Multiple tree automata

a new model of tree automata

Gwendal Collet (TU Wien), Julien David (LIPN)

Séminaire CALIN, 24 mars 2015

Outline

1. Introduction to automata: definitions and motivation
2. Description of the model: Multiple Tree Automata
3. Minimization
4. Closure properties
5. Yield of a MTA: Link with language theory

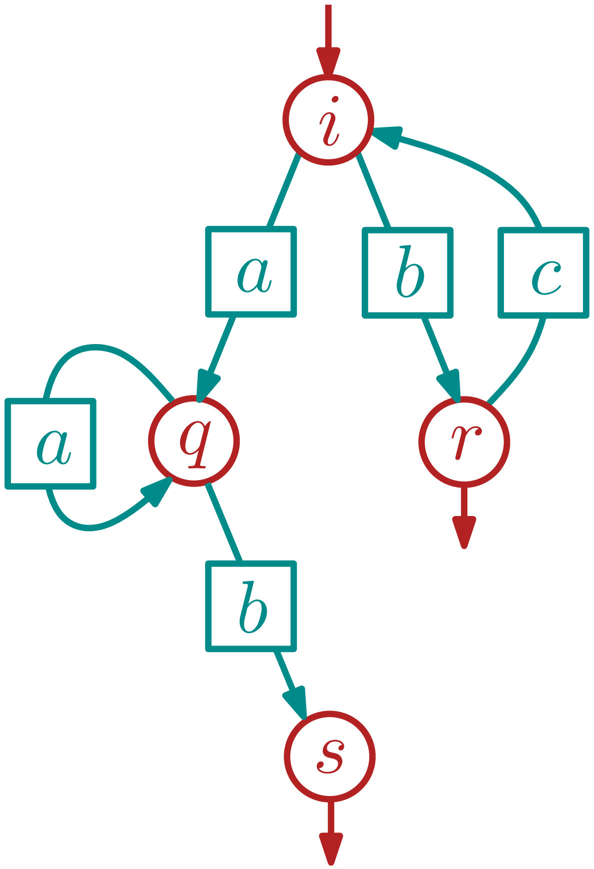
Introduction: Regular Word Automata

Finite alphabet: $a, b, c...$

Set of transitions: $\Delta \subset Q \times \Sigma \times Q$

$$\mathcal{A} = (\Sigma, Q, I, F, \delta)$$

Finite set of states: initial, final...



$i \in I, r, s \in F$

$(i, b, r), (q, a, q), \ldots \in \Delta$

$\mathcal{L}_{\mathcal{A}} = (bc)^{\star}(1 + a^{+}b)$

e.g.: $bcaaab \in \mathcal{L}_{\mathcal{A}}$

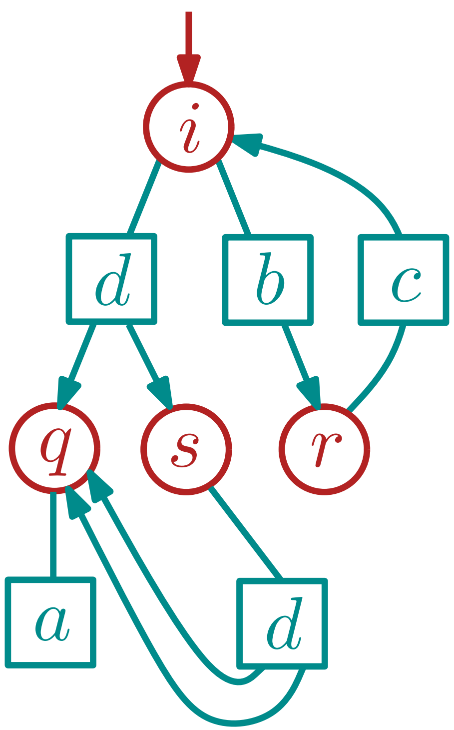
Introduction: Regular Tree Automata

Finite ranked alphabet: $a(0), b(1), c(1), d(2)\ldots$

$$\mathcal{A} = (\Sigma = \cup_{k\geq 0}\Sigma_k, Q, I, \Delta)$$

Set of transitions: $\Delta \subset \cup_{k\geq 0} Q \times \Sigma_k \times Q^k$

Finite set of states: initial, ~~final~~...

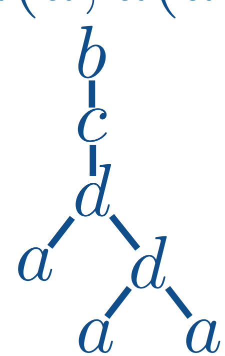


$a \in \Sigma_0$ (leaf), $b, c \in \Sigma_1, d \in \Sigma_2$

$i \in I$

$(i, b, r), (q, a, \epsilon), (s, d, (q, q)), \ldots \in \Delta$

$$\mathcal{L}_\mathcal{A} = (\underbrace{b(c(\ldots b(c(}_{\star}d(a, d(a, a)))))))$$

e.g.: 

$\in \mathcal{L}_\mathcal{A}$

Introduction: Regular Tree Automata

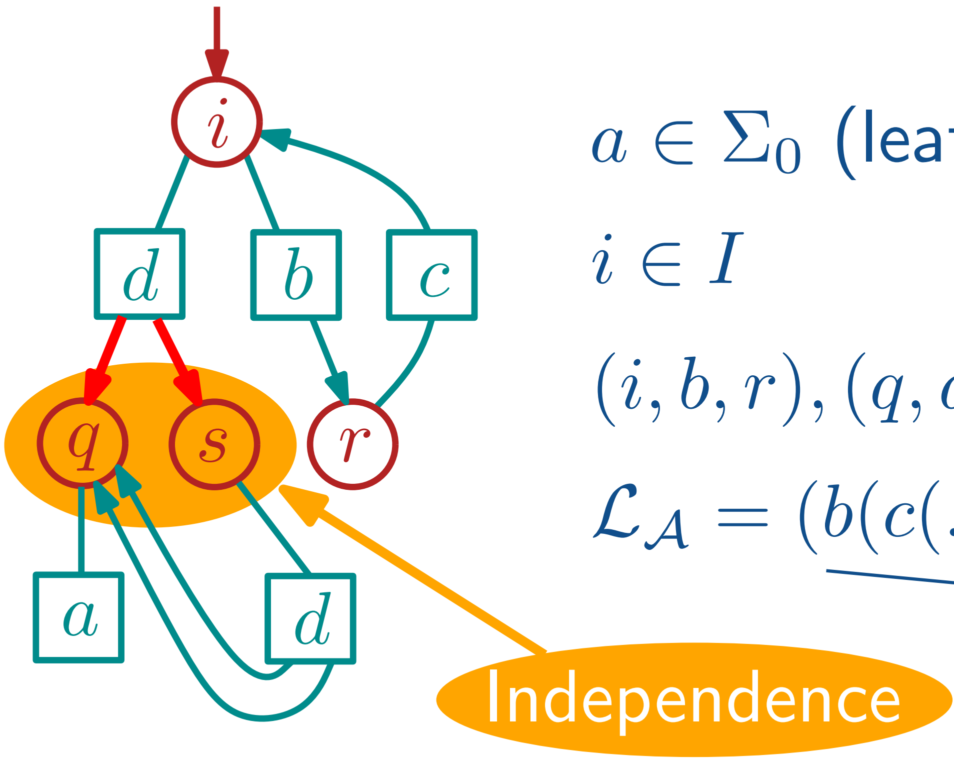
Finite ranked alphabet: $a(0), b(1), c(1), d(2)\ldots$

$$\mathcal{A} = (\Sigma = \cup_{k\geq 0}\Sigma_k, Q, I, \Delta)$$

Set of transitions:

$\Delta \subset \cup_{k\geq 0} Q \times \Sigma_k \times Q^k$

Finite set of states: initial, ~~final~~...

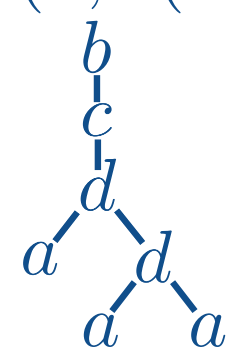


$a \in \Sigma_0$ (leaf), $b, c \in \Sigma_1, d \in \Sigma_2$

$i \in I$

$(i, b, r), (q, a, \epsilon), (s, d, (q, q)), \ldots \in \Delta$

$\mathcal{L}_{\mathcal{A}} = (\underbrace{b(c(\ldots b(c(}_{\star}d(a, d(a, a)))))))$

e.g.:  $\in \mathcal{L}_{\mathcal{A}}$

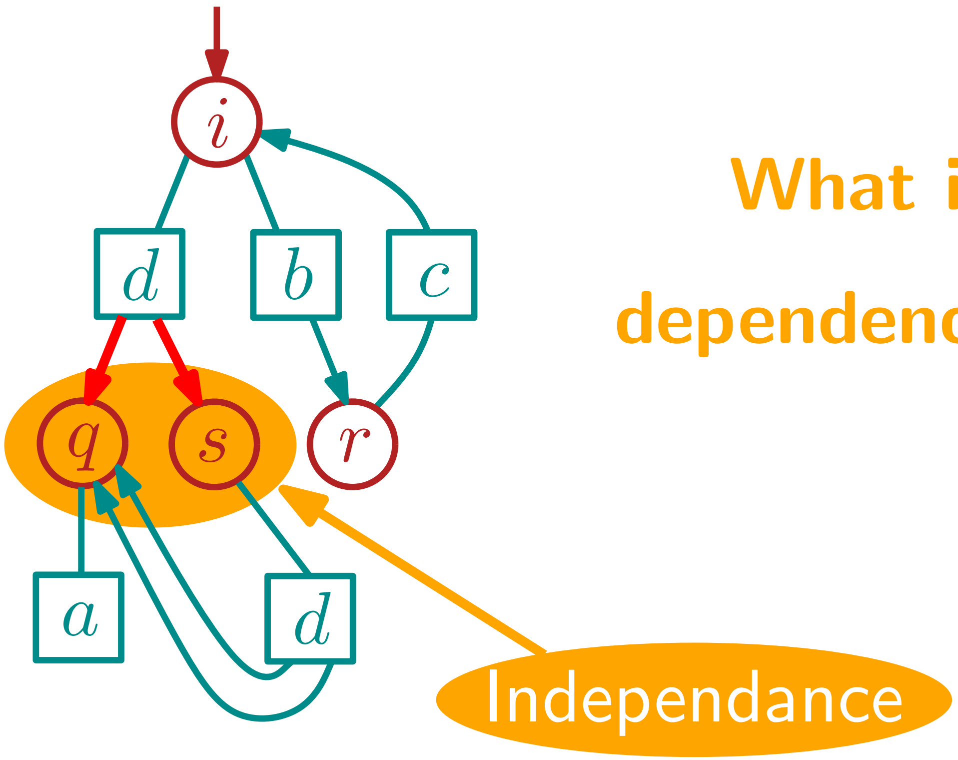
Introduction: Regular Tree Automata

Finite ranked alphabet: $a(0), b(1), c(1), d(2)\ldots$

$$\mathcal{A} = (\Sigma = \cup_{k\geq 0}\Sigma_k, Q, I, \Delta)$$

Set of transitions: $\Delta \subset \cup_{k\geq 0} Q \times \Sigma_k \times Q^k$

Finite set of states: initial, ~~final~~...

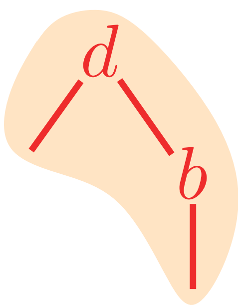


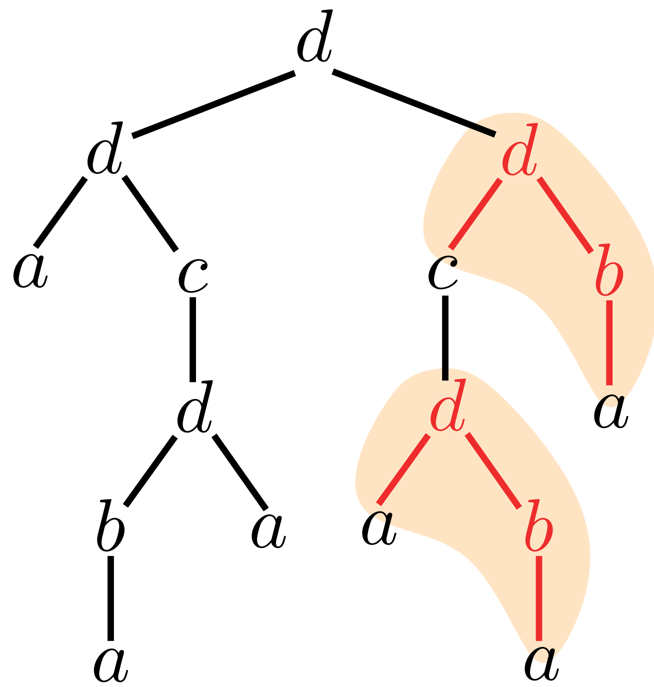
What if we could handle dependencies between children?

Introduction: Motivation

Random sampling of trees
controlling the number of occurrences of a given **pattern**

Pattern



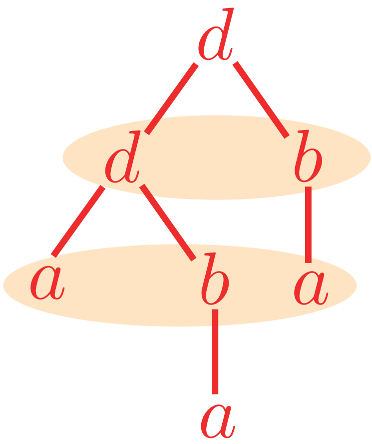


2 occurrences

Introduction: Motivation

Random sampling of trees
controlling the number of occurrences of a given **pattern**

Pattern



When reading the tree top-down:

Dependencies between nodes at a same height

Idea (C., David, Jacquot 2014):

- Use refined tree automata which count occurrences of a given pattern $\rightarrow$ need to handle dependencies
- Translate the associated tree grammar into a system of equations on generating series
- Design a bivariate Boltzmann sampler with the GS

Multiple Tree Automata (MTA)

Finite ranked alphabet: $a(0), b(1), c(1), d(2)\ldots$

$$\mathcal{A} = (\Sigma = \cup_{k\geq 0}\Sigma_k, Q = \cup_{\ell\geq 1} Q_\ell, I, \Delta)$$

Finite ranked set of states

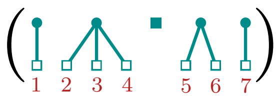
Initial states $\in Q_1$

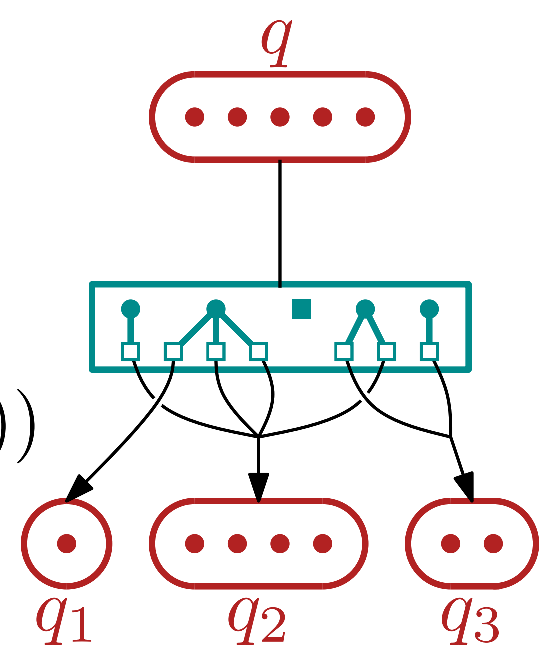
Set of transitions: $\Delta \subset \cup_{\ell\geq 1} Q_\ell \times \Sigma^\ell \times Part \times Q^\star$

$$(q, (a_1, \ldots, a_\ell), P = (p_1, \ldots, p_r), (q_1, \ldots, q_r))$$

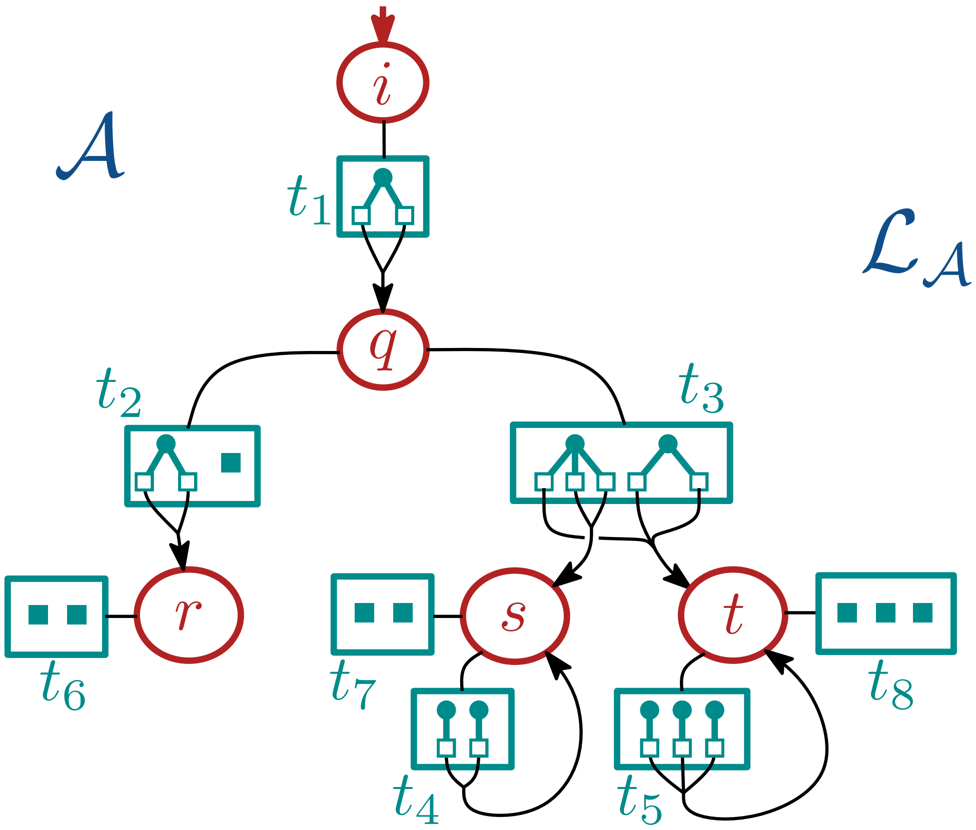
such that: $|P| = \sum_{i=1}^{\ell} rank(a_i)$

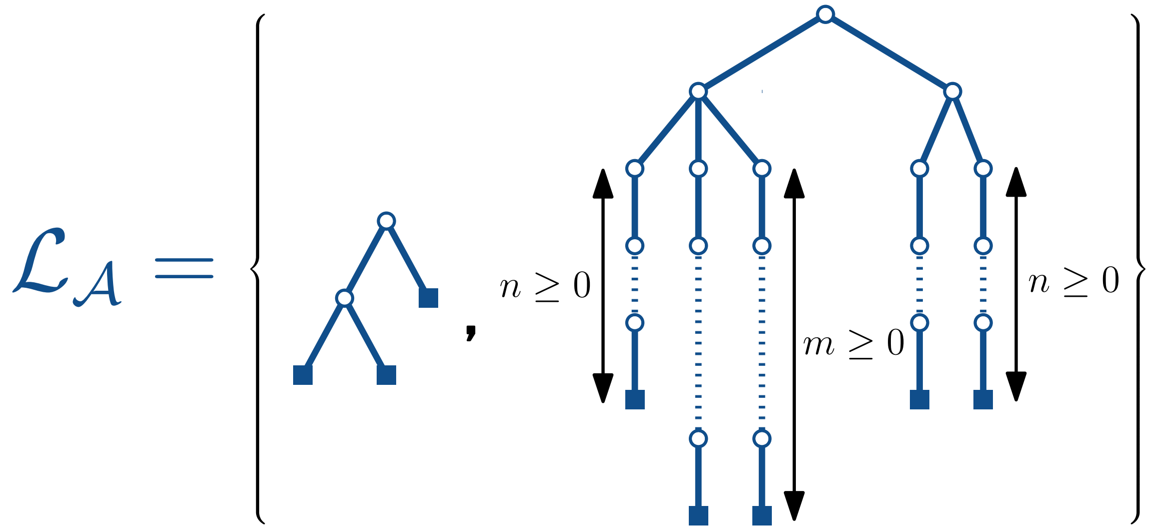
$$\forall 1 \leq j \leq r, rank(q_j) = |p_j|$$

$(q, ($  $), \{\{2\}, \{1,3,4,6\}, \{5,7\}\}, (q_1, q_2, q_3))$

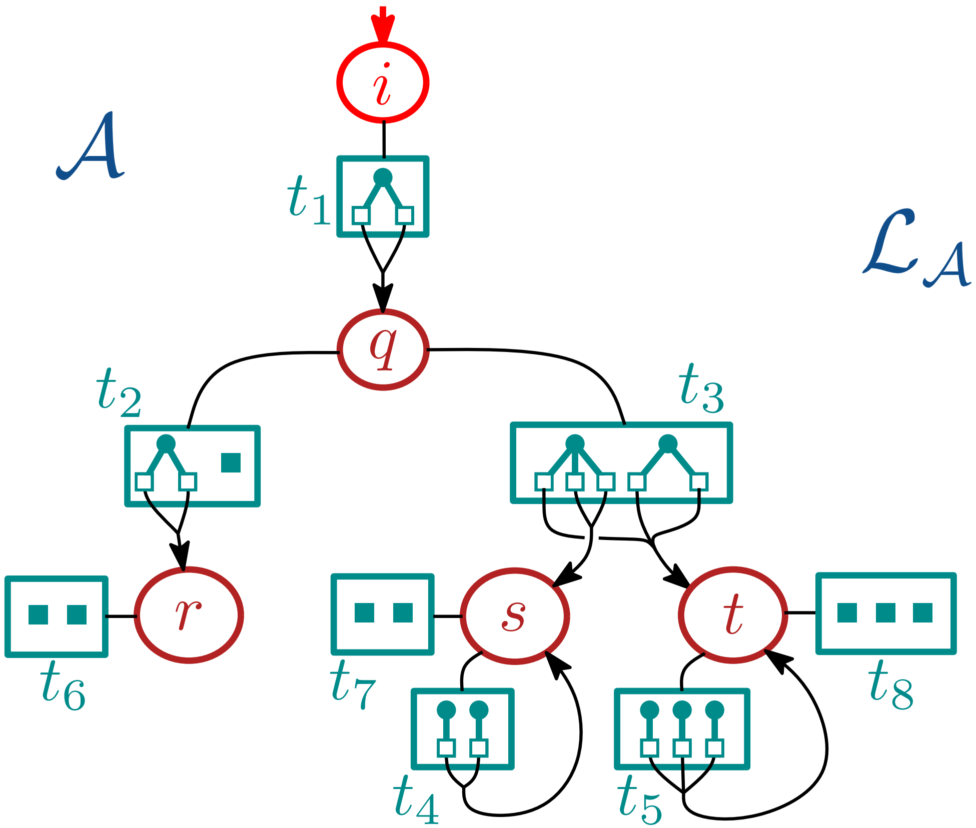


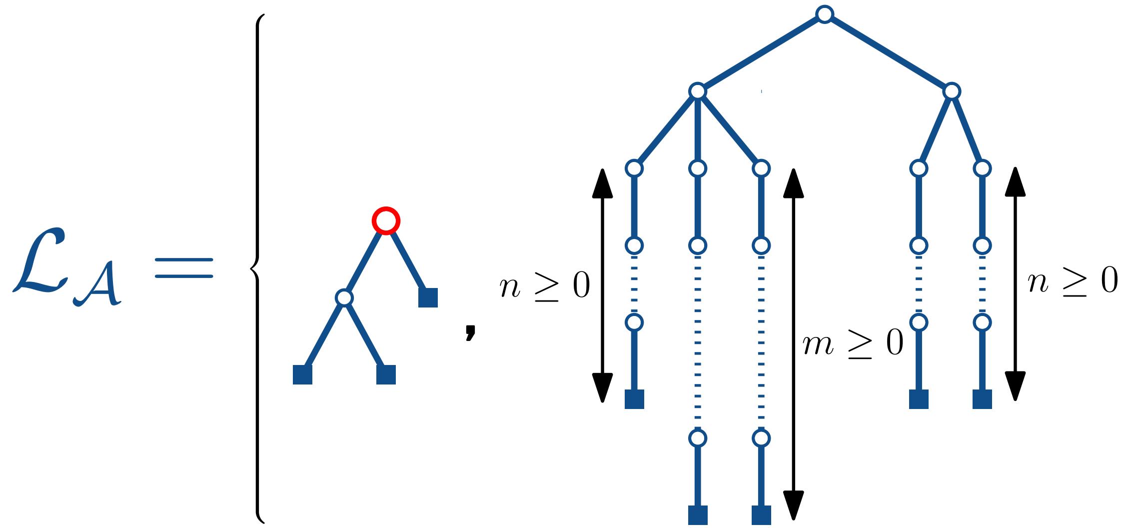
Multiple Tree Automata (MTA)



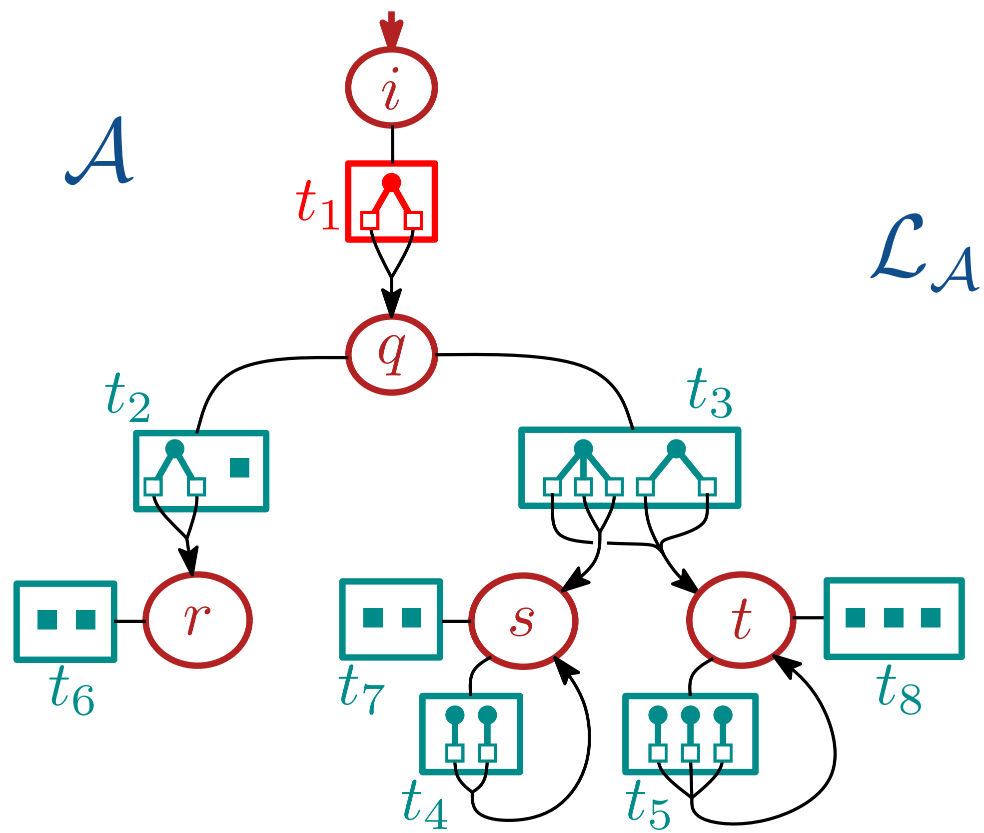


Multiple Tree Automata (MTA)

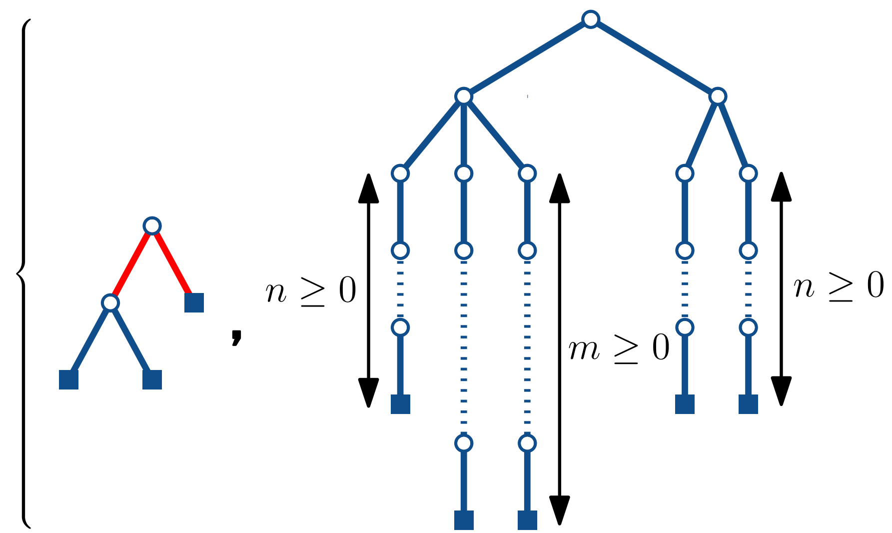




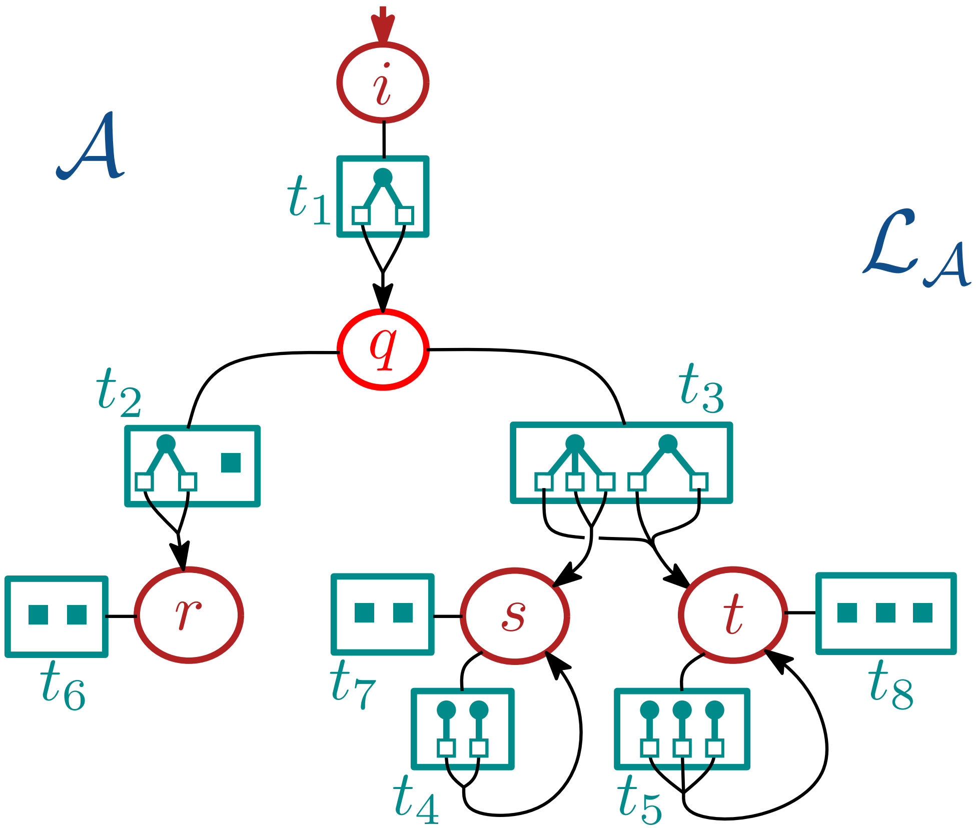
Multiple Tree Automata (MTA)



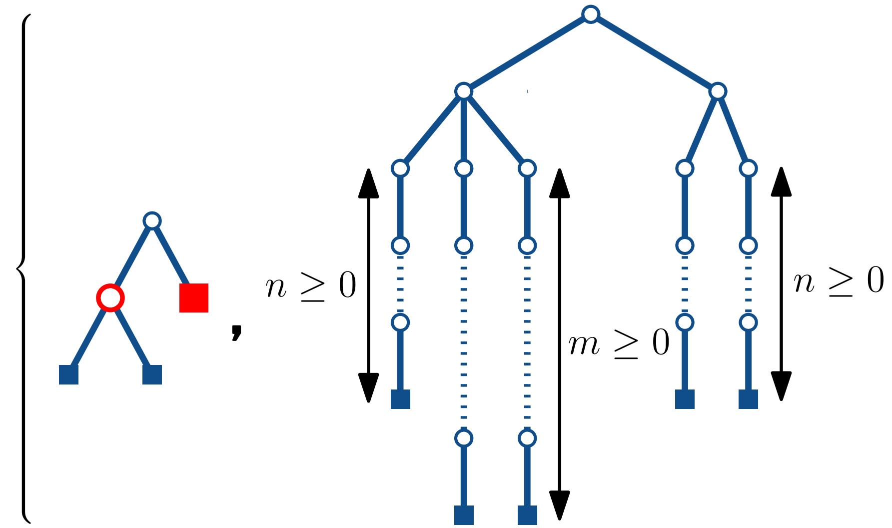
$\mathcal{L}_\mathcal{A} =$



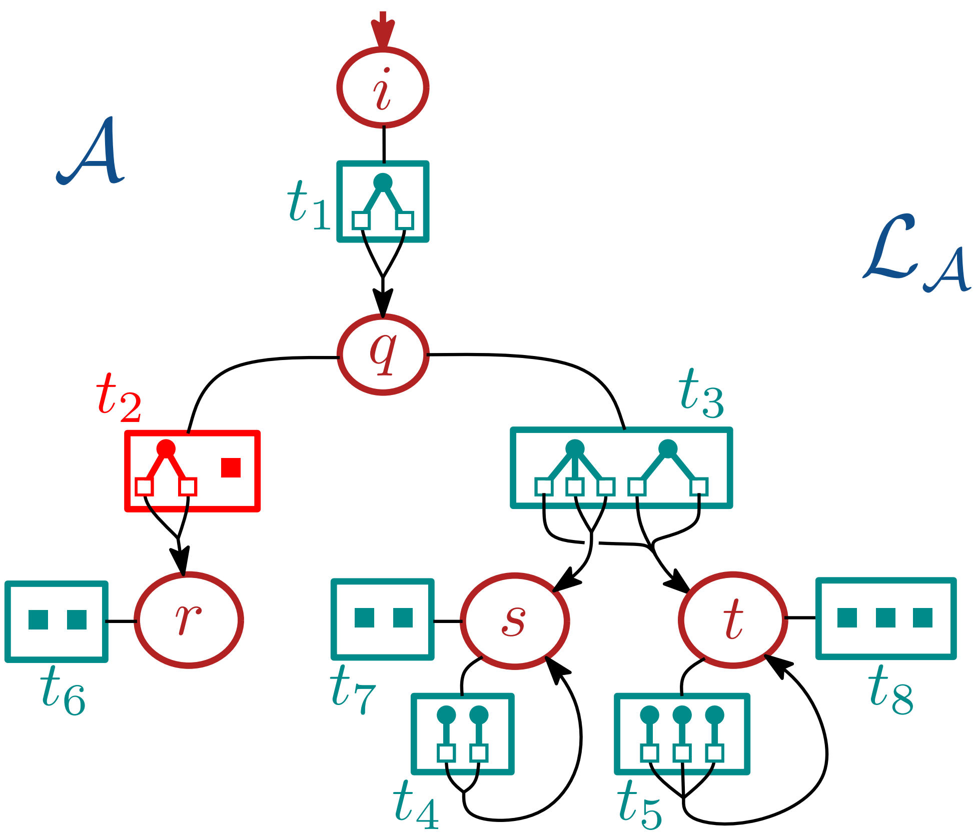
Multiple Tree Automata (MTA)



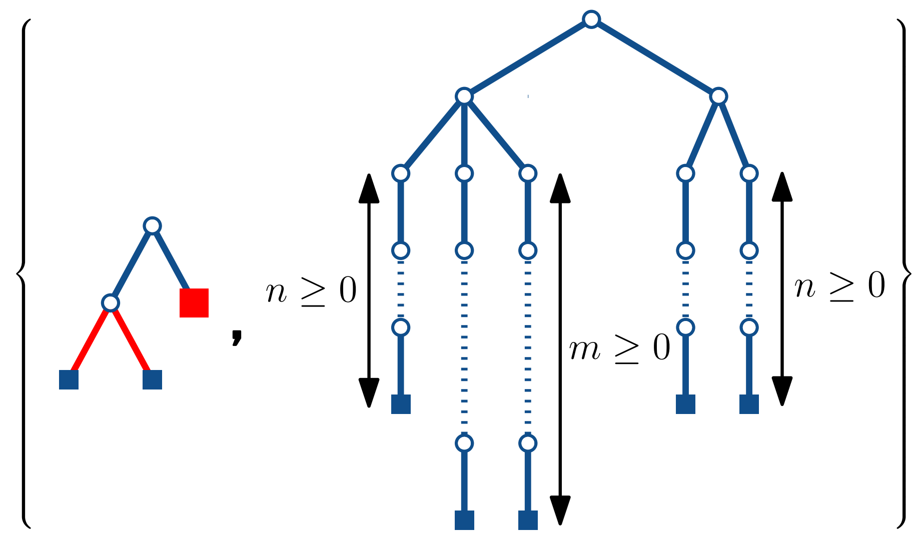
$\mathcal{L}_{\mathcal{A}} =$



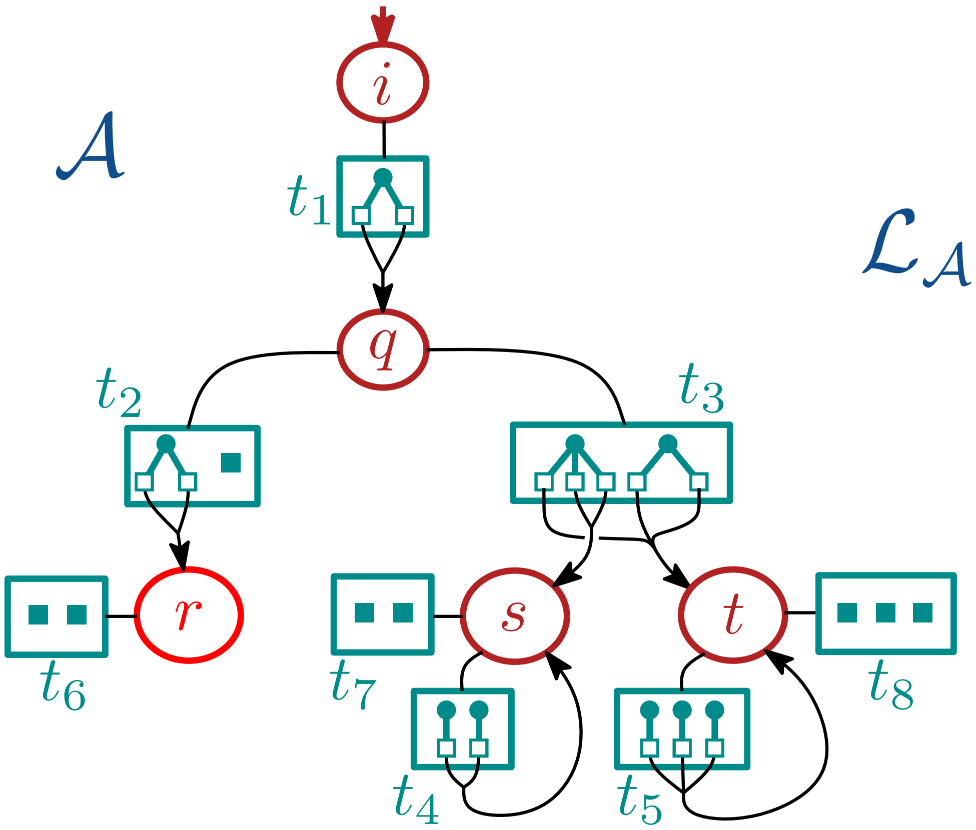
Multiple Tree Automata (MTA)



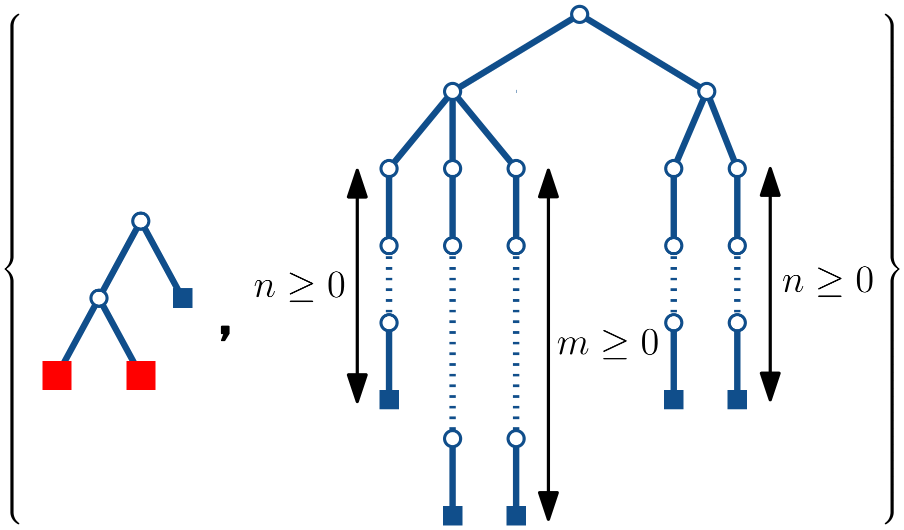
$\mathcal{L}_\mathcal{A} =$



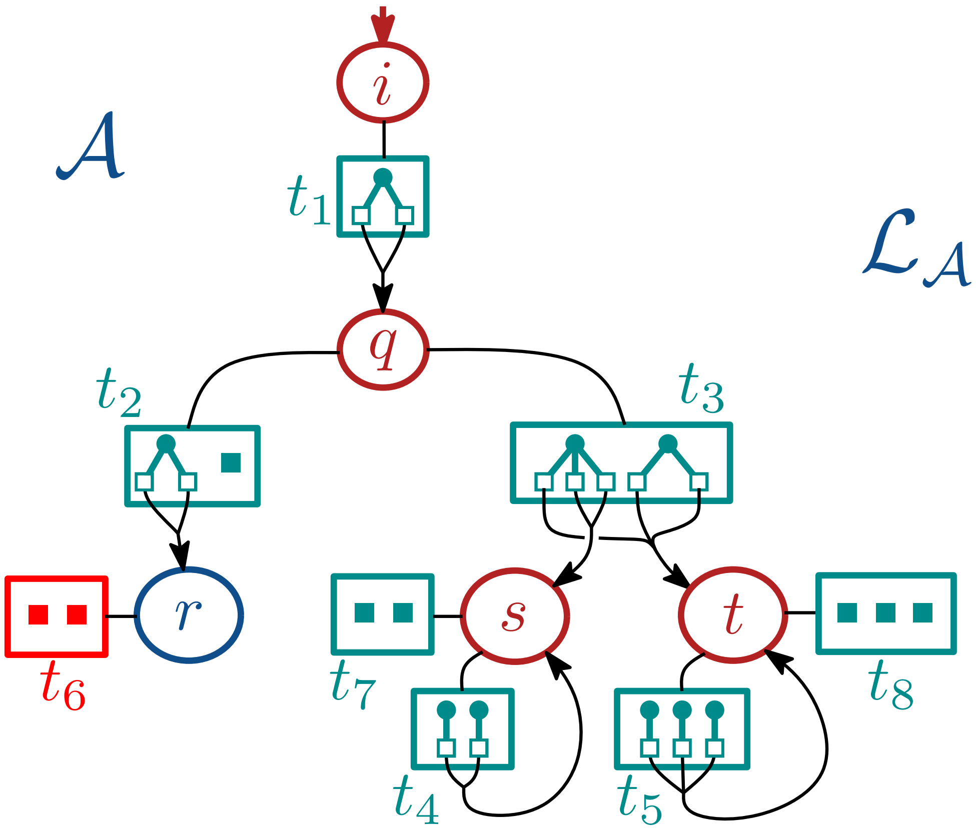
Multiple Tree Automata (MTA)



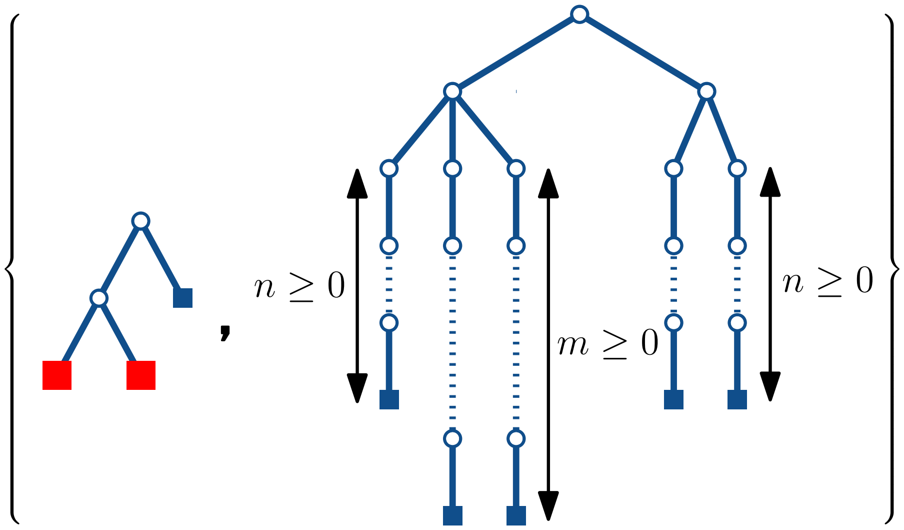
$\mathcal{L}_{\mathcal{A}} =$



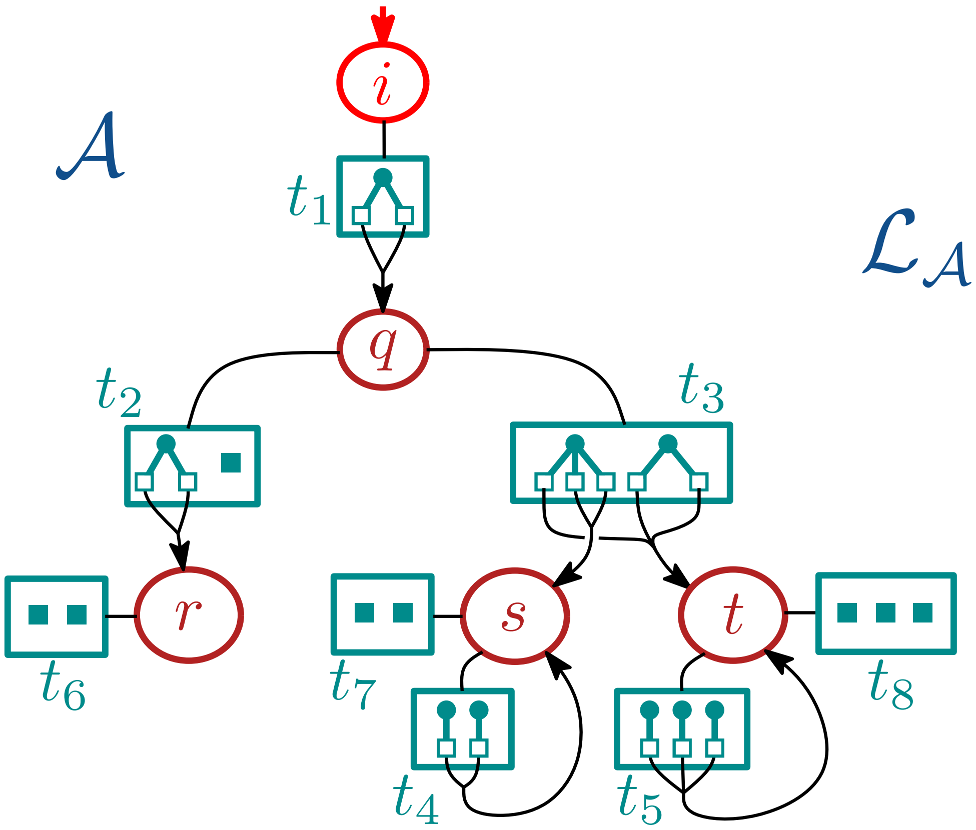
Multiple Tree Automata (MTA)

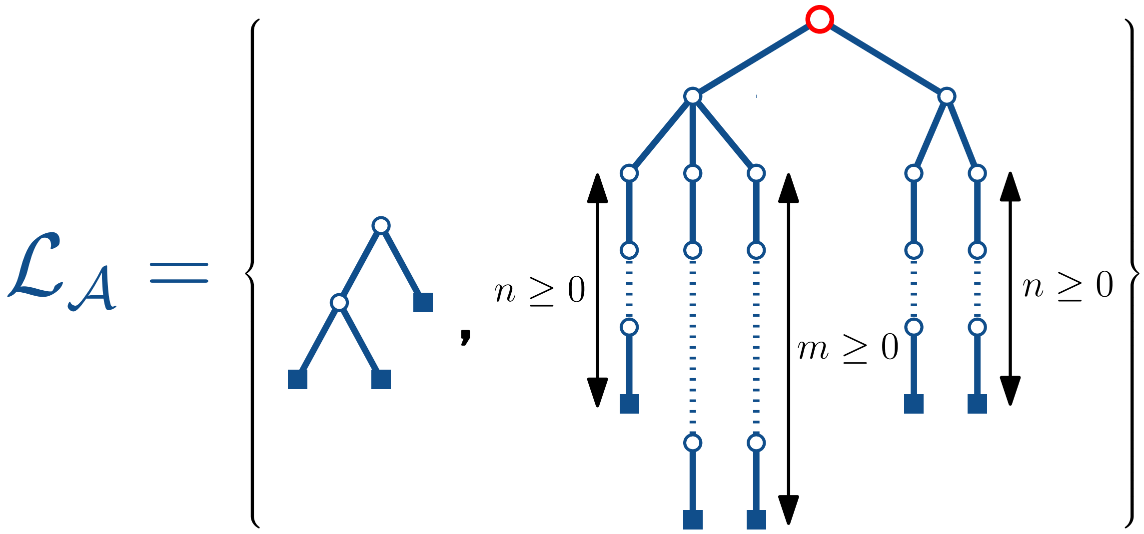


$\mathcal{L}_\mathcal{A} =$

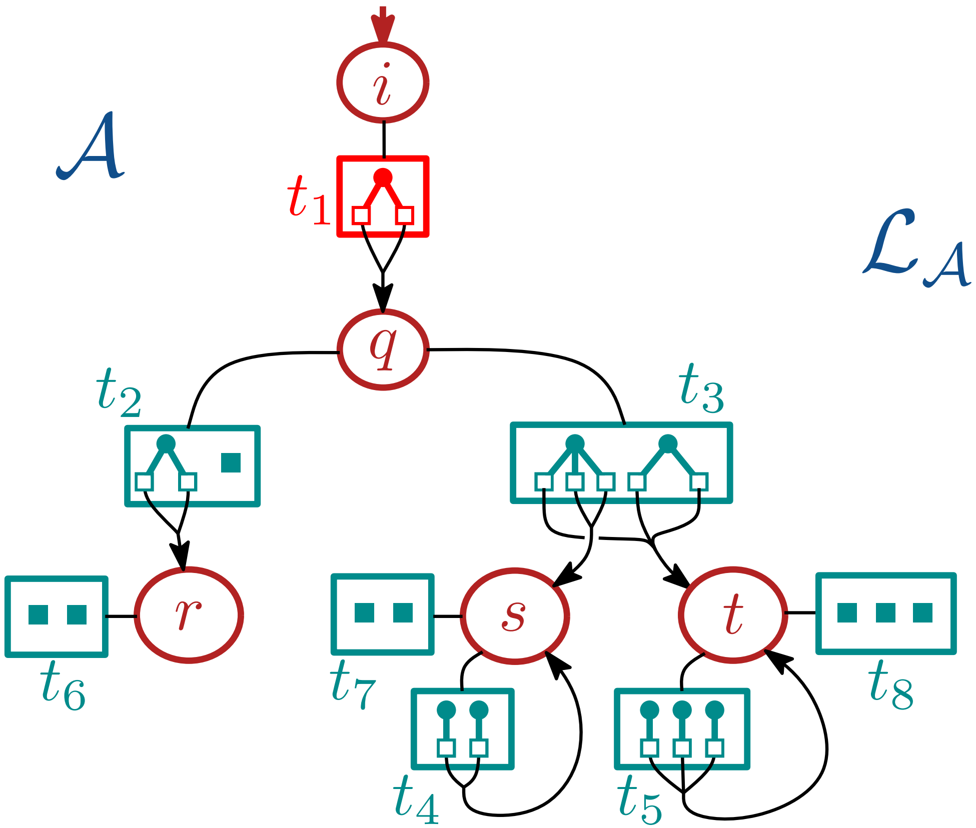


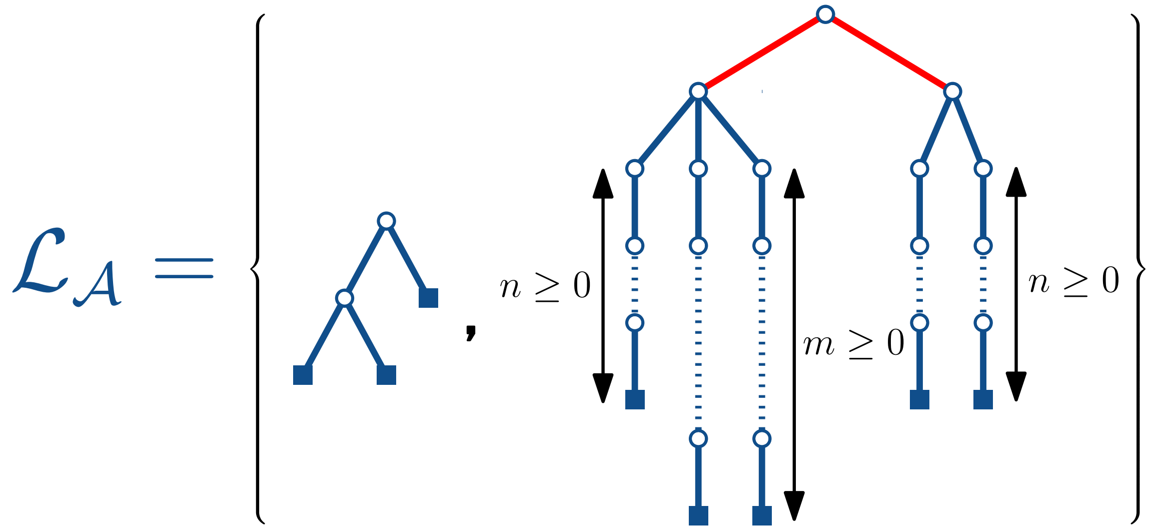
Multiple Tree Automata (MTA)



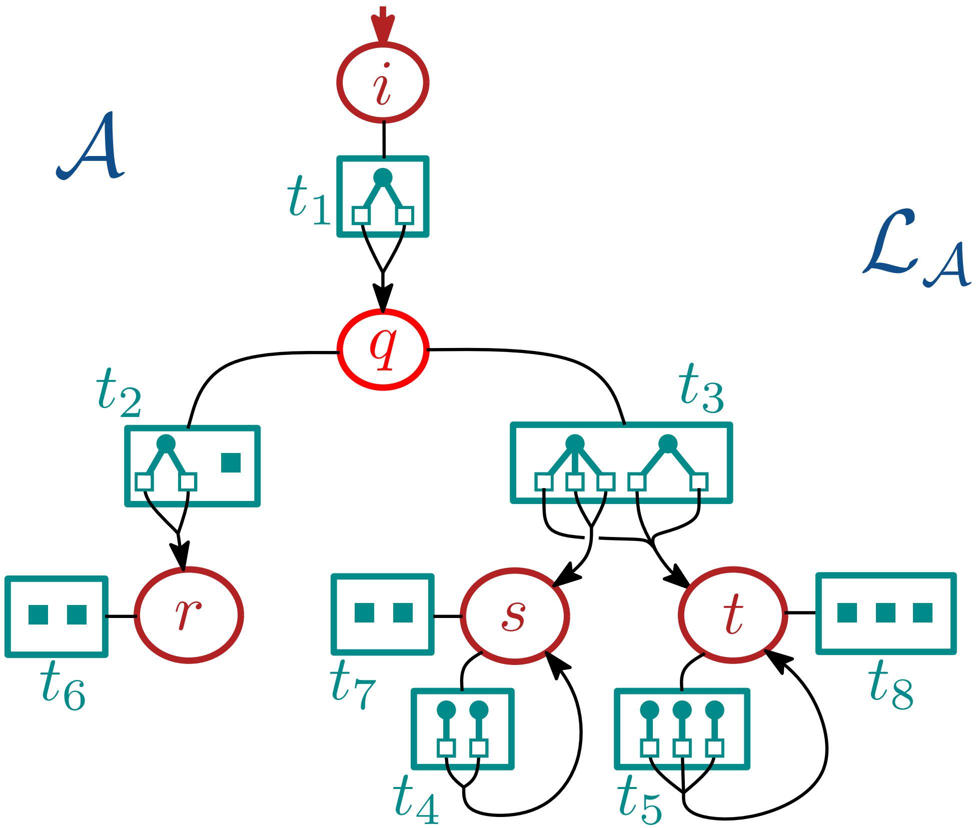


Multiple Tree Automata (MTA)

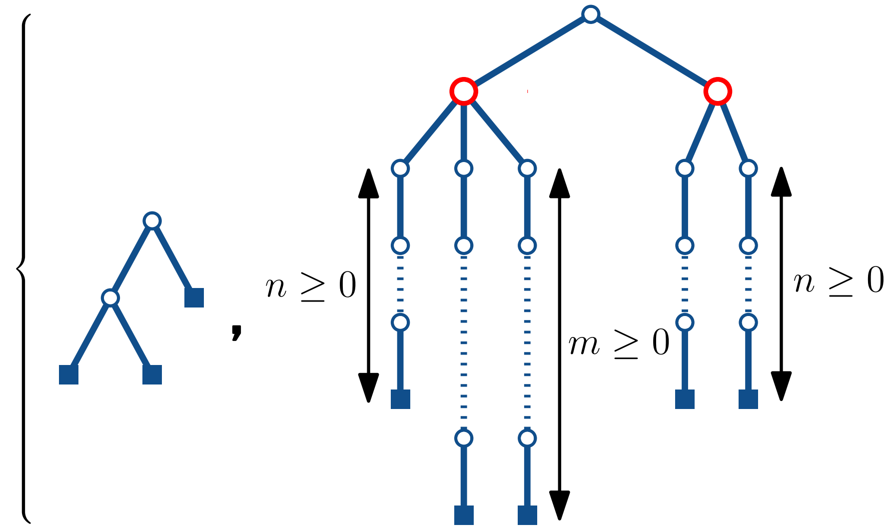




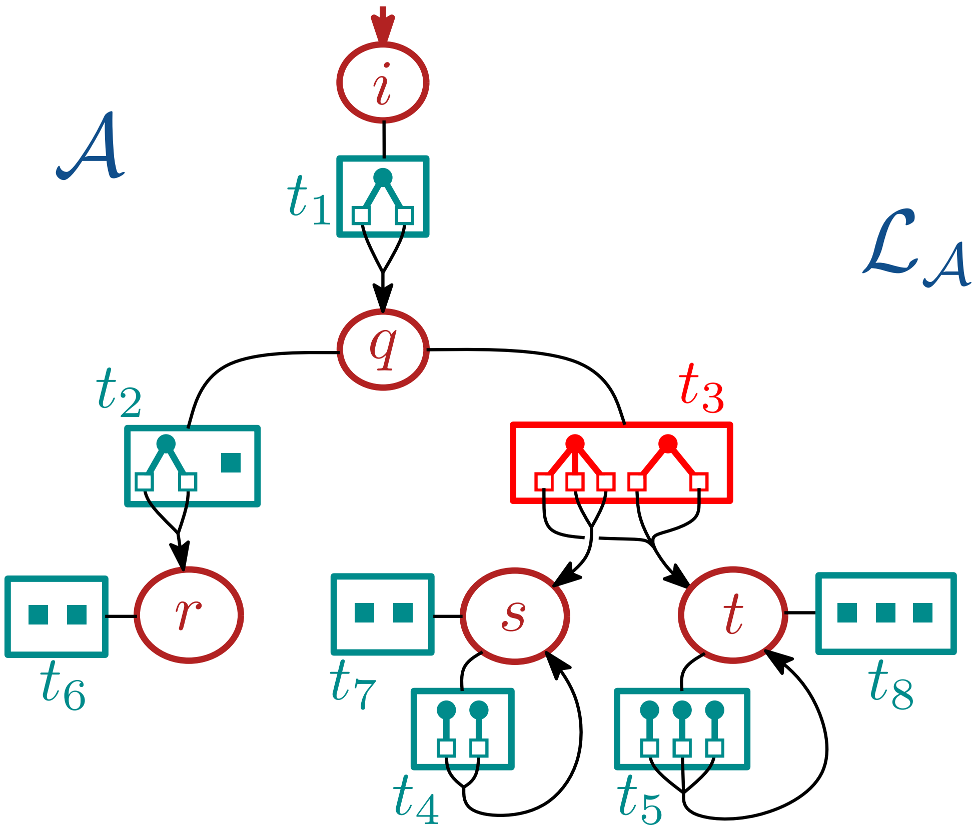
Multiple Tree Automata (MTA)

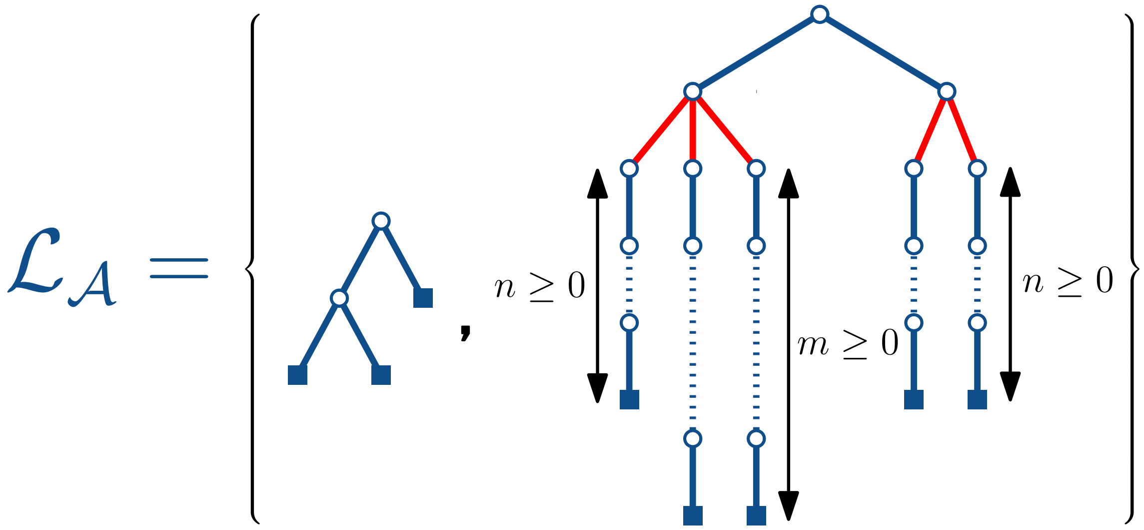


$\mathcal{L}_{\mathcal{A}} =$

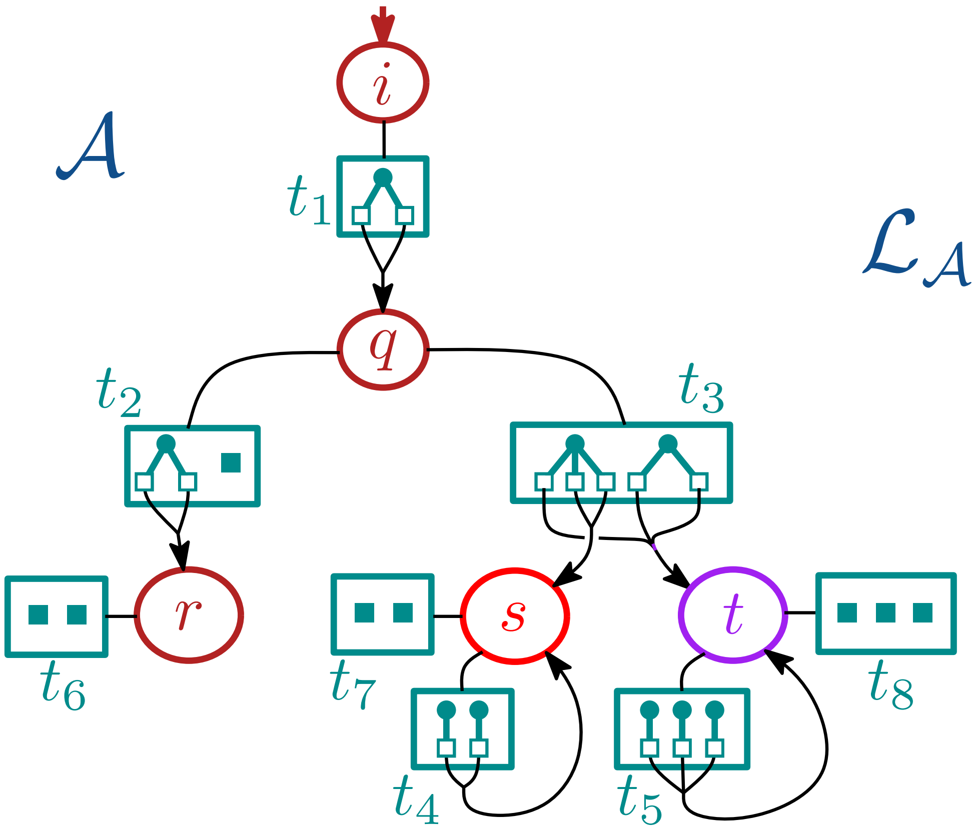


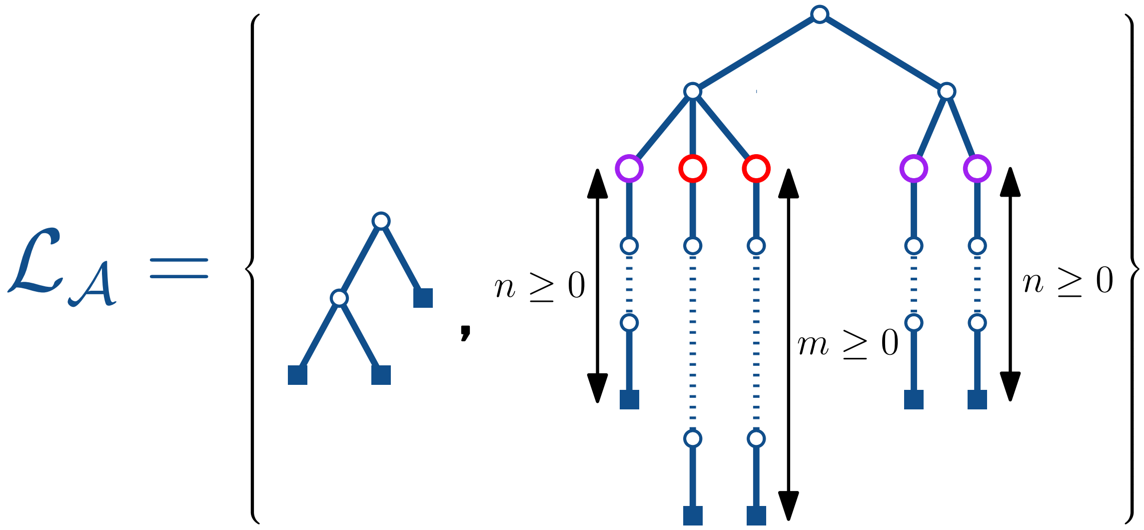
Multiple Tree Automata (MTA)



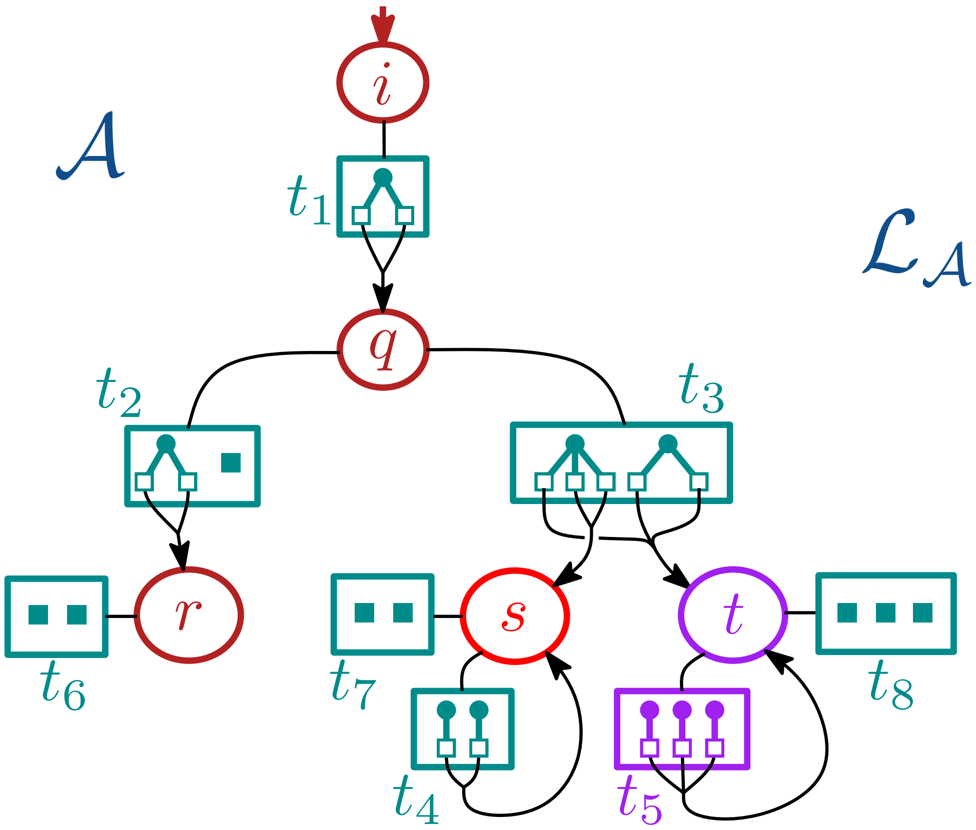


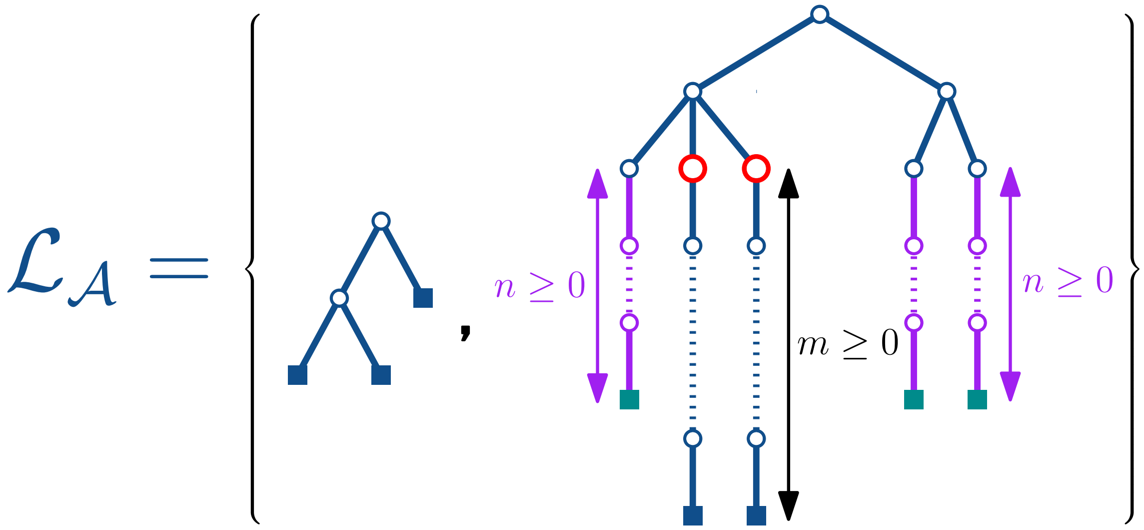
Multiple Tree Automata (MTA)



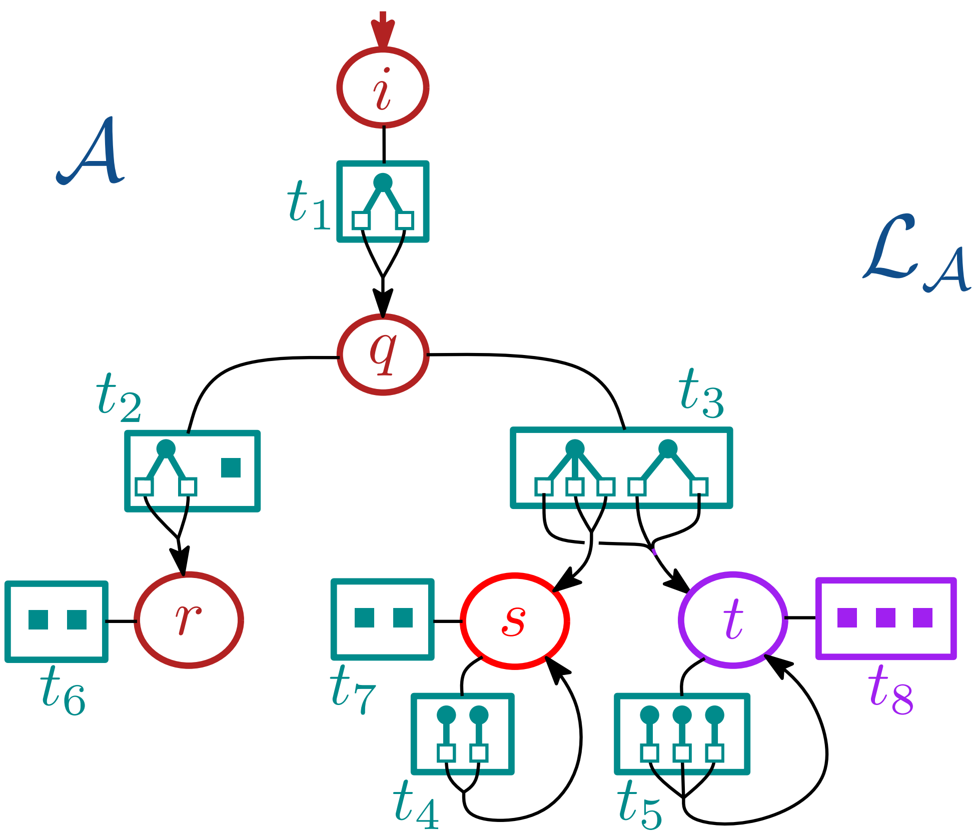


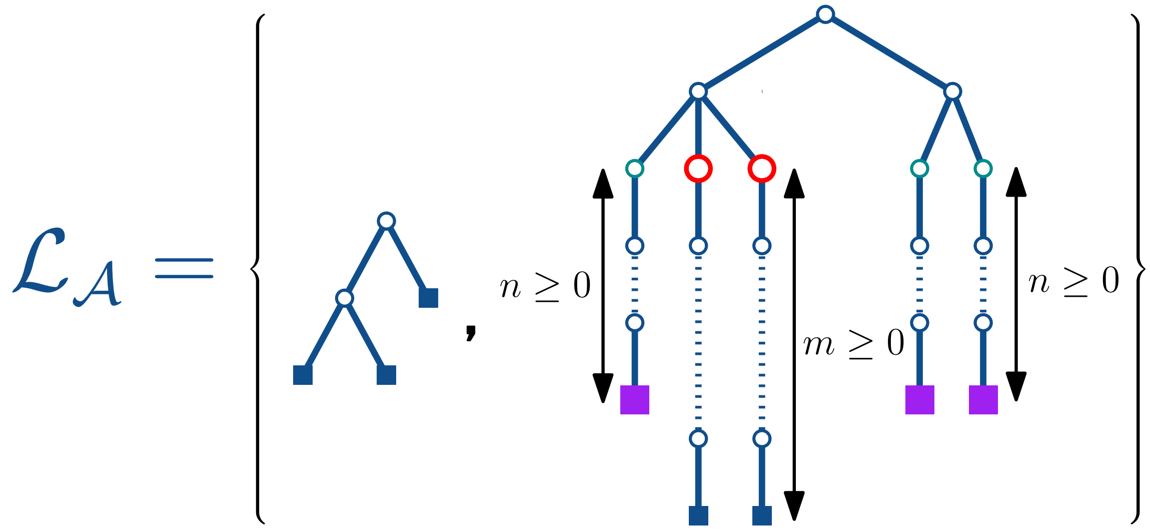
Multiple Tree Automata (MTA)



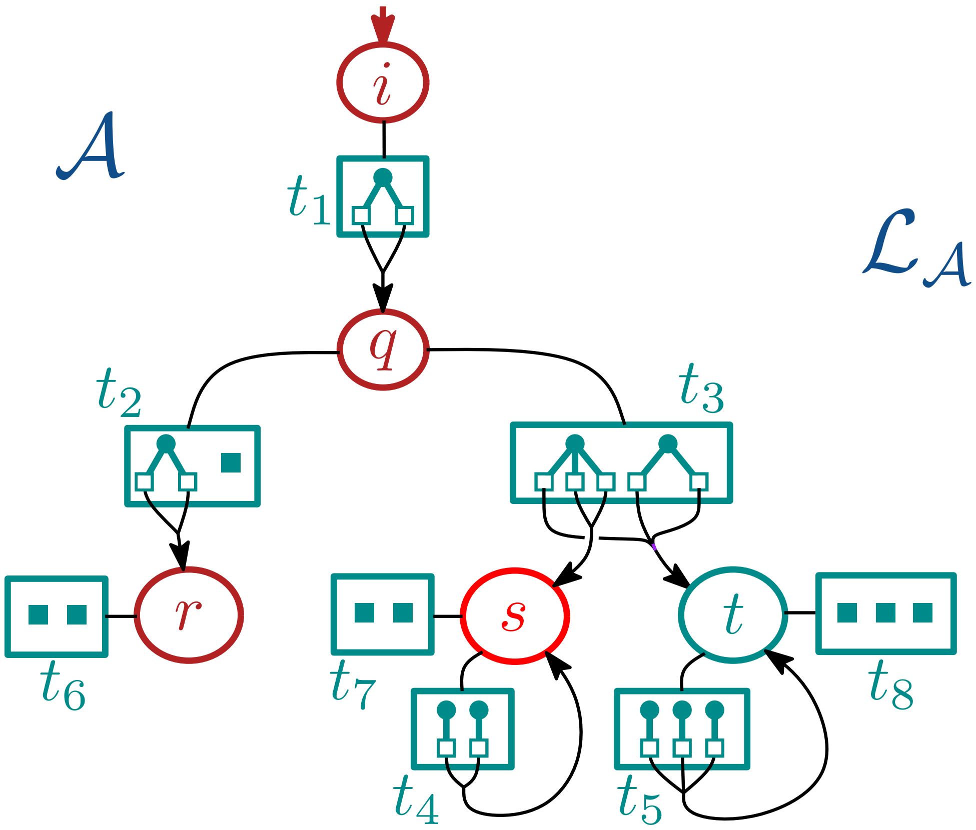


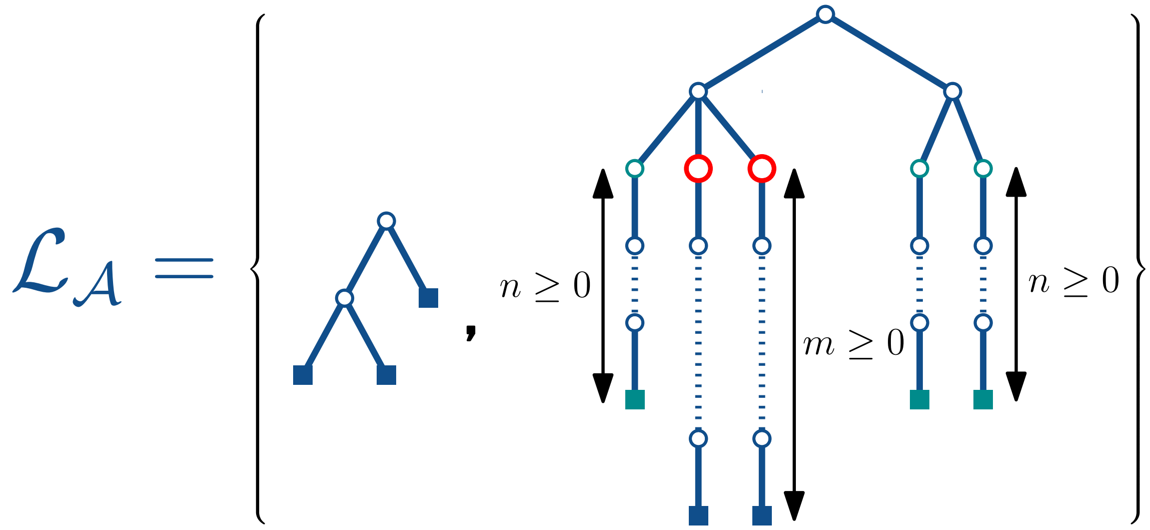
Multiple Tree Automata (MTA)



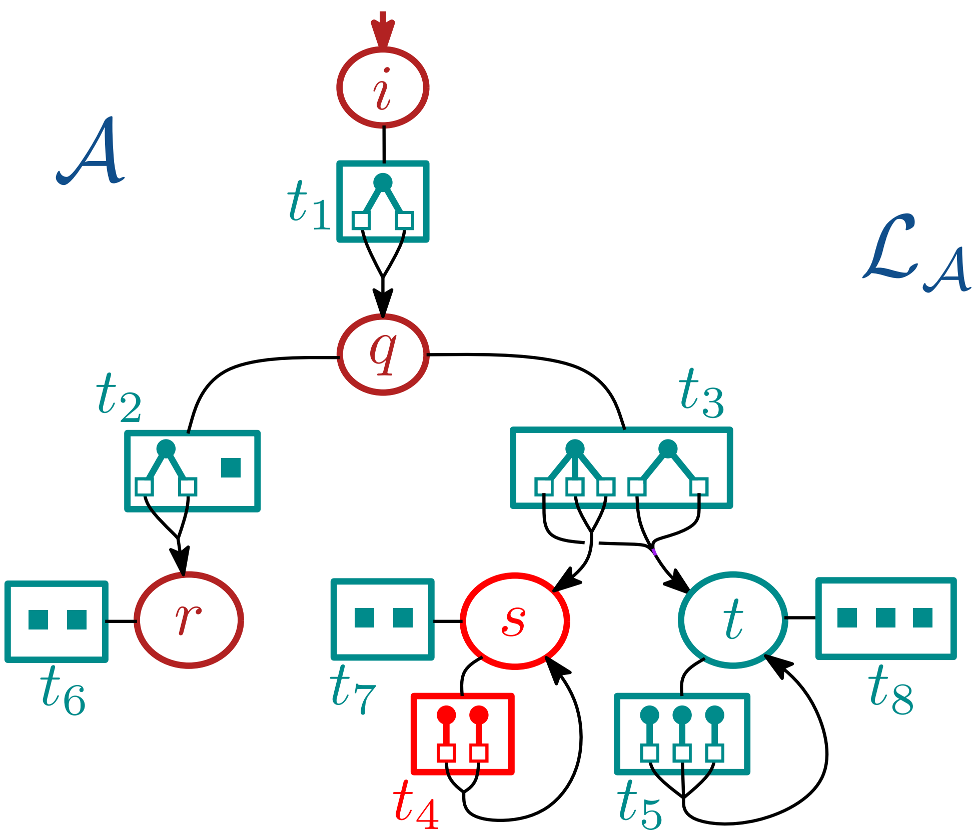


Multiple Tree Automata (MTA)

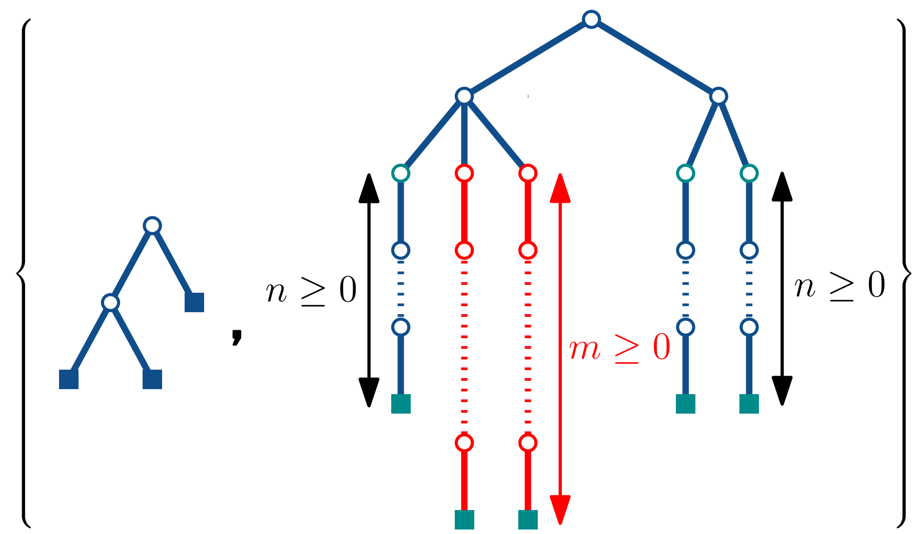




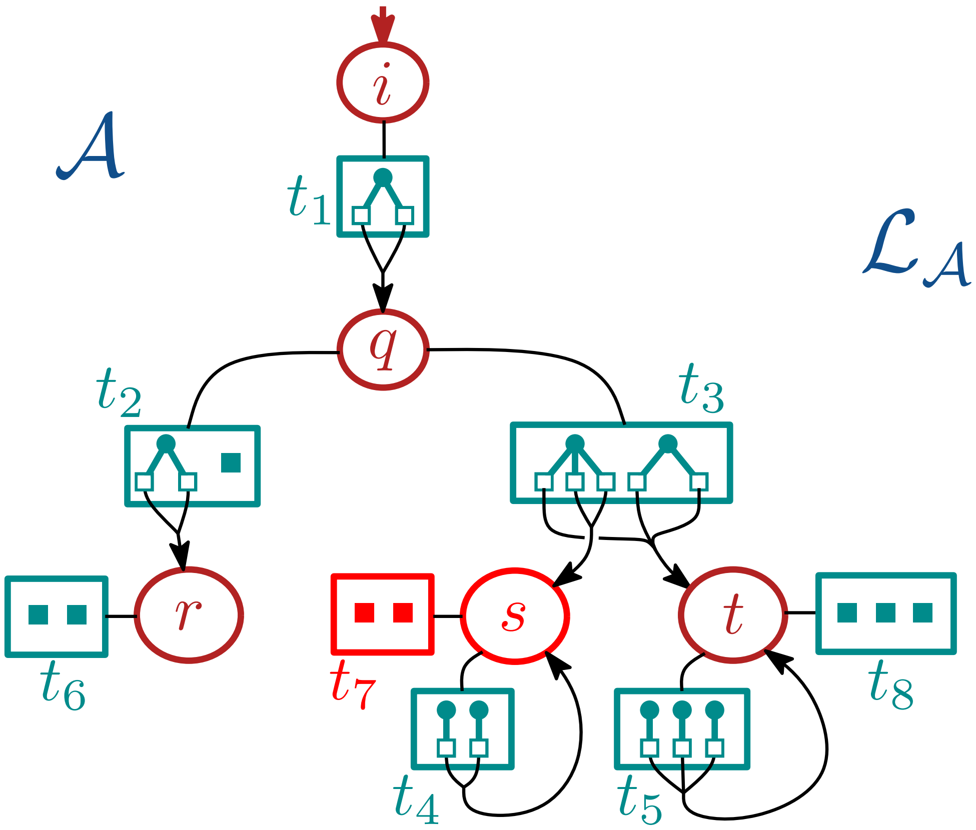
Multiple Tree Automata (MTA)

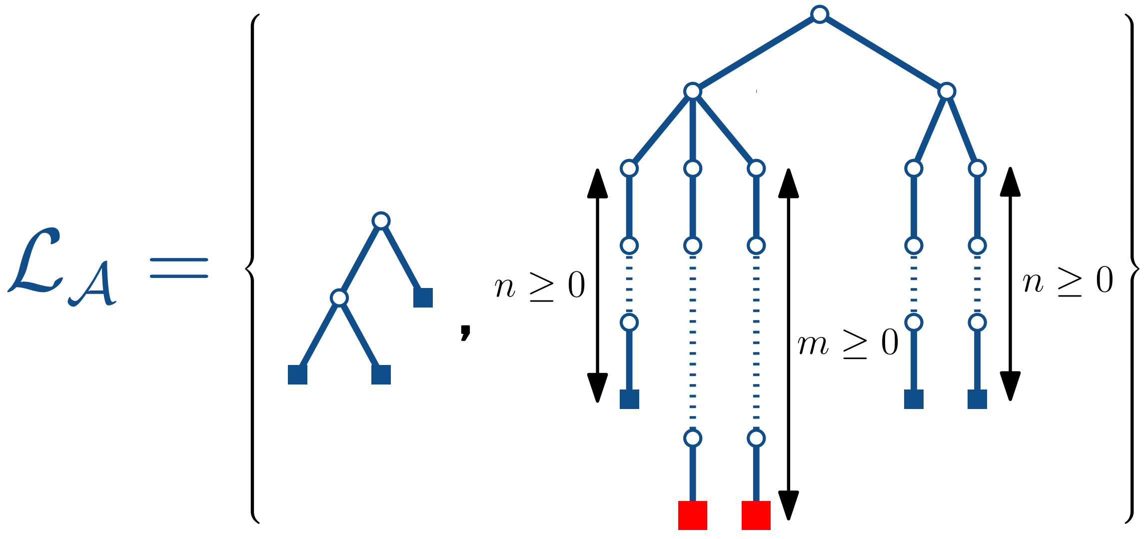


$\mathcal{L}_{\mathcal{A}} =$

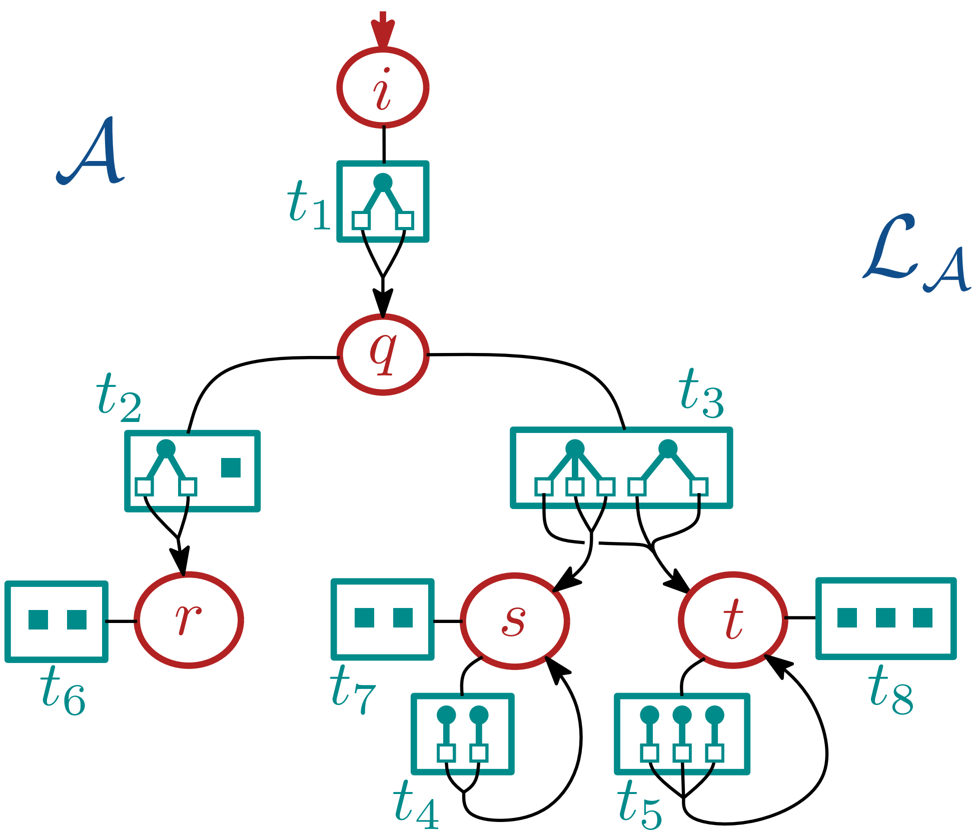


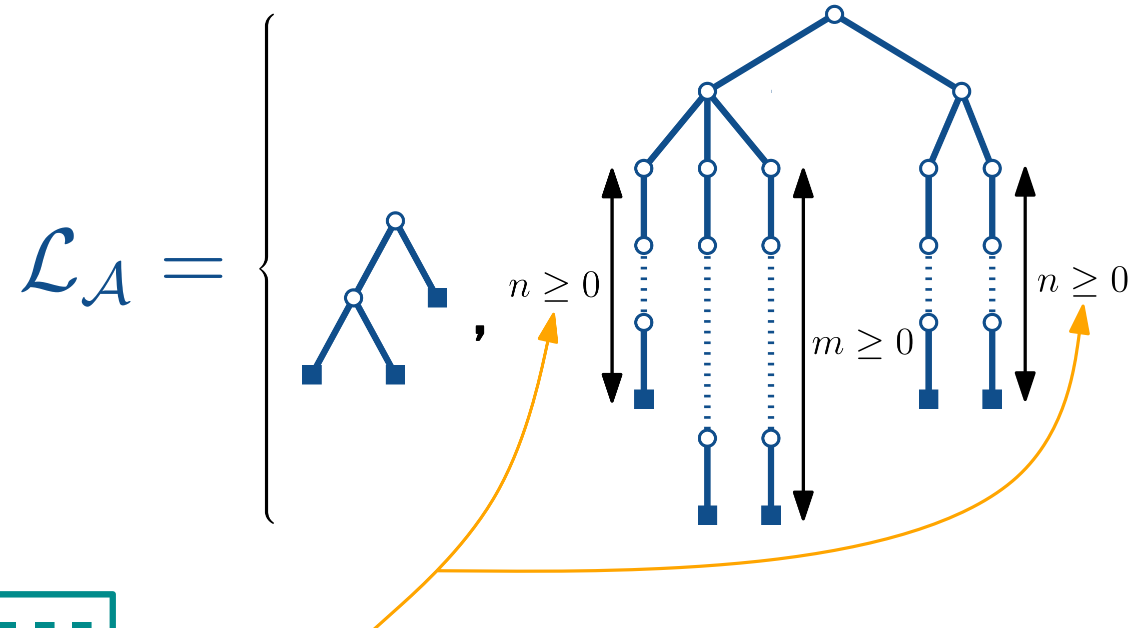
Multiple Tree Automata (MTA)





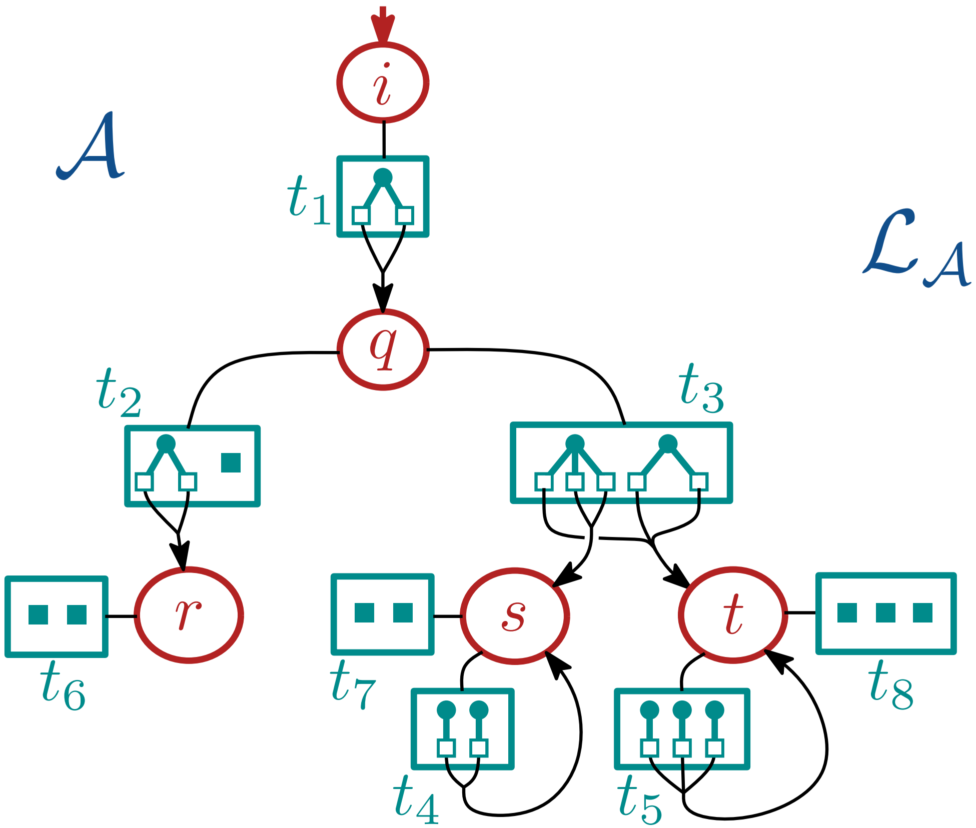
Multiple Tree Automata (MTA)



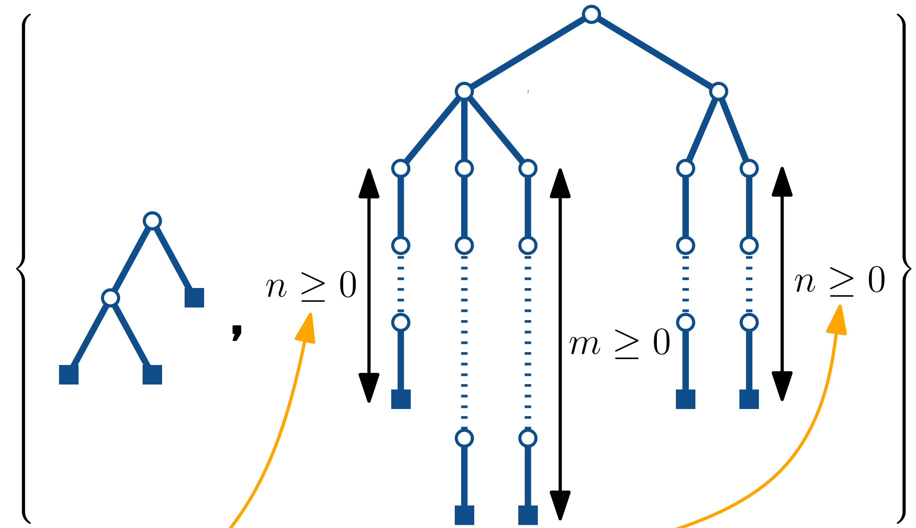


same n: can handle dependencies between a **bounded** number of nodes at the same height

Multiple Tree Automata (MTA)



$\mathcal{L}_{\mathcal{A}} = \{$



$\}$

same n: can handle dependencies between a **bounded** number of nodes at the same height

Def [Non-determinism]:

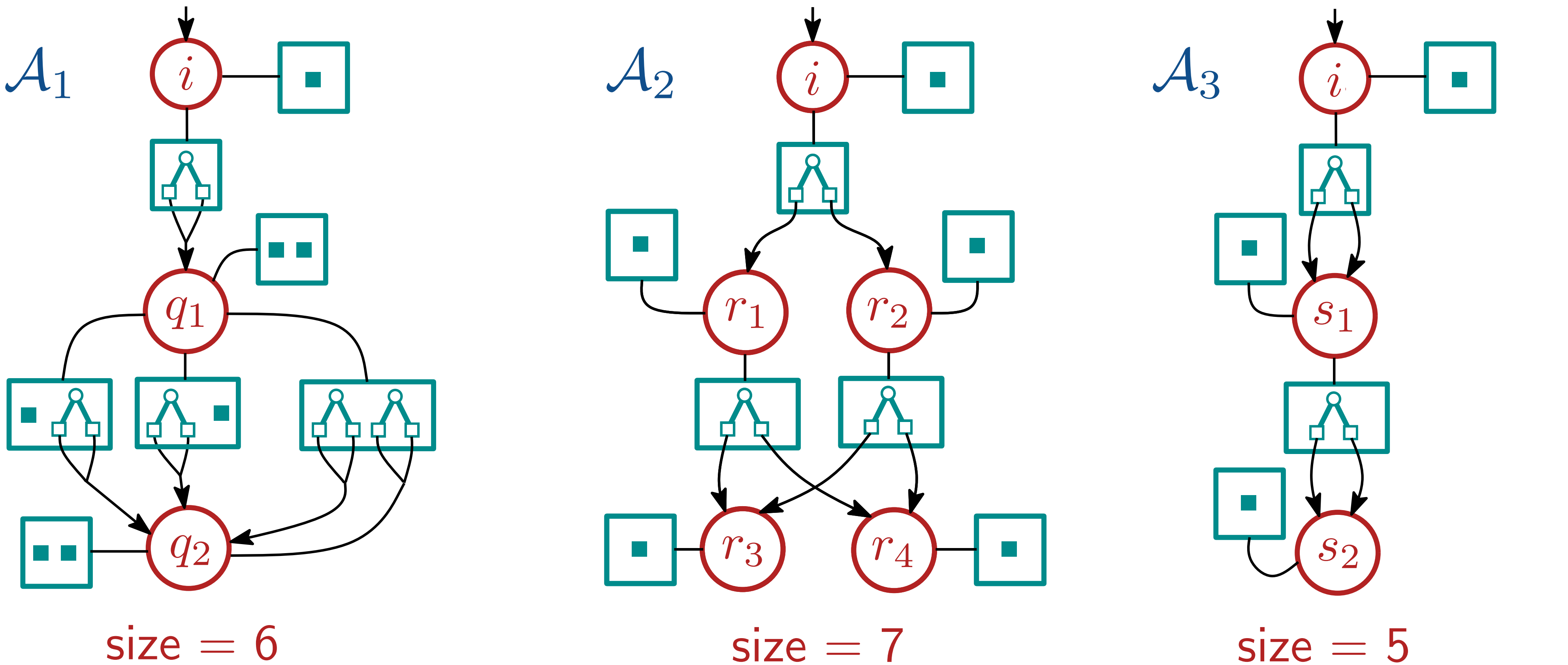
Non-deterministic MTA iff $|I| > 1$ or $\exists q \in Q_k, (a_1, \ldots, a_k) \in \Sigma^k$,

$$(q, (a_1, \ldots, a_k), P, \vec{p}) \text{ and } (q, (a_1, \ldots, a_k), P', \vec{p'}) \in \Delta$$

Deterministic MTA otherwise.

Minimization: size of a MTA

Minimize = Compute the smallest equivalent Deterministic MTA
Size = Number of transitions → Not enough anymore!



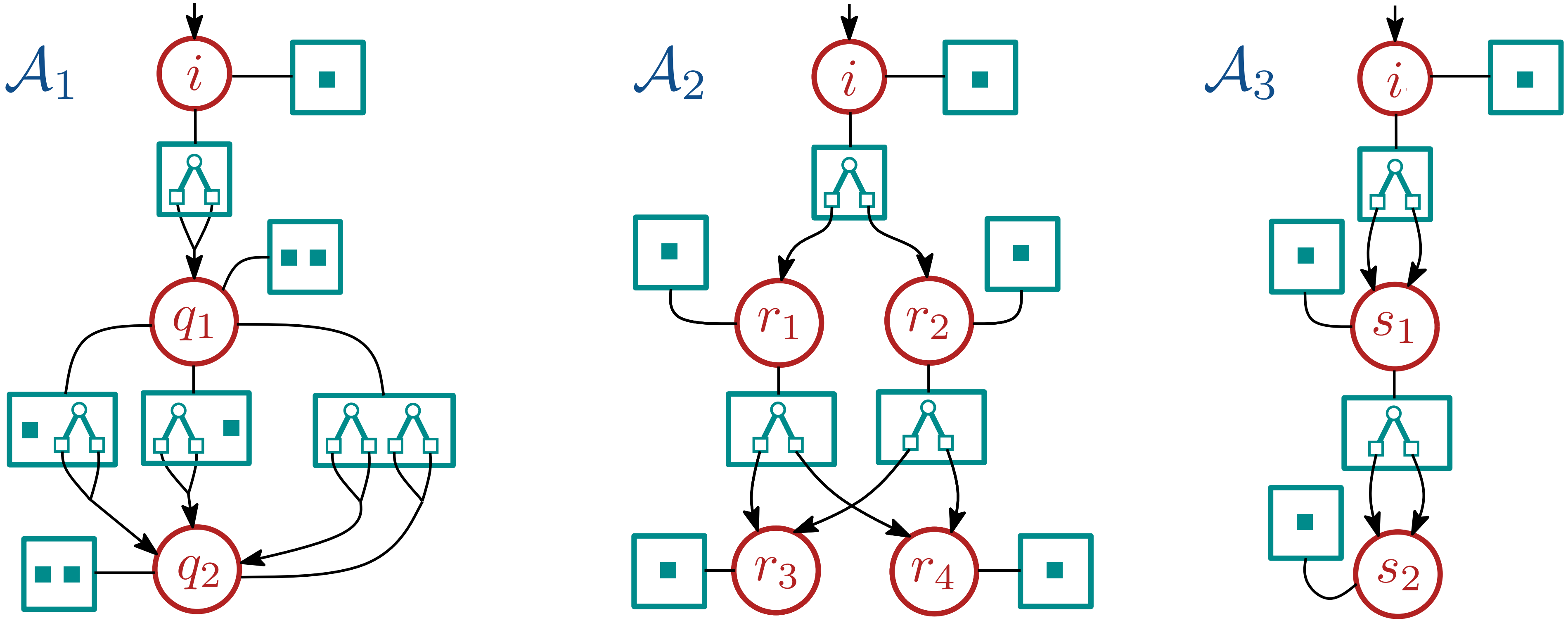
$\mathcal{L}_{\mathcal{A}_1} = \mathcal{L}_{\mathcal{A}_2} = \mathcal{L}_{\mathcal{A}_3} = \{$ Binary trees of height less than 3$\}$

Minimization: size of a MTA

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

Size = Total length of transitions



size = 12   size = 8   size = 5

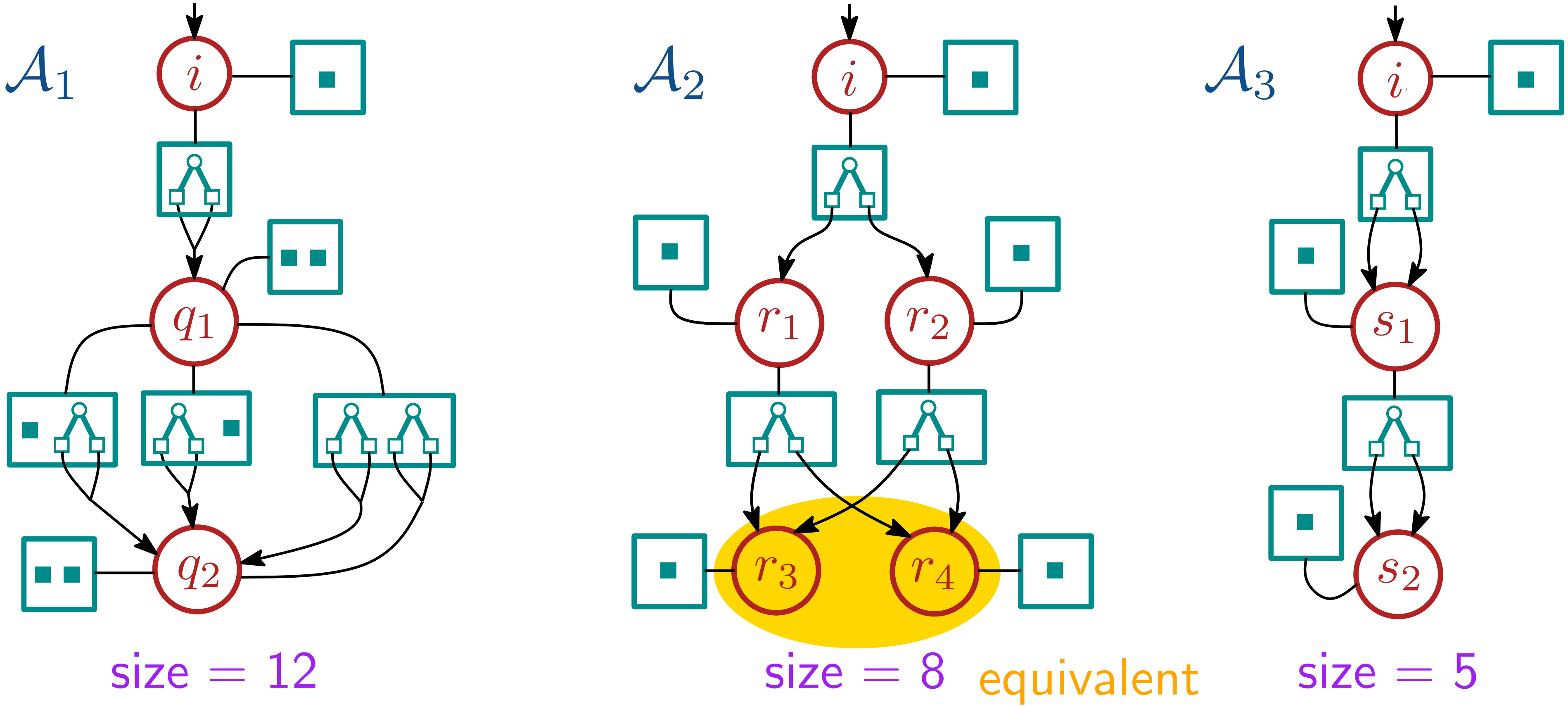
$$\mathcal{L}_{\mathcal{A}_1} = \mathcal{L}_{\mathcal{A}_2} = \mathcal{L}_{\mathcal{A}_3} = \{\text{ Binary trees of height less than 3}\}$$

Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

Size = Total length of transitions

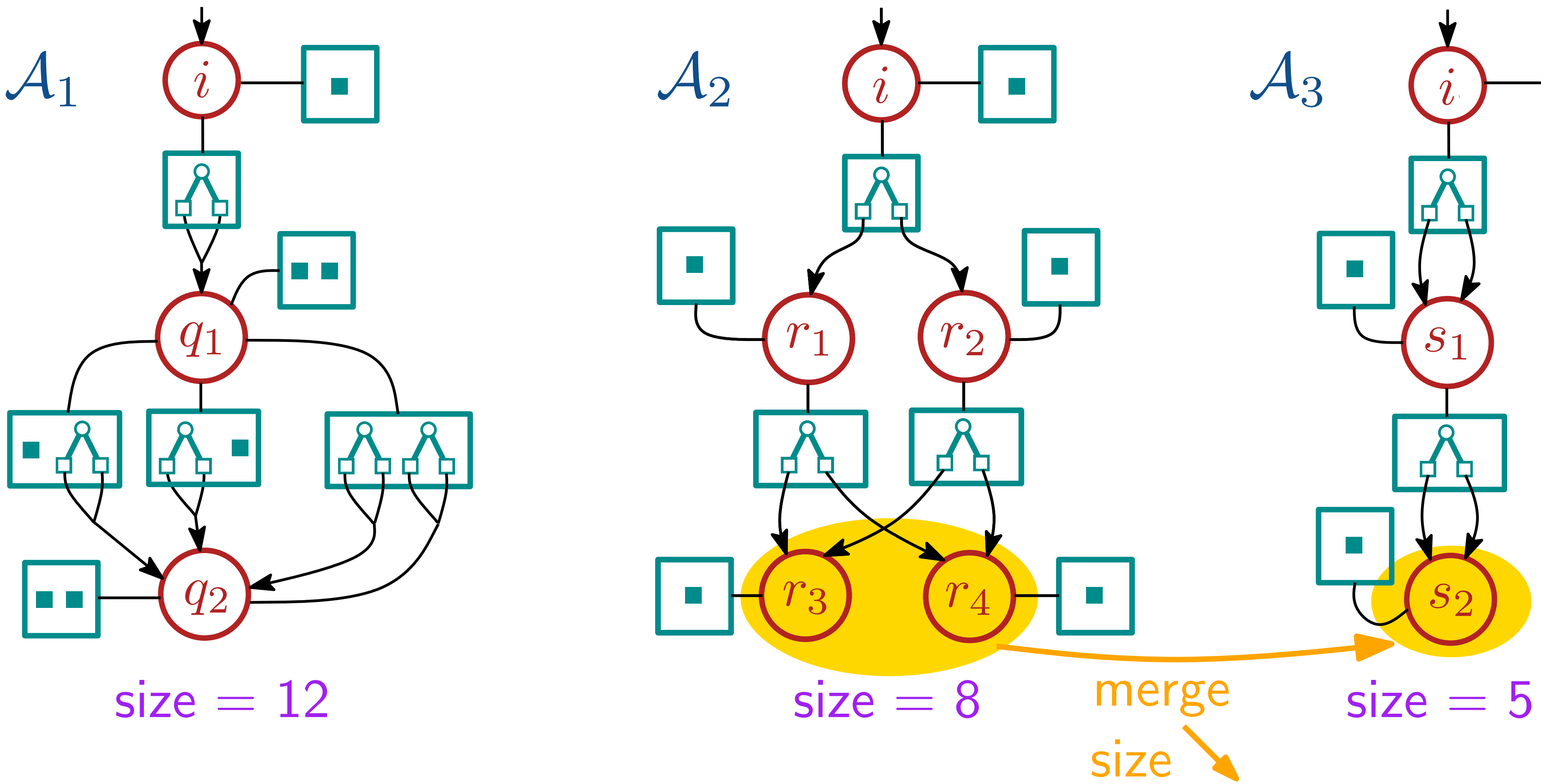


Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

Size = Total length of transitions

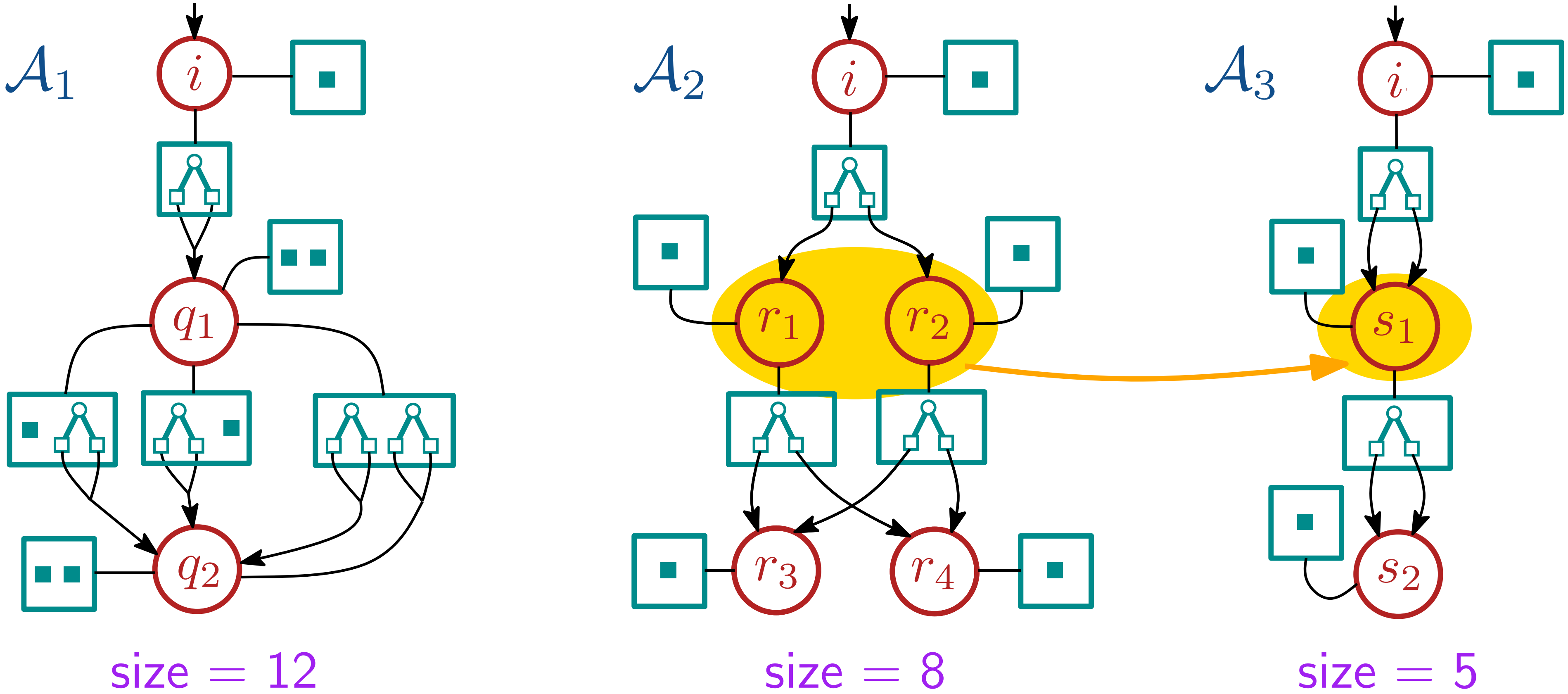


Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

Size = Total length of transitions

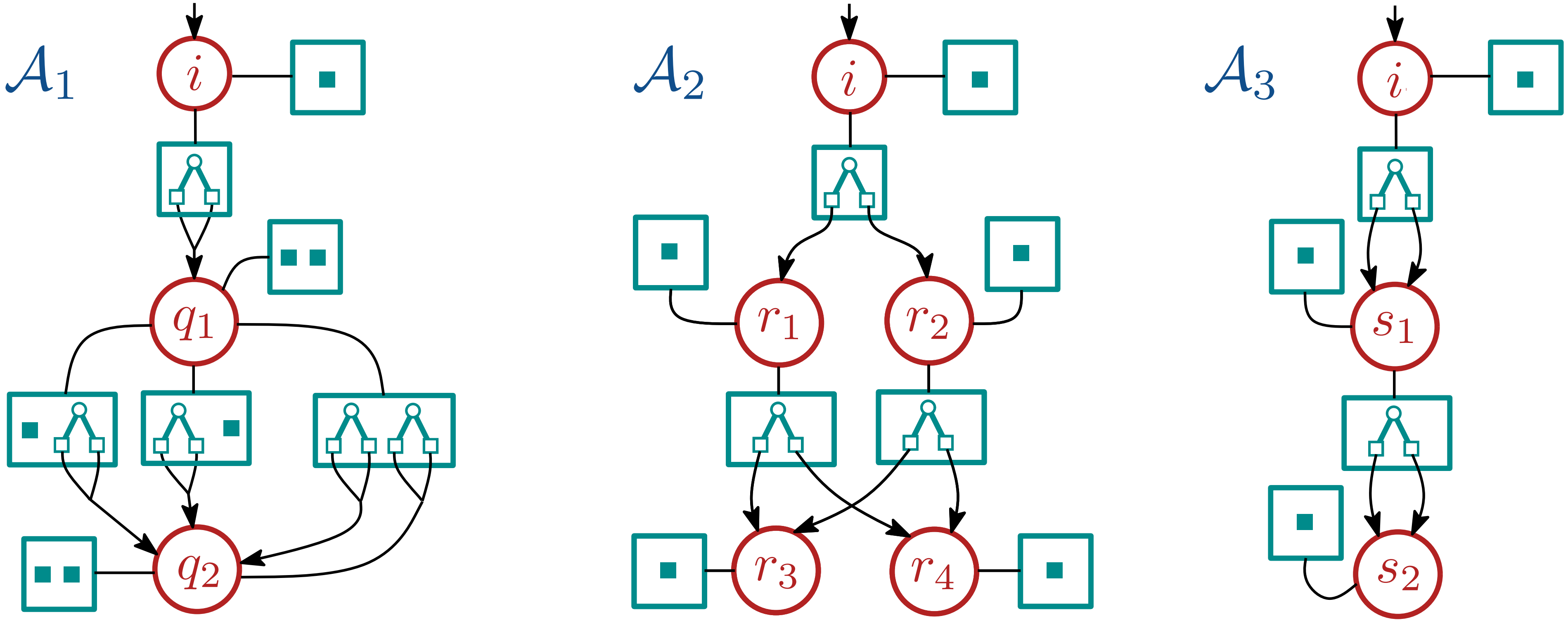


Minimization: state equivalence

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

Size = Total length of transitions



size = 12 size = 8 size = 5

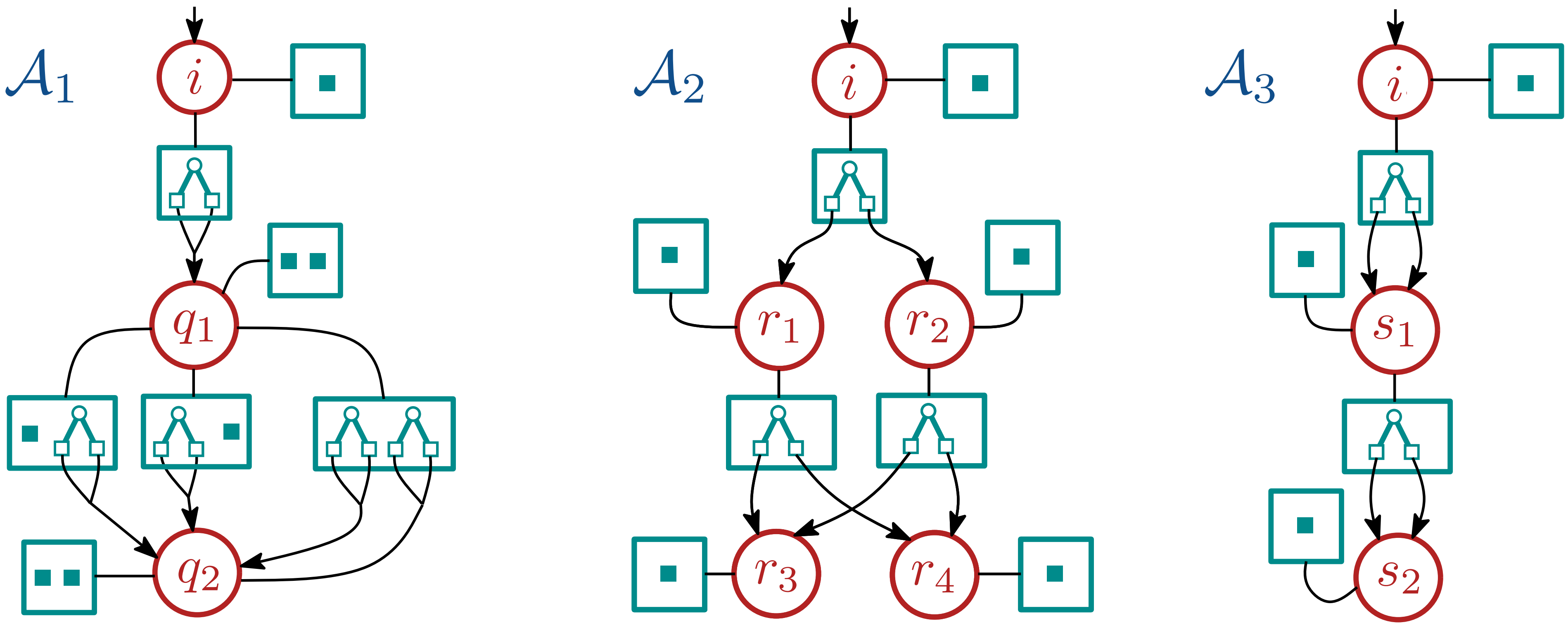
⚠ No equivalent states

Minimization: splitting

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

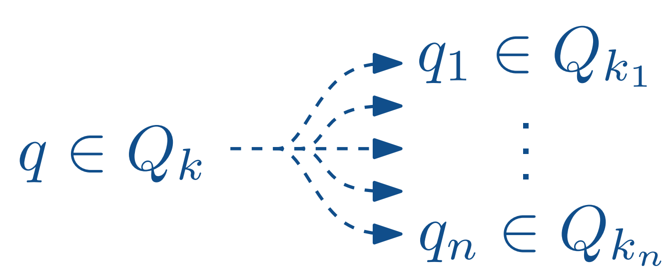
Size = Total length of transitions



size = 12 size = 8 size = 5

⚠ No equivalent states

New operation: **splitting**



$\sum k_i = k$

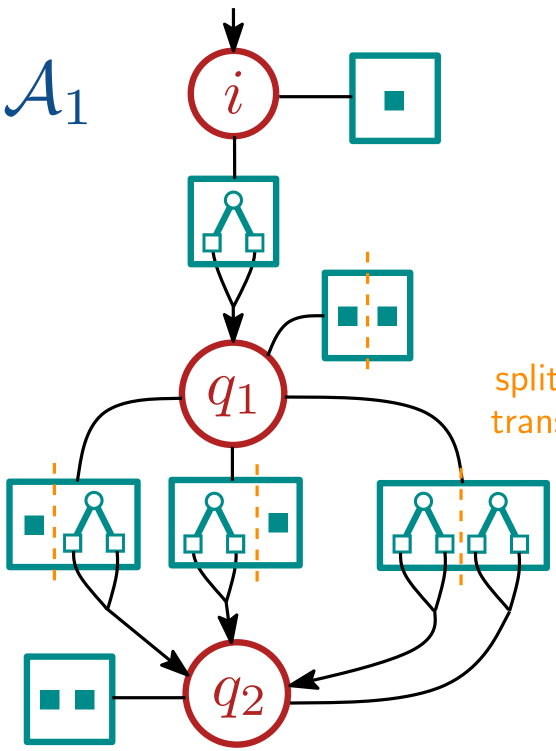
Minimization: splitting

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

Size = Total length of transitions

$\mathcal{A}_1$



splittable transitions + independence = splittable state

size = 12

⚠ No equivalent states

New operation: **splitting**

$q \in Q_k$ ⇢ $q_1 \in Q_{k_1}$, $\vdots$, $q_n \in Q_{k_n}$

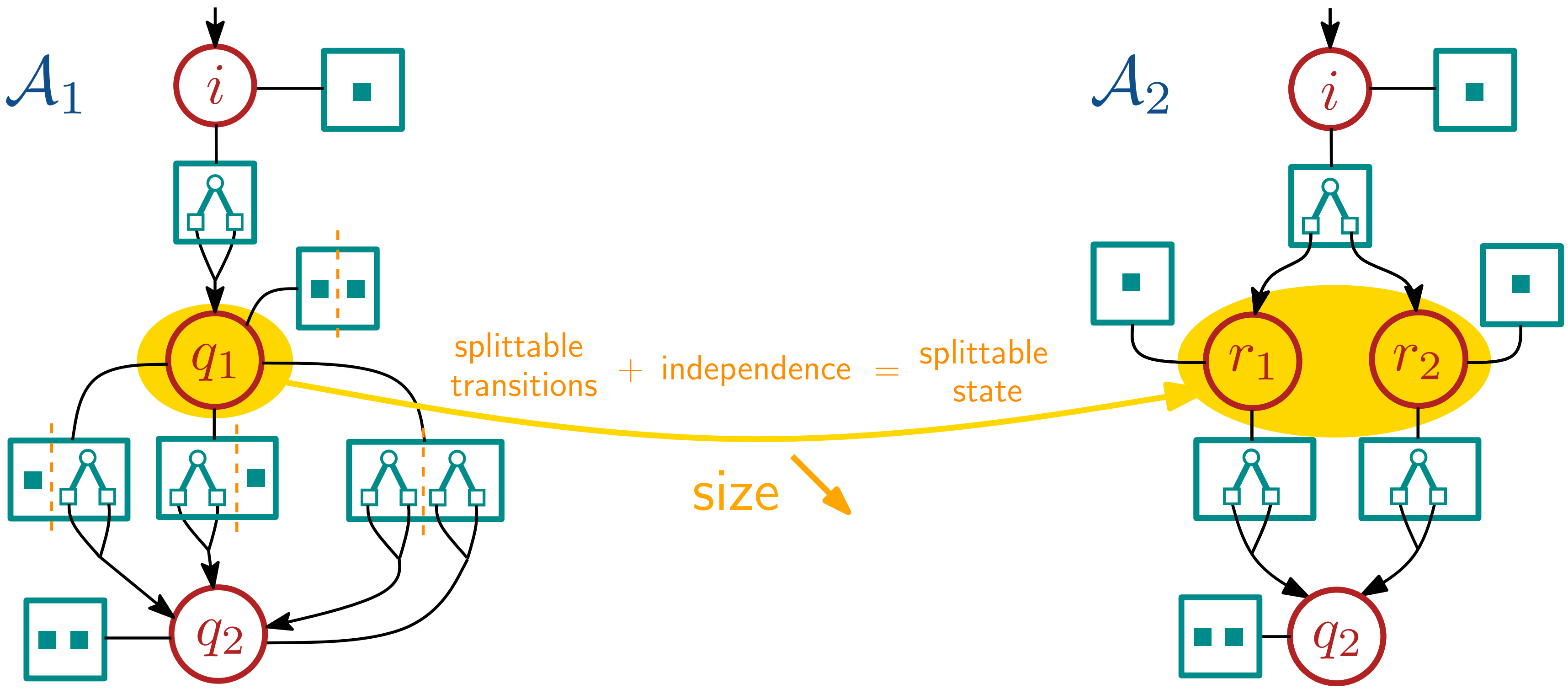
$\sum k_i = k$

Minimization: splitting

Minimize = Compute the smallest equivalent Deterministic MTA

~~Size = Number of transitions~~ → Not enough anymore!

Size = Total length of transitions



size = 12

size = 7

No equivalent states

New operation: **splitting**

$q \in Q_k$ ⇢ $q_1 \in Q_{k_1}$, ⋮, $q_n \in Q_{k_n}$

$\sum k_i = k$

Minimization: minimal DMTA

Minimize = Compute the smallest equivalent Deterministic MTA

Size = Total length of transitions

Theorem

A MTA without equivalent or splittable states is minimal.

This minimal automaton can be computed for any DMTA.

Sketch of the minimization algorithm

- Compute and merge any equivalent states.
- Compute and split any splittable states.
- Repeat until a fixpoint is reached.

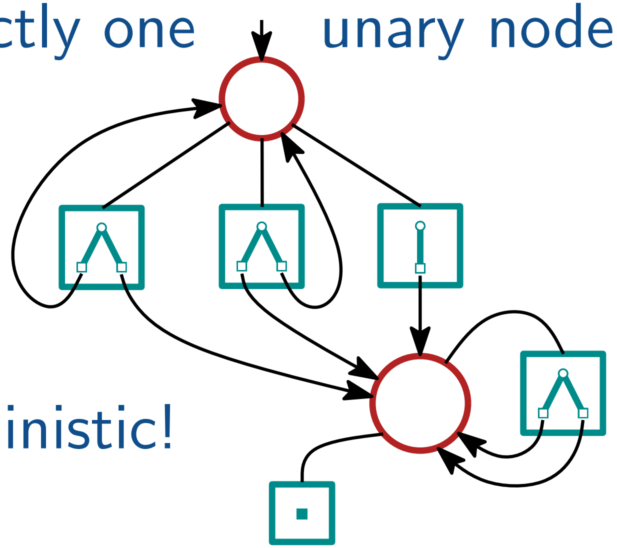
Closure properties of the tree languages

Theorem

1. MTA are closed under union and concatenation.
3. Non-deterministic MTA are strictly more powerful than deterministic ones.
2. MTA are not closed under complementation.

Proof:

1. Straightforward
2. Language of unary-binary trees with exactly one unary node



non-deterministic!

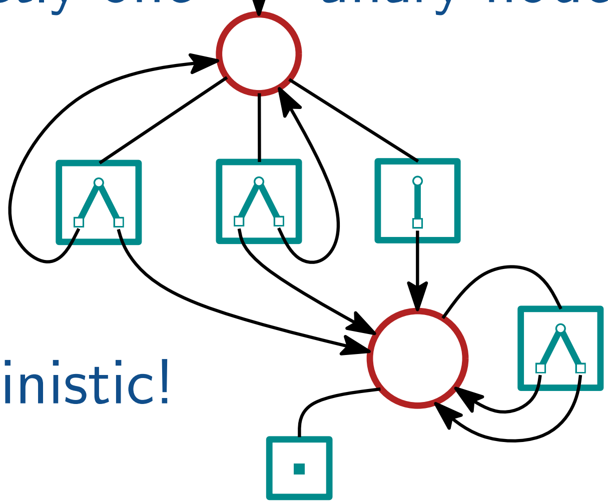
Closure properties of the tree languages

Theorem

1. MTA are closed under union and concatenation.
3. Non-deterministic MTA are strictly more powerful than deterministic ones.
2. MTA are not closed under complementation.

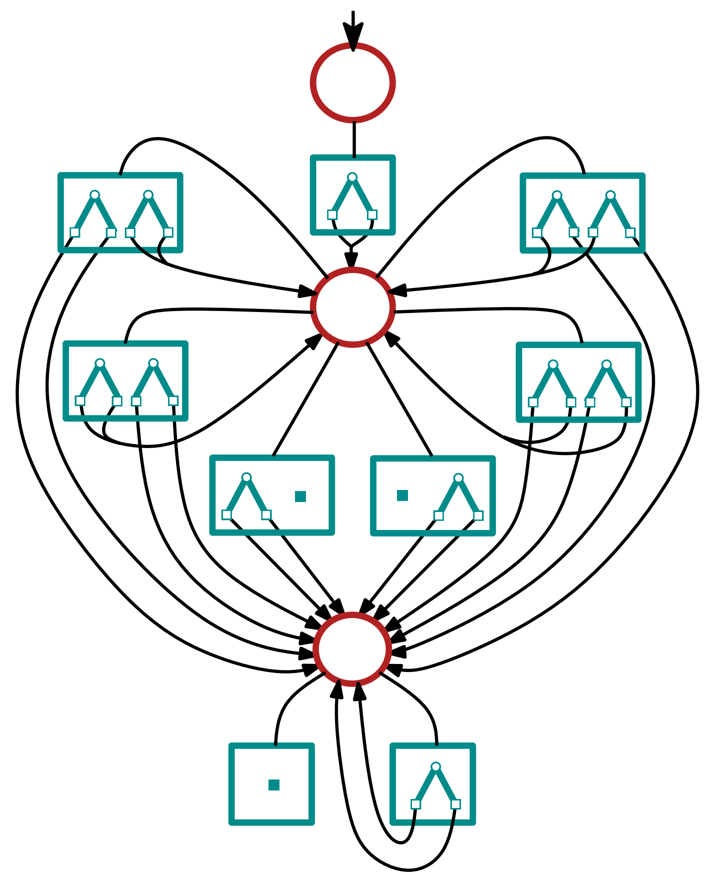
Proof:

1. Straightforward
2. Language of unary-binary trees with exactly one unary node



non-deterministic!

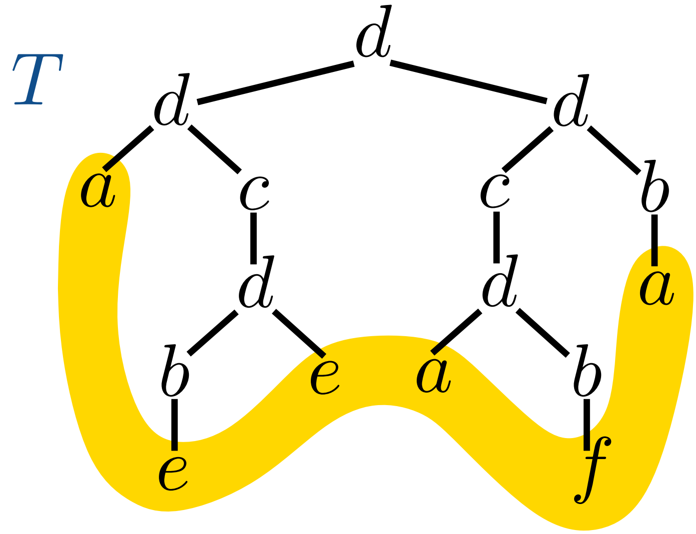
3...



Yield of a MTA

Def [Yield of an MTA $\mathcal{A}$]:

Word language $Yield(\mathcal{A}) = \{border(T) : T \in \mathcal{L}_{\mathcal{A}}\}$



$border(T) = aeeafa$

Theorem

$Yield$(MTA) are equivalent to **LCFRS** languages.

Context-free ⊂ Mildly context-sensitive ⊂ Context-sensitive

Linear Context-Free Rewriting Systems

Further works

Conjecture: MTA are closed under intersection.

$\rightarrow$ Semi-algorithm by computing joint dependences, believed to terminate eventually...

Further works

Conjecture: MTA are closed under intersection.

$\rightarrow$ Semi-algorithm by computing joint dependences, believed to terminate eventually...

What about **Bottom-up** MTA?

$\rightarrow$ useful for parsing

$\rightarrow$ more expressive in Deterministic Regular TA

Further works

Conjecture: MTA are closed under intersection.

$\rightarrow$ Semi-algorithm by computing joint dependences, believed to terminate eventually...

What about **Bottom-up** MTA?

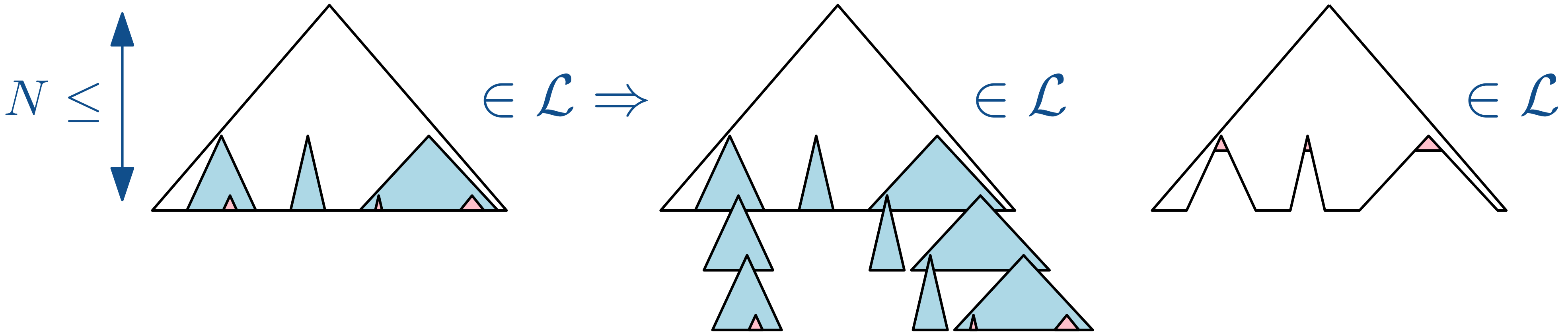
$\rightarrow$ useful for parsing

$\rightarrow$ more expressive in Deterministic Regular TA

Characterize the tree languages recognized by MTA

$\rightarrow$ Regular TL $\subset$ Multiple TL $\subset$ Context-free TL

$\rightarrow$ Pumping lemma, swapping lemma, other tools?



Thank you!