mathcal{A}^{\mathsf{B}} \preceq 0$

$$\exists N\ \forall \text{tree } t\ \exists \text{run } \rho\ \forall \text{path } \pi : \max\{\text{checks in } \pi\} < N$$

Uniform universality

- B-semantics: $\mathcal{A}^{\mathsf{B}} \preceq 0$

$$\exists N\ \forall \text{tree } t\ \exists \text{run } \rho\ \forall \text{path } \pi : \max\{\text{checks in } \pi\} < N$$

- S-semantics: $\mathcal{A}^{\mathsf{S}} \preceq 0$

$$\exists N\ \forall \text{tree } t\ \forall \text{run } \rho\ \exists \text{path } \pi : \min\{\text{checks in } \pi\} < N$$

Uniform universality game

S-semantics: $\mathcal{A}^{\mathsf{S}} \preceq 0$:

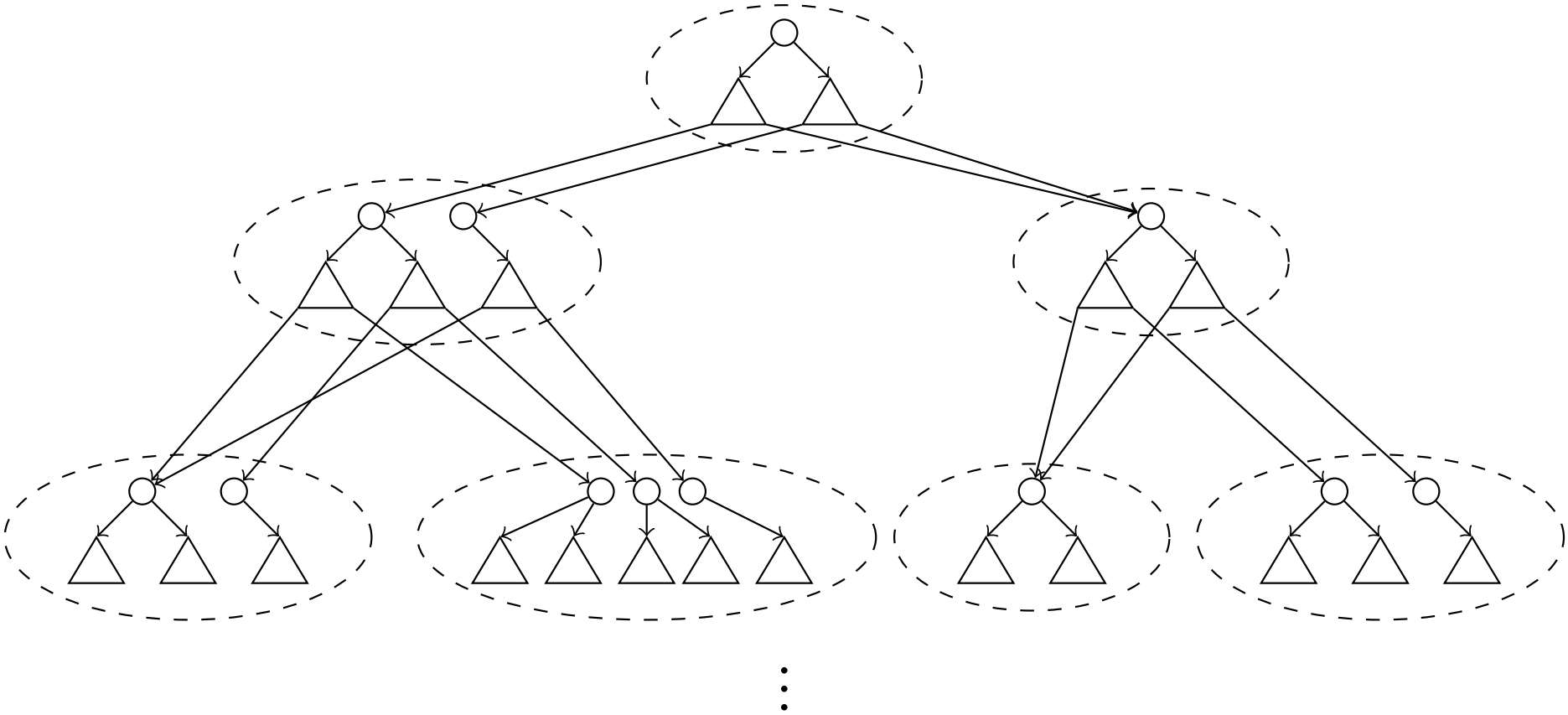
$$\exists N\ \forall \text{tree } t\ \forall \text{run } \rho\ \exists \text{path } \pi : \min\{\text{checks in } \pi\} < N$$

The game starts in the initial state of $\mathcal{A}$.

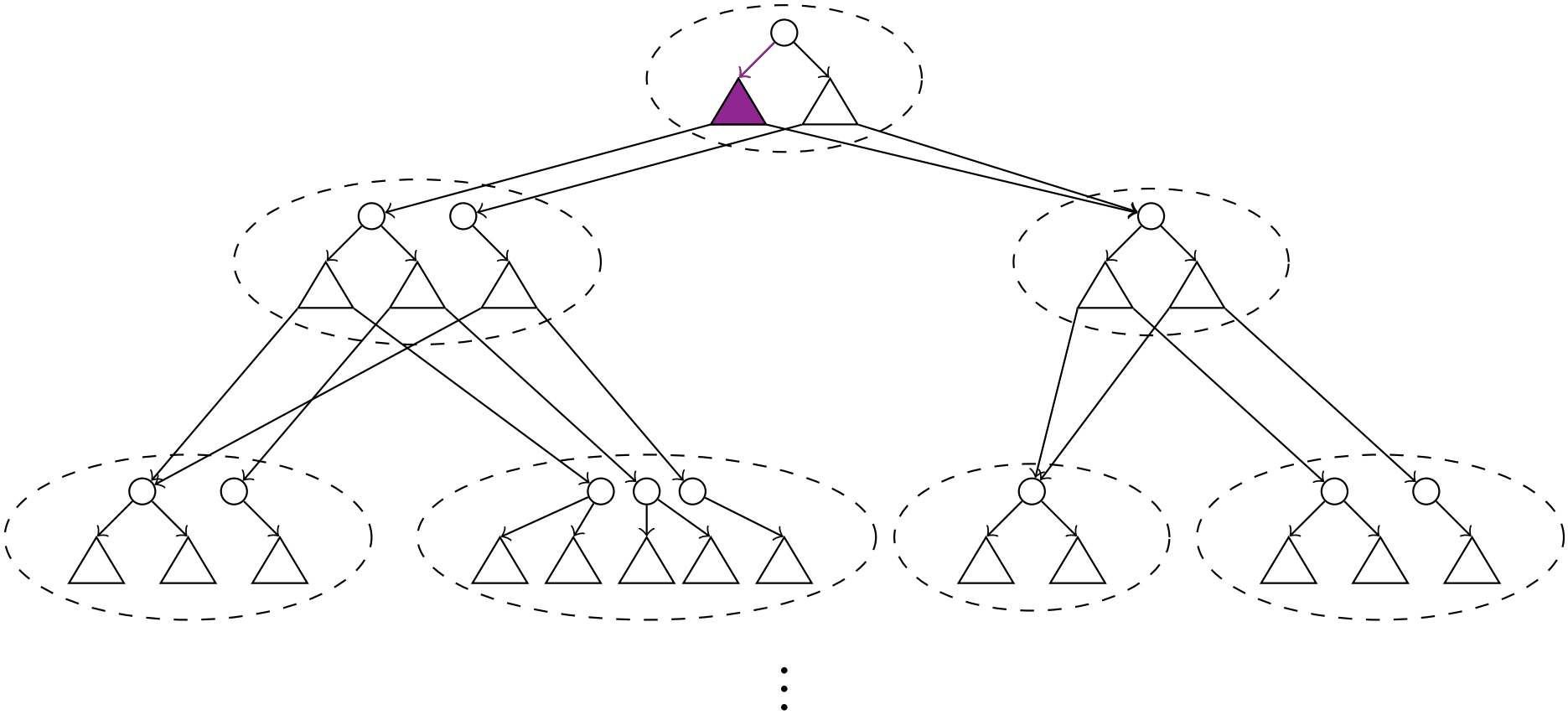
- From a state q player Automaton chooses a transition, e.g., (q, a, q_1, q_2)
- Player Pathfinder chooses a direction q_1 or q_2.
- The game continues in the state chosen by Pathfinder.
- If Automaton chooses a leaf transition (q, a), the play stops.

The winning condition depends on N: Automaton wins a play iff it reaches a leaf transition and the cost of the play is at least N.

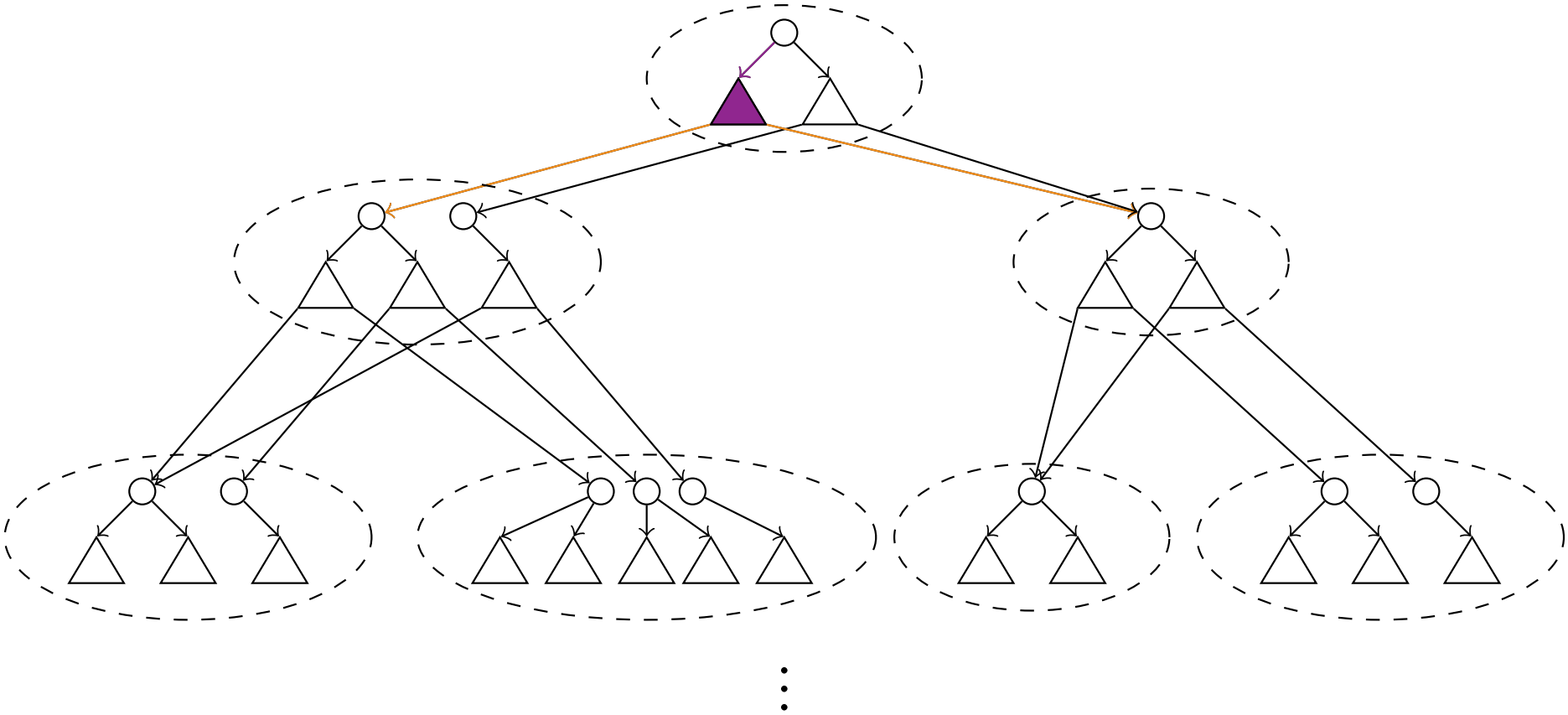
Illustration



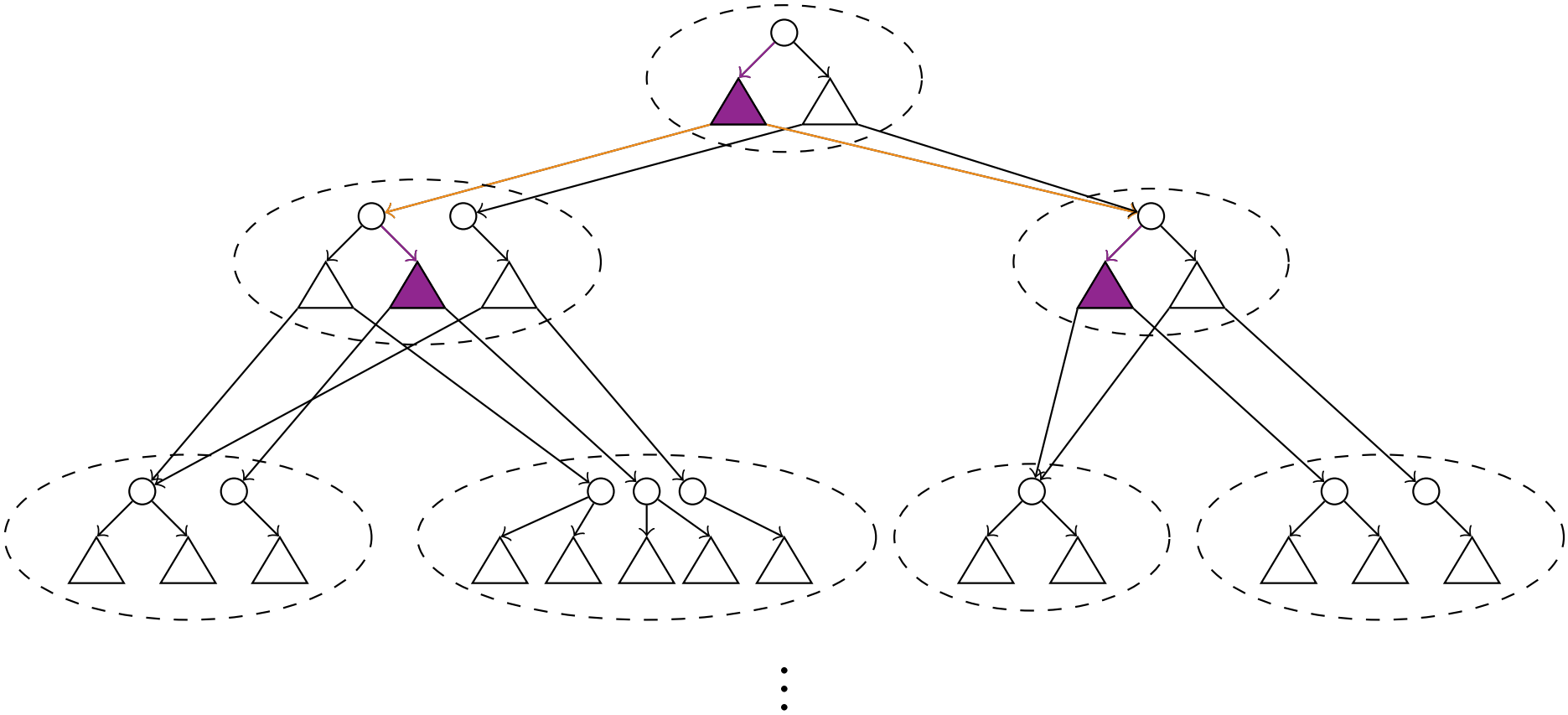
Illustration



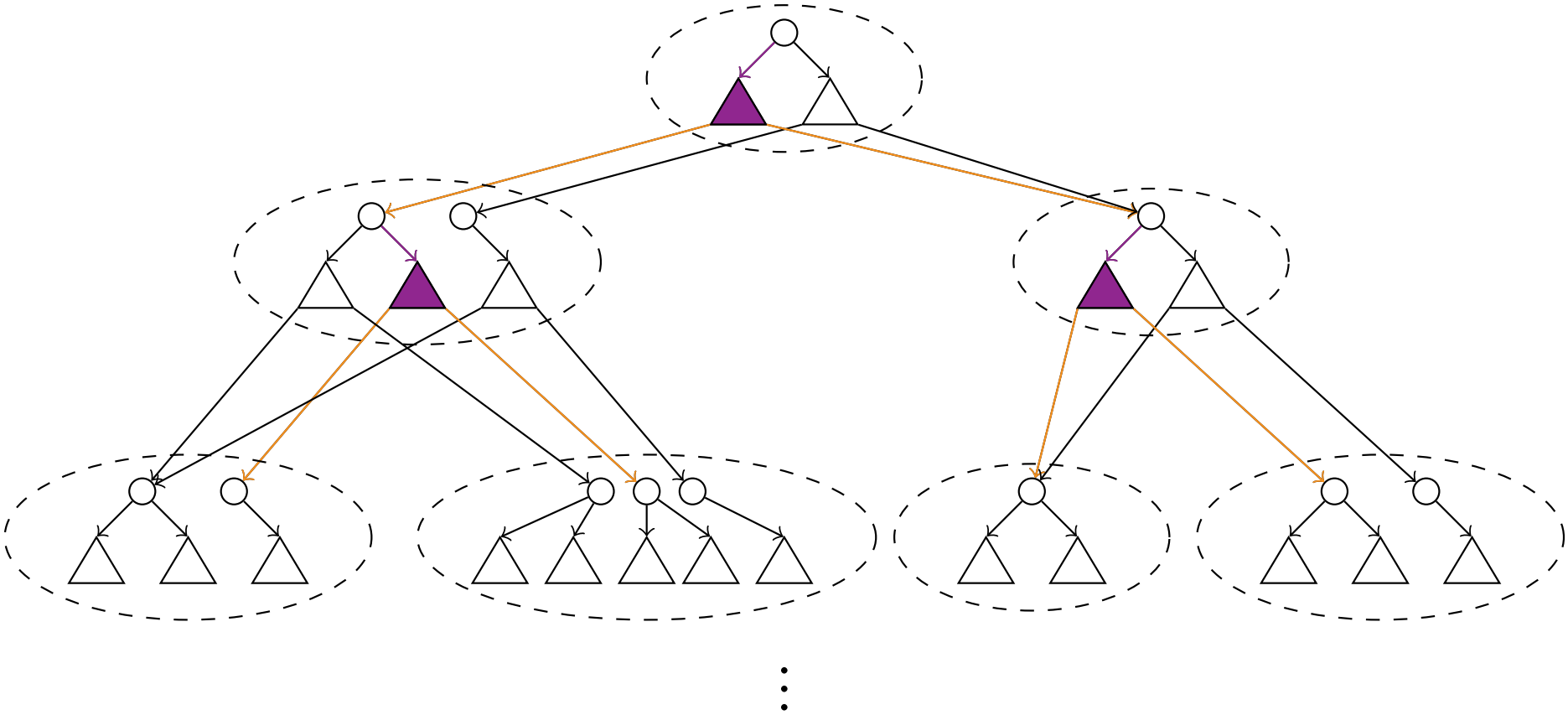
Illustration



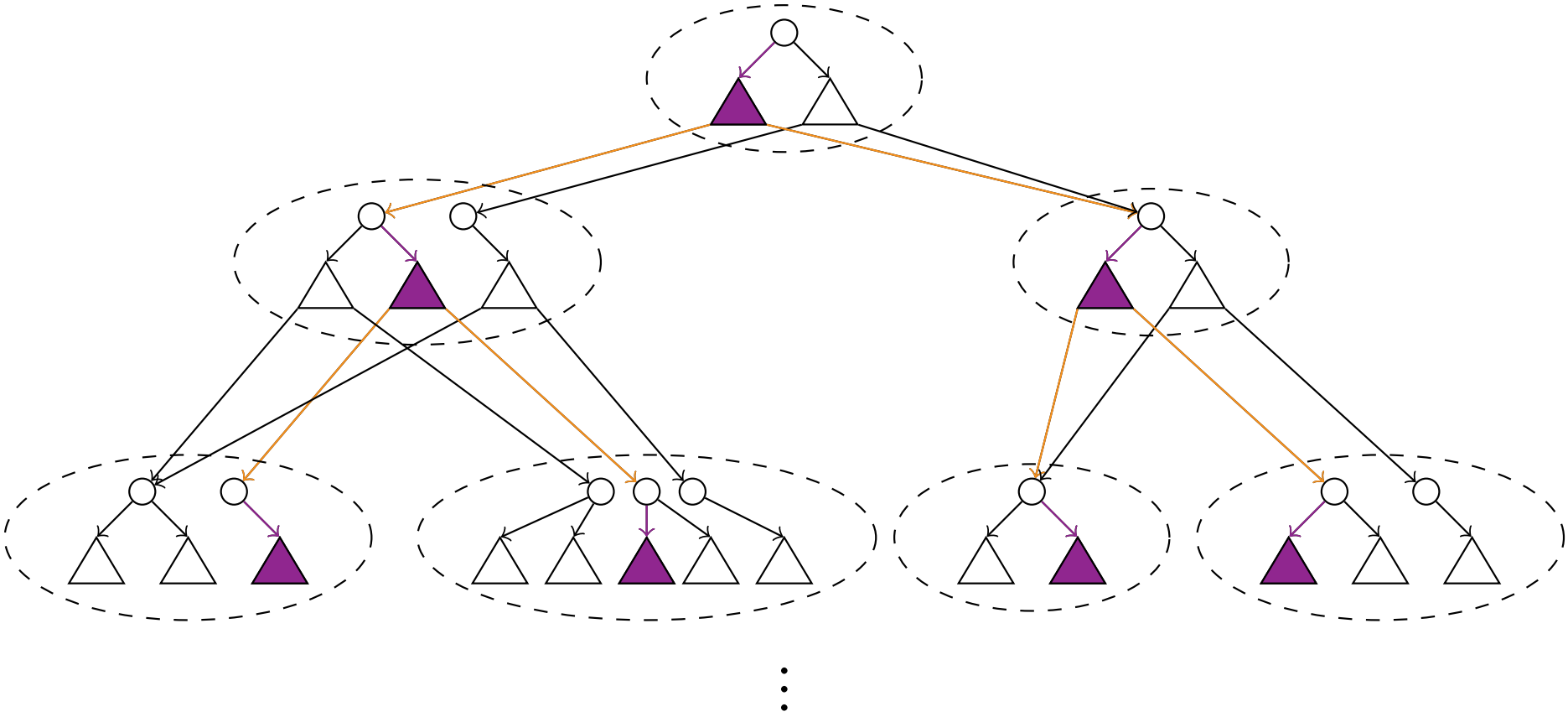
Illustration



Illustration

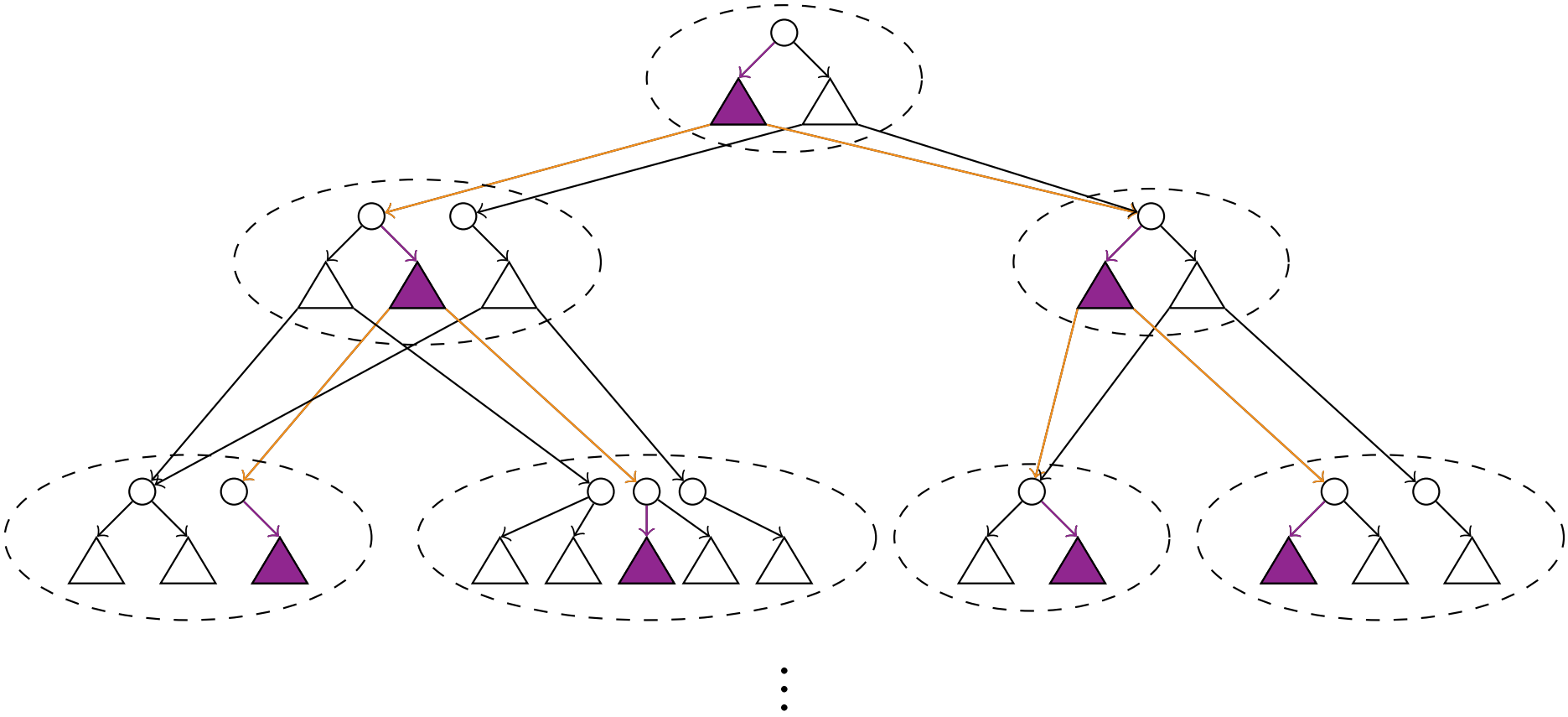


Illustration



winning strategy of Automaton $\leftrightarrow$ accepting run of cost $\geq N$

Illustration



winning strategy of Automaton $\leftrightarrow$ accepting run of cost $\geq N$

winning strategy of Pathfinder $\leftrightarrow$ all accepting runs of cost $< N$

Solving uniform universality

Observation 1. The automaton is uniformly universal iff Pathfinder has a winning strategy in the uniform universality game for some N.

Solving uniform universality

Observation 1. The automaton is uniformly universal iff Pathfinder has a winning strategy in the uniform universality game for some N.

Observation 2. Checking whether Automaton has a winning strategy for every N can be reduced to a Muller game: Automaton wants to reach a loop in which every counter is incremented but neither reset nor checked.

Solving uniform universality

Observation 1. The automaton is uniformly universal iff Pathfinder has a winning strategy in the uniform universality game for some N.

Observation 2. Checking whether Automaton has a winning strategy for every N can be reduced to a Muller game: Automaton wants to reach a loop in which every counter is incremented but neither reset nor checked.

Consequence. Deciding $\mathcal{A}^{\mathsf{S}} \preceq 0$ can be reduced to solving a Muller game of size of the automaton $\mathcal{A}$.

The general question $\mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}$ can be solved in a similar fashion.

Solving uniform universality

Observation 1. The automaton is uniformly universal iff Pathfinder has a winning strategy in the uniform universality game for some N.

Observation 2. Checking whether Automaton has a winning strategy for every N can be reduced to a Muller game: Automaton wants to reach a loop in which every counter is incremented but neither reset nor checked.

Consequence. Deciding $\mathcal{A}^{\mathsf{S}} \preceq 0$ can be reduced to solving a Muller game of size of the automaton $\mathcal{A}$.

The general question $\mathcal{A}_1^{\mathsf{S}} \preceq \mathcal{A}_2^{\mathsf{B}}$ can be solved in a similar fashion.

To solve the other cases, e.g., $\mathcal{A}_1^{\mathsf{B}} \preceq \mathcal{A}_2^{\mathsf{B}}$ we transform between the models.

Transformations

Complementation: from B to S

Given a B-automaton $\mathcal{A}$, construct an S-automaton $\widetilde{A}$ such that

$$\mathcal{A}^{\mathsf{B}} \approx \widetilde{\mathcal{A}}^{\mathsf{S}}$$

Complementation: from B to S

Given a B-automaton $\mathcal{A}$, construct an S-automaton $\widetilde{A}$ such that

$$\mathcal{A}^{\mathsf{B}} \approx \widetilde{\mathcal{A}}^{\mathsf{S}}$$

From $\min\max$ to $\max\min$:

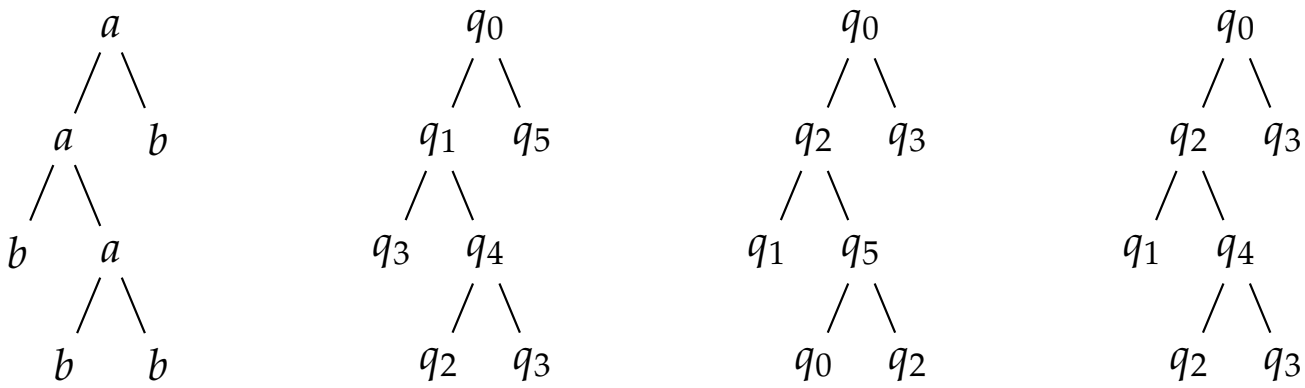
B-automaton:

- minimize over all runs ρ
- while maximizing over all paths π in ρ

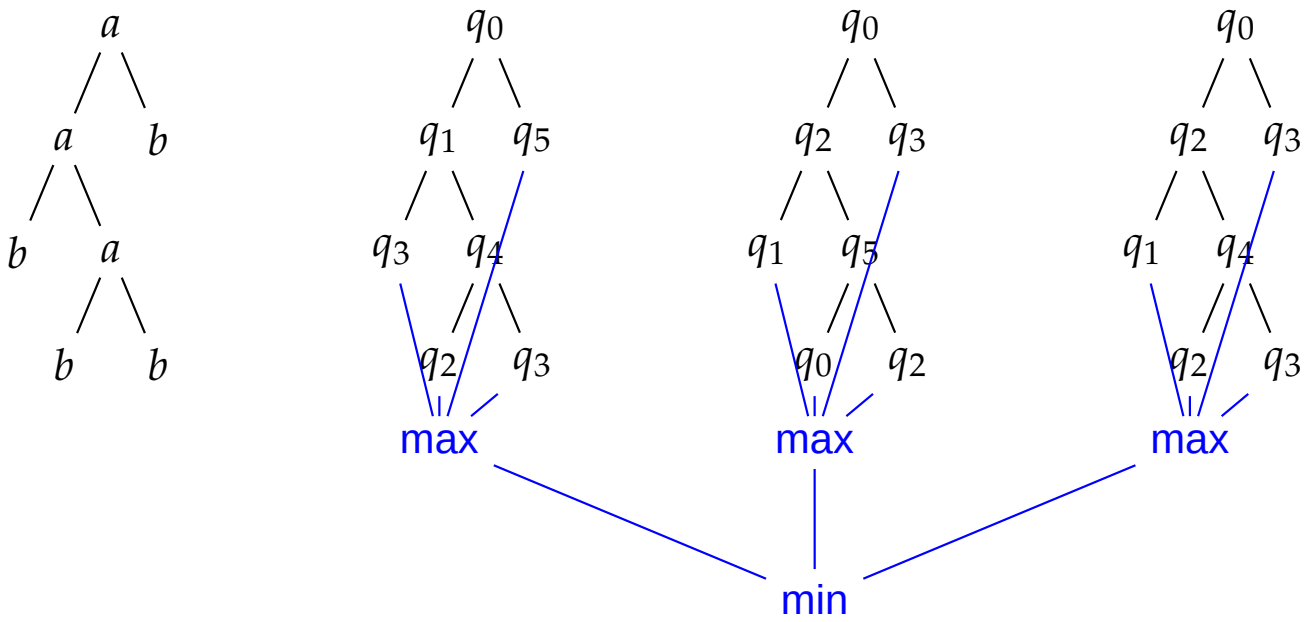
Equivalently:

- maximize over sets Π of paths that contain one path from each run
- and minimize over all paths π in Π.

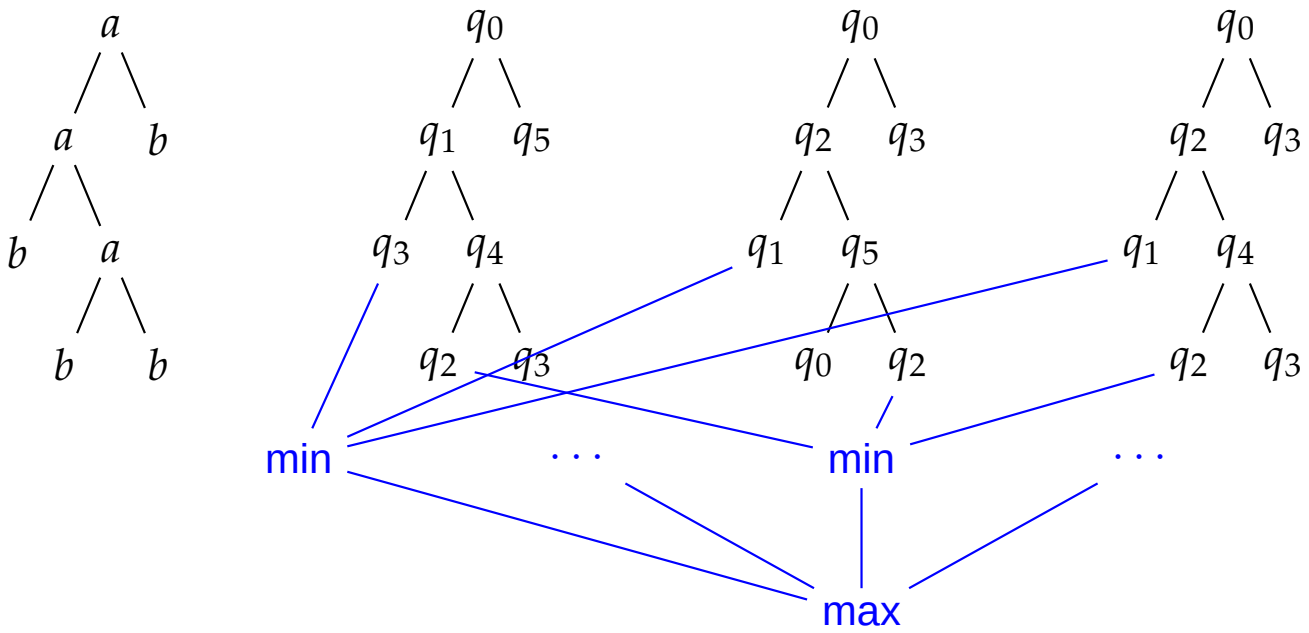
From B to S



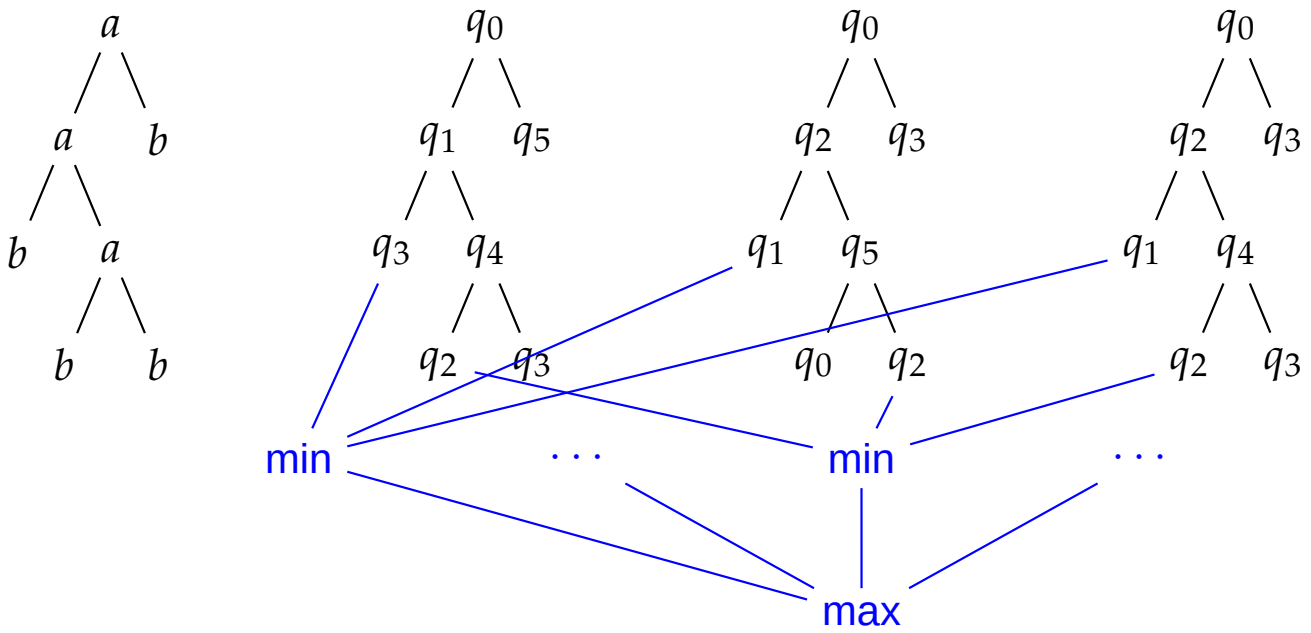
From B to S


a
a
b
b
a
b
b
q_0
q_1
q_5
q_3
q_4
q_2
q_3
q_0
q_2
q_3
q_1
q_5
q_0
q_2
q_0
q_2
q_3
q_1
q_4
q_2
q_3
max
max
max
min

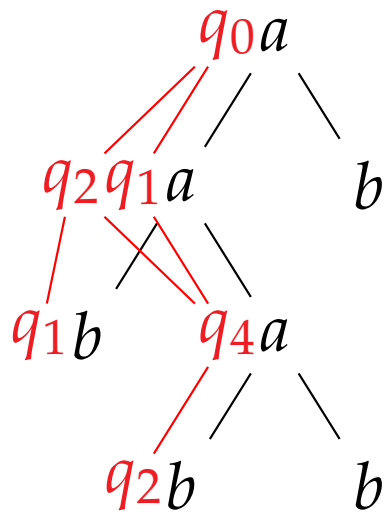
From B to S


a
a
b
b
a
b
b
q_0
q_1
q_5
q_3
q_4
q_2
q_3
q_0
q_2
q_3
q_1
q_5
q_0
q_2
q_0
q_2
q_3
q_1
q_4
q_2
q_3
min
· · ·
min
· · ·
max

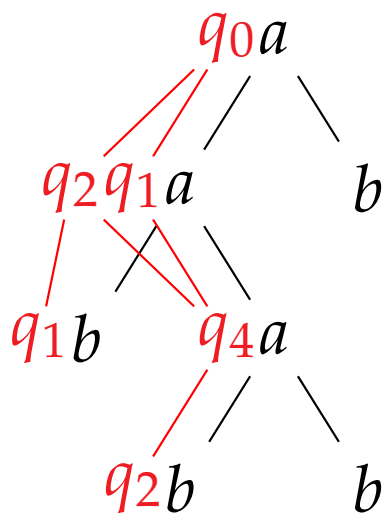
From B to S



- Input for S-automaton: trees annotated with run paths of the B-automaton
- S-automaton computes minimum cost over annotated paths
- Removing annotations by projection: max over all annotations



Techniques for solving the problems



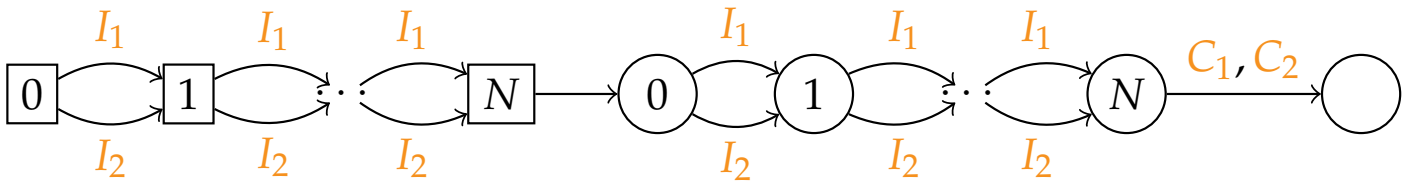
- It suffices to consider rather simple combinations of run paths for the annotations $\rightsquigarrow$ positional (or finite memory) determinacy of membership game
- S-automaton computing minimum cost over annotated paths $\rightsquigarrow$ use results on word automata: history determinism

Games

In our setting: acyclic games (finite duration)

B-games: Player 0 (circle) wants to minimize the counter values.

Example:

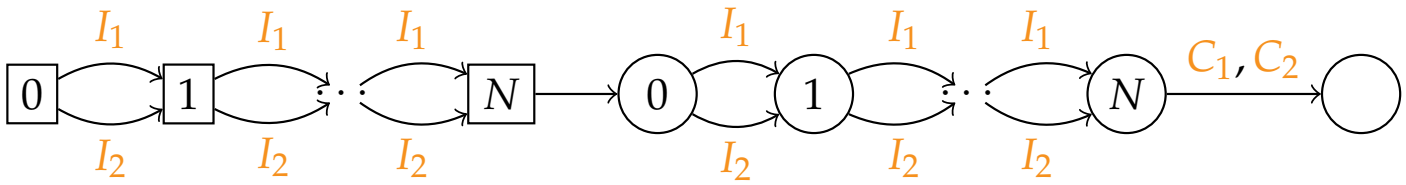


Games

In our setting: acyclic games (finite duration)

B-games: Player 0 (circle) wants to minimize the counter values.

Example:



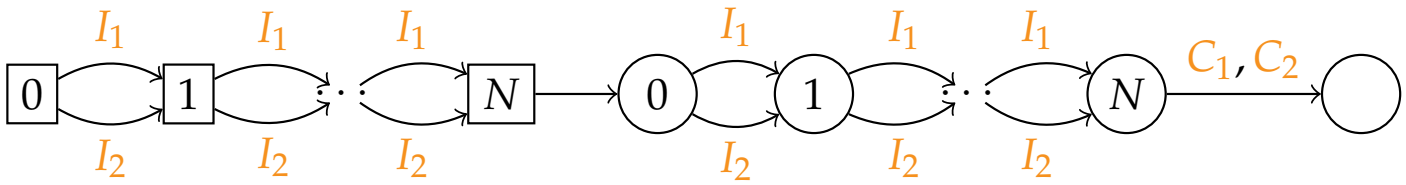
- Optimal strategy: value N; needs memory of size N

Games

In our setting: acyclic games (finite duration)

B-games: Player 0 (circle) wants to minimize the counter values.

Example:



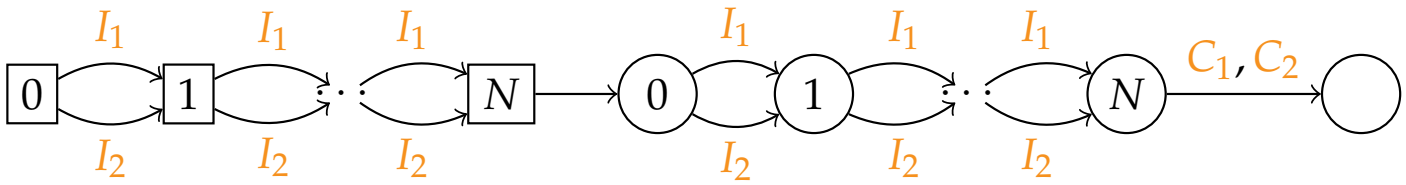
- Optimal strategy: value N; needs memory of size N
- Positional strategy: value $\frac{3}{2}N \approx N$

Games

In our setting: acyclic games (finite duration)

B-games: Player 0 (circle) wants to minimize the counter values.

Example:



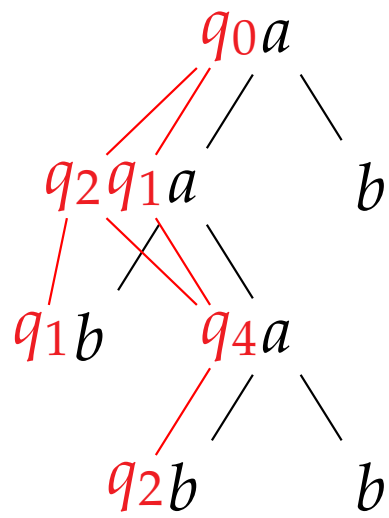
- Optimal strategy: value N; needs memory of size N
- Positional strategy: value $\frac{3}{2}N \approx N$

Results:

- For nested B-games positional strategies are sufficient (up to $\approx$) [Colcombet/L.'08]
- For B- and S-games strategies with finite memory are sufficient (up to $\approx$): finite memory reduction to nested B-games

From trees to words and back

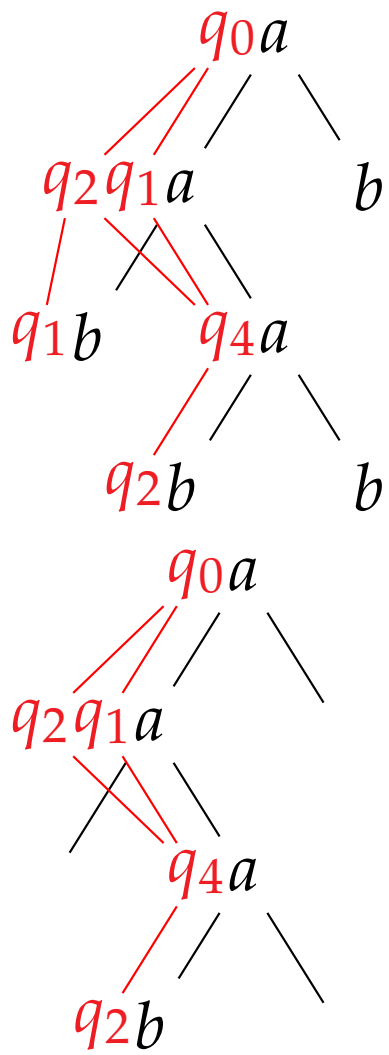
- Input for S-automaton: trees annotated with run paths
- S-automaton computes minimum cost over annotated paths
- Removing annotations by projection: max over all annotations



From trees to words and back

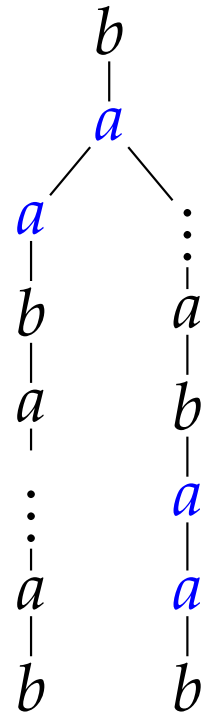
Construction of an S-automaton computing minimum cost over annotated paths

- On a single branch a B-automaton on words can easily compute the minimal cost of the path annotation
- Use results on words: this can also be done by an S-automaton (on words)
- Run this S-automaton over all branches of the the tree (only works for history deterministic automata)



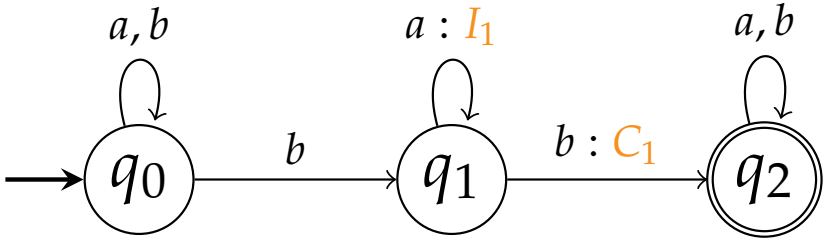
History determinism

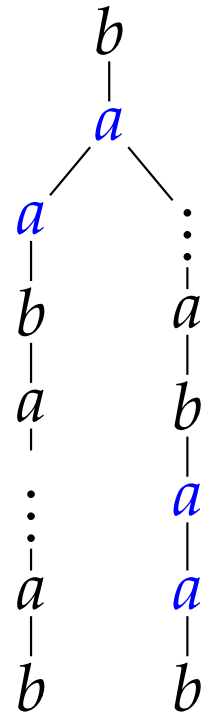
Example: Cost of a tree: among the shortest a-blocks on each path the maximum. Construct B-automaton computing this function.



History determinism

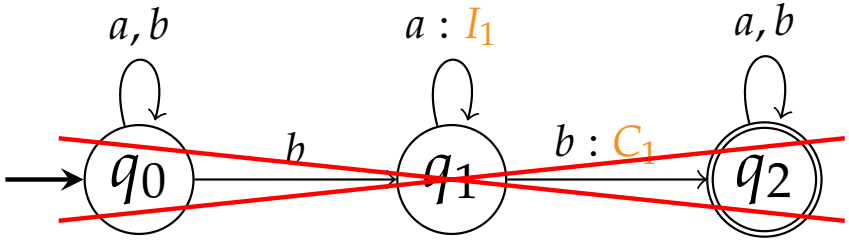
Example: Cost of a tree: among the shortest a-blocks on each path the maximum. Construct B-automaton computing this function.



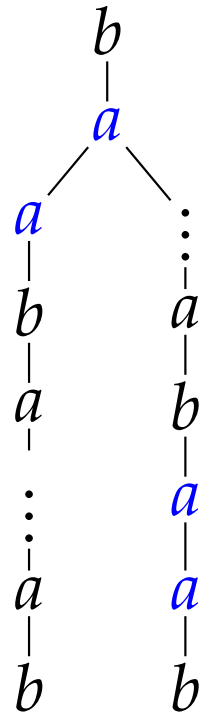


History determinism

Example: Cost of a tree: among the shortest a-blocks on each path the maximum. Construct B-automaton computing this function.

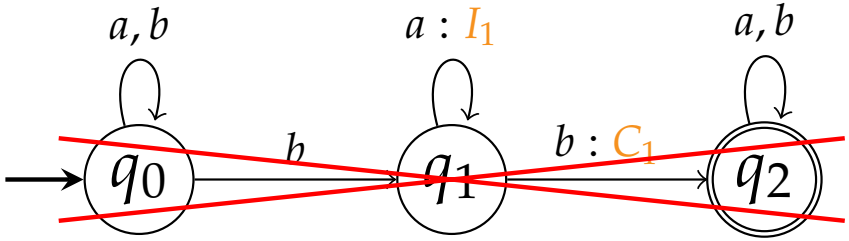


Decisions depend on the future, which is not unique in a tree.



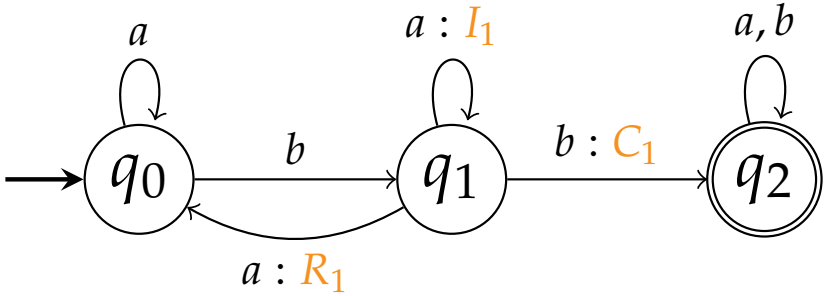
History determinism

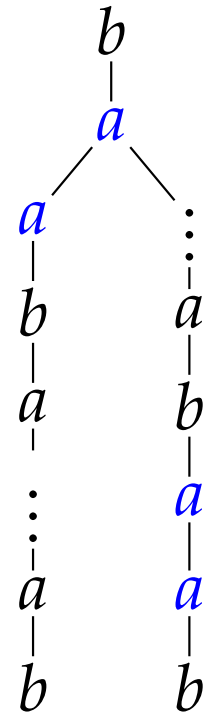
Example: Cost of a tree: among the shortest a-blocks on each path the maximum. Construct B-automaton computing this function.



Decisions depend on the future, which is not unique in a tree.

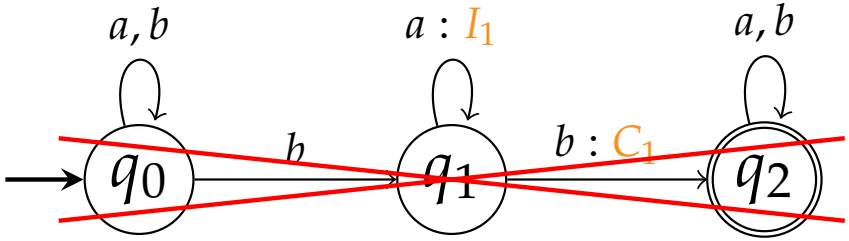
Better:

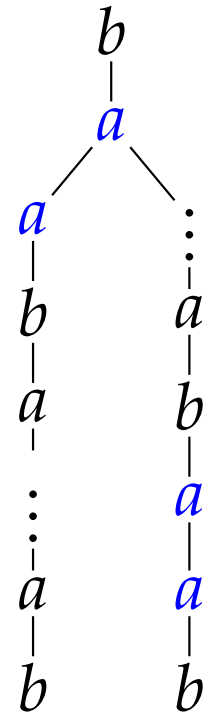




History determinism

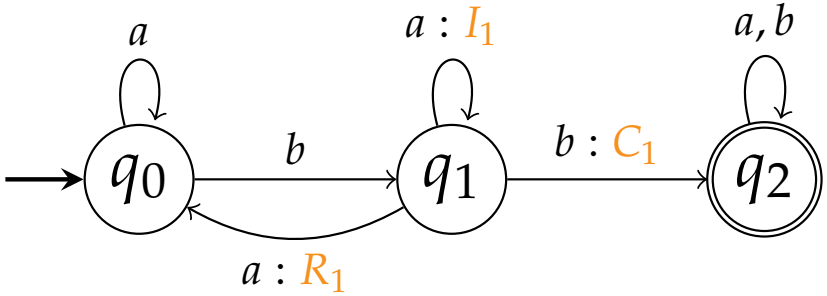
Example: Cost of a tree: among the shortest a-blocks on each path the maximum. Construct B-automaton computing this function.





Decisions depend on the future, which is not unique in a tree.

Better:



Theorem (Colcombet'09). Every B-/S-automaton on finite words can be made *history deterministic*.

Finite trees – results

1. The models of tree automata with any combination of the following properties all compute the same class of cost functions (modulo $\approx$):
   - B-/S-semantics
   - nested/non-nested counters
   - nondeterministic/alternating
   - with/without ε-transitions
2. The inclusion problem $f \preceq g$ is decidable, where f and g are given by cost tree automata.

Conclusion

- Research on boundedness problems for tree automata originally motivated by problem on infinite trees
- General theory for automata on finite trees to develop techniques and gain better understanding
  - Games
  - History determinism
- Main open problem on infinite trees:
  Show that in membership games for distance-parity automata finite memory strategies are sufficient!