Verification of infinite state systems

Angelo Montanari and Gabriele Puppis

Department of Mathematics and Computer Science
University of Udine, Italy
{montana,puppis}@dimi.uniud.it

Go

In this part

We reduce the model checking problem for MSO logic over

- colored semi-infinite lines
- colored infinite trees

to suitable acceptance problems respectively for

- sequential Büchi automata (Büchi Theorem exploited)
- Rabin tree automata (Rabin Theorem exploited)

In analogy to the case of the semi-infinite line, Rabin Theorem

Theorem (Rabin '69)

For any MSO-formula ψ with free variables $X_1, \ldots, X_m$, one can compute a Rabin tree automaton $\mathcal{A}_\psi$ over $\mathbb{B}^m$ such that, for every tuple of unary predicates $P_1, \ldots, P_m \subseteq \mathbb{B}^*$

$$(\mathbb{B}^*, \delta_0, \delta_1, \bar{P}) \vDash \psi[P_1/X_1, \ldots, P_m/X_m] \quad \text{iff} \quad \mathcal{T}_{2,\bar{P}} \in \mathscr{L}(\mathcal{A}_\psi)$$

can be exploited to reduce the decision problem for the MSO-theory of an *expanded* infinite complete tree $(\mathbb{B}^*, \delta_0, \delta_1, \bar{P})$ to the acceptance problem of $\mathcal{T}_{2,\bar{P}}$ (i.e., the **characteristic colored tree** encoding $(\mathbb{B}^*, \delta_0, \delta_1, \bar{P})$) for Rabin tree automata.

Definition (Acceptance problem)

The **acceptance problem** of a colored and complete infinite binary tree $\mathcal{T}$, denoted $Acc_{\mathcal{T}}$, consists in deciding, for any Rabin tree automaton $\mathcal{A}$, whether

$$\mathcal{T} \in \mathscr{L}(\mathcal{A}) \qquad (\mathcal{A} \textit{ accepts } \mathcal{T})$$

Definition (Acceptance problem)

The **acceptance problem** of a colored and complete infinite binary tree $\mathcal{T}$, denoted $Acc_{\mathcal{T}}$, consists in deciding, for any Rabin tree automaton $\mathcal{A}$, whether

$$\mathcal{T} \in \mathscr{L}(\mathcal{A}) \qquad (\mathcal{A} \text{ accepts } \mathcal{T})$$

Corollary (of RabinTheorem)

The problem of deciding the MSO-theory of a colored and complete infinite binary tree $\left(\mathbb{B}^*, \delta_0, \delta_1, \bar{P}\right)$ *is reducible to the problem* $Acc_{\mathcal{T}_{2,\bar{P}}}$.

Such a result can be easily generalized to k-ary and non-complete trees (i.e., trees with leaves).

To this end, we slightly modify the notion of tree automaton:

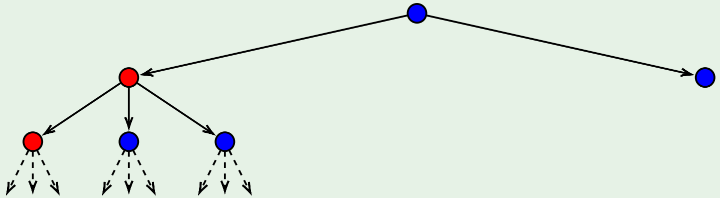
1. the transition relation Δ is now a subset of $Q \times C \times Q^k$
2. the automaton reads a dummy symbol $\perp$
   if a vertex of the input tree is missing

To this end, we slightly modify the notion of tree automaton:

1. the transition relation Δ is now a subset of $Q \times C \times Q^k$
2. the automaton reads a dummy symbol $\perp$ if a vertex of the input tree is missing

Example

Consider the ternary non-complete $\{red, blue\}$-colored tree



and the Rabin tree automaton having

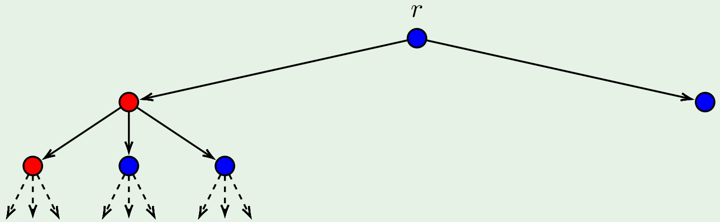
- three states, r, b, and d, that signal which color was seen last
- transitions $(r/b, red, r, r, r)$, $(r/b, blue, b, b, b)$, $(r, dummy, d, d, d)$, $(b, dummy, d, d, d)$, $(d, dummy, d, d, d)$

To this end, we slightly modify the notion of tree automaton:

1. the transition relation Δ is now a subset of $Q \times C \times Q^k$
2. the automaton reads a dummy symbol $\perp$ if a vertex of the input tree is missing

Example

Consider the ternary non-complete $\{red, blue\}$-colored tree



and the Rabin tree automaton having

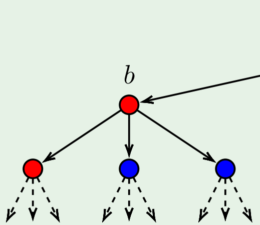
- three states, r, b, and d, that signal which color was seen last
- transitions $(r/b, red, r, r, r)$, $(r/b, blue, b, b, b)$, $(r, dummy, d, d, d)$, $(b, dummy, d, d, d)$, $(d, dummy, d, d, d)$

To this end, we slightly modify the notion of tree automaton:

1. the transition relation Δ is now a subset of $Q \times C \times Q^k$
2. the automaton reads a dummy symbol $\perp$ if a vertex of the input tree is missing

Example

Consider the ternary non-complete $\{red, blue\}$-colored tree



and the Rabin tree automaton having

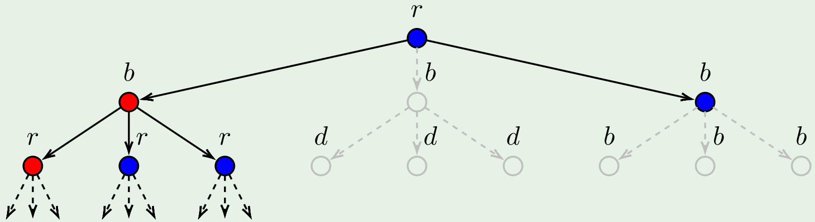
- three states, r, b, and d, that signal which color was seen last
- transitions $(r/b, red, r, r, r)$, $(r/b, blue, b, b, b)$, $(r, dummy, d, d, d)$, $(b, dummy, d, d, d)$, $(d, dummy, d, d, d)$

To this end, we slightly modify the notion of tree automaton:

1. the transition relation Δ is now a subset of $Q \times C \times Q^k$
2. the automaton reads a dummy symbol $\perp$ if a vertex of the input tree is missing

Example

Consider the ternary non-complete $\{red, blue\}$-colored tree



and the Rabin tree automaton having

- three states, r, b, and d, that signal which color was seen last
- transitions $(r/b, red, r, r, r)$, $(r/b, blue, b, b, b)$, $(r, dummy, d, d, d)$, $(b, dummy, d, d, d)$, $(d, dummy, d, d, d)$

Proposition

The problem of deciding the MSO-theory of a k-ary (possibly incomplete) colored tree $\left(D, \delta_0, ..., \delta_{k-1}, \bar{P}\right)$ is reducible to the problem $Acc_{\mathcal{T}_{k,\bar{P}}}$.

In the following, we describe a method to identify infinite colored trees $\mathcal{T}$, including incomplete ones, for which $Acc_{\mathcal{T}}$ is decidable.

Proposition

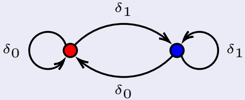
The acceptance problem of any **regular colored tree** (i.e., the unfolding of a finite colored graph) is decidable.

Proposition

The acceptance problem of any **regular colored tree** (i.e., the unfolding of a finite colored graph) is decidable.

Proof

Let $\mathcal{G}$ be a finite colored graph

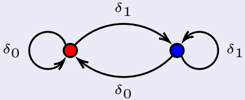


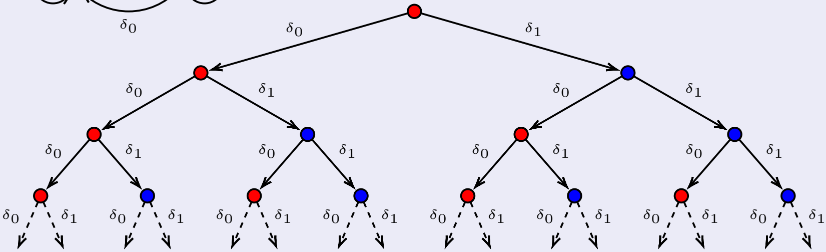
Proposition

The acceptance problem of any **regular colored tree** (i.e., the unfolding of a finite colored graph) is decidable.

Proof

Let $\mathcal{G}$ be a finite colored graph and $\mathcal{T}$ its unfolding.



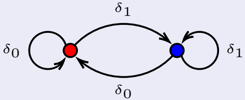


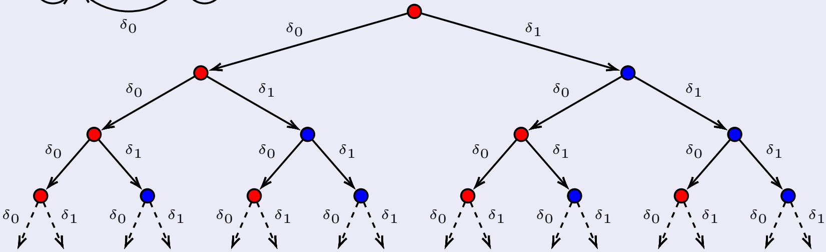
Proposition

The acceptance problem of any **regular colored tree** (i.e., the unfolding of a finite colored graph) is decidable.

Proof

We view $\mathcal{G}$ as an automaton $\mathcal{A}_{\mathcal{T}}$ recognizing the singleton $\{\mathcal{T}\}$.





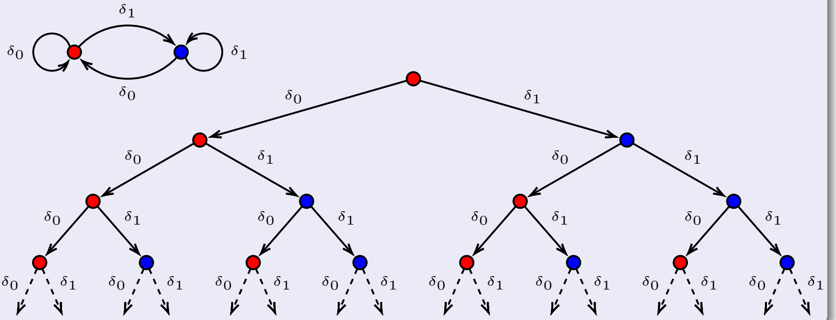
Proposition

The acceptance problem of any **regular colored tree** (i.e., the unfolding of a finite colored graph) is decidable.

Proof

We view $\mathcal{G}$ as an automaton $\mathcal{A}_{\mathcal{T}}$ recognizing the singleton $\{\mathcal{T}\}$.

$\Rightarrow$ given any automaton $\mathcal{A}$, $\mathcal{T} \in \mathscr{L}(\mathcal{A})$ iff $\mathscr{L}(\mathcal{A}) \cap \mathscr{L}(\mathcal{A}_{\mathcal{T}}) \neq \emptyset$



Goal

We now want to extend the class of colored trees for which the acceptance problem turns out to be decidable.

Goal

We now want to extend the class of colored trees for which the acceptance problem turns out to be decidable.

Idea

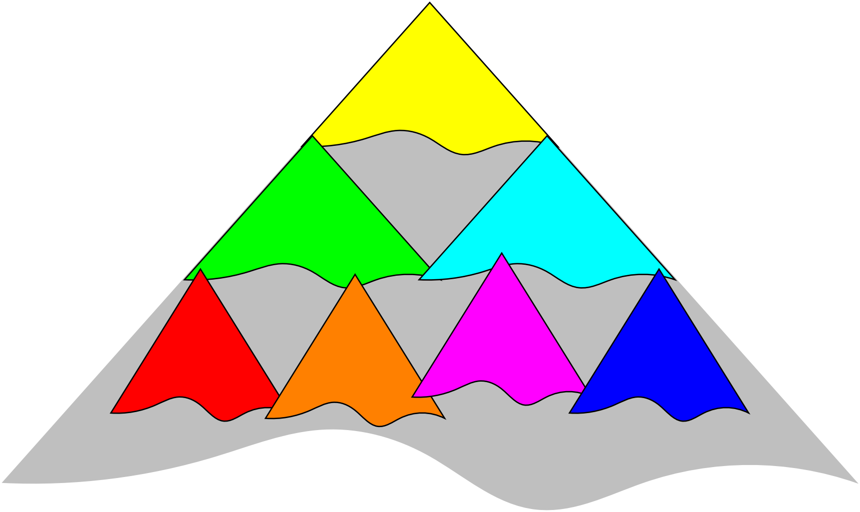
Reduce the acceptance problem of a non regular tree $\mathcal{T}$ to an equivalent acceptance problem of a regular tree $\vec{\mathcal{T}}$:

1. decompose $\mathcal{T}$ into factors
2. 'distill' the relevant features of each factor F (features describe the behavior of a given automaton $\mathcal{A}$ on F)
3. reason on the feature tree $\vec{\mathcal{T}}$ (i.e., a tree-shaped arrangement of features).

The *features* of F w.r.t. $\mathcal{A}$ are called $\mathcal{A}$-**type** of F.
The *feature tree* $\vec{\mathcal{T}}$ is called $\mathcal{A}$-**contraction** of $\mathcal{T}$.

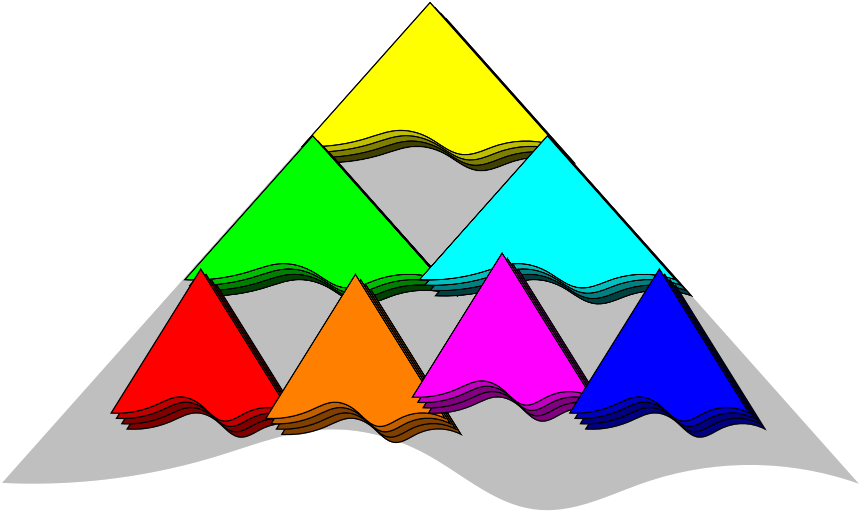
A picture of the method:

Given a tree $\mathcal{T}$, decompose it into **factors** ...



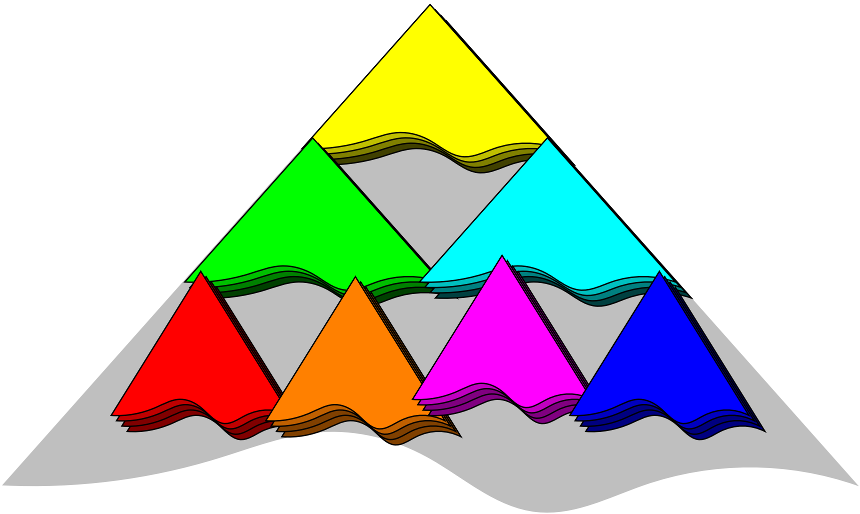
A picture of the method:

... then consider the equivalence classes
induced by the $\mathcal{A}$-**types** of the factors ...



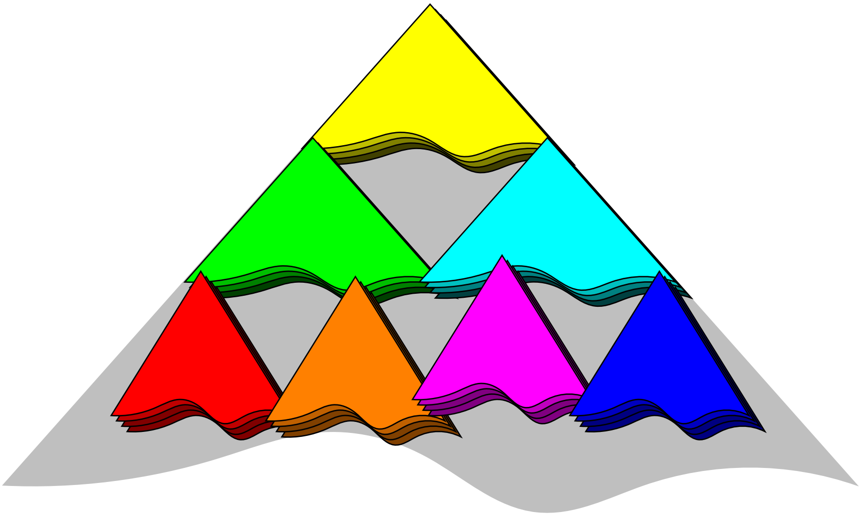
A picture of the method:

... The automaton $\mathcal{A}$ has the **same behavior**
on all trees in each equivalence class



A picture of the method:

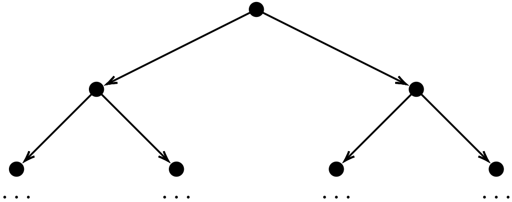
$\Rightarrow$ We replace $\mathcal{A}$ with an automaton $\vec{\mathcal{A}}$ that runs on the (possibly regular) $\mathcal{A}$-**contraction** and mimics $\mathcal{A}$



Definition (Factorization)

A **factorization** of a tree $\mathcal{T}$ is an *uncolored* tree Π such that

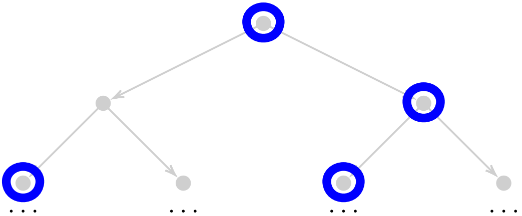
- $\{root(\mathcal{T})\} \subseteq \mathcal{D}om(\Pi) \subseteq \mathcal{D}om(\mathcal{T})$



Definition (Factorization)

A **factorization** of a tree $\mathcal{T}$ is an *uncolored* tree Π such that

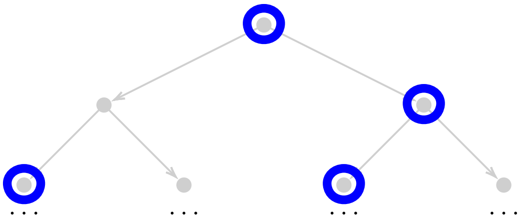
- $\{root(\mathcal{T})\} \subseteq \mathcal{D}om(\Pi) \subseteq \mathcal{D}om(\mathcal{T})$



Definition (Factorization)

A **factorization** of a tree $\mathcal{T}$ is an *uncolored* tree Π such that

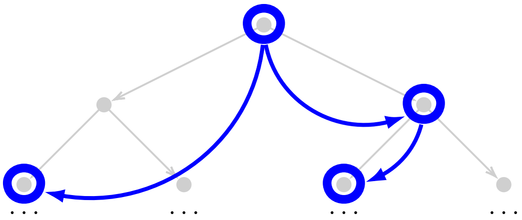
- $\{root(\mathcal{T})\} \subseteq \mathcal{D}om(\Pi) \subseteq \mathcal{D}om(\mathcal{T})$
- the edges are given by the *ancestor relation* of $\mathcal{T}$



Definition (Factorization)

A **factorization** of a tree $\mathcal{T}$ is an *uncolored* tree Π such that

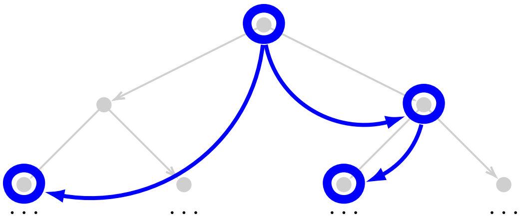
- $\{root(\mathcal{T})\} \subseteq \mathcal{D}om(\Pi) \subseteq \mathcal{D}om(\mathcal{T})$
- the edges are given by the *ancestor relation* of $\mathcal{T}$



Definition (Factorization)

A **factorization** of a tree $\mathcal{T}$ is an *uncolored* tree Π such that

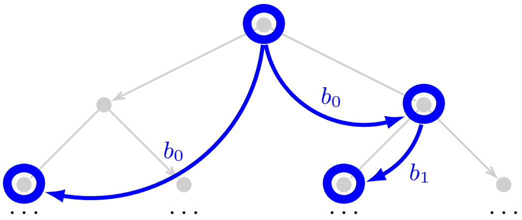
- $\{root(\mathcal{T})\} \subseteq \mathcal{D}om(\Pi) \subseteq \mathcal{D}om(\mathcal{T})$
- the edges are given by the *ancestor relation* of $\mathcal{T}$
- the edge labels are chosen arbitrarily from a finite set B.



Definition (Factorization)

A **factorization** of a tree $\mathcal{T}$ is an *uncolored* tree Π such that

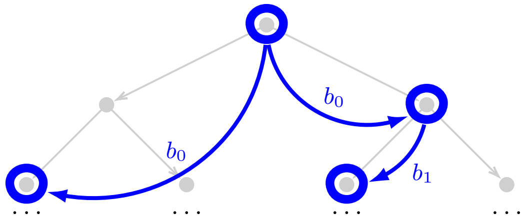
- $\{root(\mathcal{T})\} \subseteq \mathcal{D}om(\Pi) \subseteq \mathcal{D}om(\mathcal{T})$
- the edges are given by the *ancestor relation* of $\mathcal{T}$
- the edge labels are chosen arbitrarily from a finite set B.



Definition (Factorization)

A **factorization** of a tree $\mathcal{T}$ is an *uncolored* tree Π such that

- $\{root(\mathcal{T})\} \subseteq \mathcal{D}om(\Pi) \subseteq \mathcal{D}om(\mathcal{T})$
- the edges are given by the *ancestor relation* of $\mathcal{T}$
- the edge labels are chosen arbitrarily from a finite set B.



Note: Π can be a non-deterministic tree
and it can have even unbounded/infinite degree.

Definition (Factor)

For any $u \in \mathcal{D}om(\Pi)$, the **factor** $\mathcal{T}_u$ of $\mathcal{T}$ in u
is the subgraph of $\mathcal{T}$ induced by the set

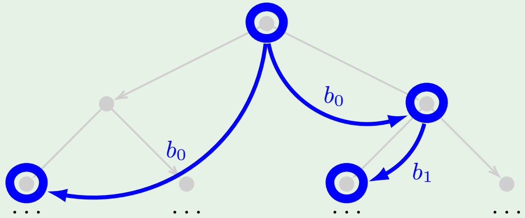
$$\left\{ v \in \mathcal{D}om(T) \;:\; u \sqsubseteq v \text{ and } v \sqsubseteq u' \text{ for all successors } u' \text{ of } u \text{ in } \Pi \right\}$$

Definition (Factor)

For any $u \in \mathcal{D}om(\Pi)$, the **factor** $\mathcal{T}_u$ of $\mathcal{T}$ in u is the subgraph of $\mathcal{T}$ induced by the set

$$\{v \in \mathcal{D}om(T) \,:\, u \sqsubseteq v \text{ and } v \sqsubseteq u' \text{ for all successors } u' \text{ of } u \text{ in } \Pi\}$$

Example

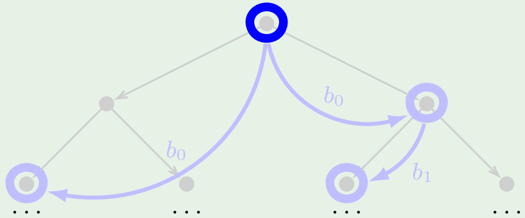


Definition (Factor)

For any $u \in \mathcal{D}om(\Pi)$, the **factor** $\mathcal{T}_u$ of $\mathcal{T}$ in u is the subgraph of $\mathcal{T}$ induced by the set

$$\{ v \in \mathcal{D}om(T) \ : \ u \sqsubseteq v \text{ and } v \not\sqsubseteq u' \text{ for all successors } u' \text{ of } u \text{ in } \Pi \}$$

Example

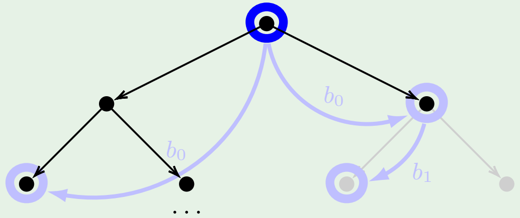


Definition (Factor)

For any $u \in \mathcal{D}om(\Pi)$, the **factor** $\mathcal{T}_u$ of $\mathcal{T}$ in u is the subgraph of $\mathcal{T}$ induced by the set

$$\{ v \in \mathcal{D}om(T) \ : \ u \sqsubseteq v \text{ and } v \not\sqsubseteq u' \text{ for all successors } u' \text{ of } u \text{ in } \Pi \}$$

Example

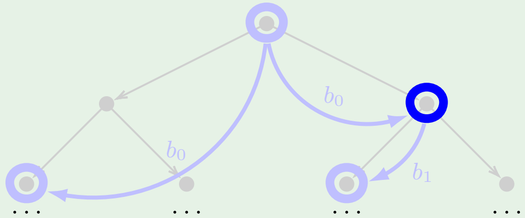


Definition (Factor)

For any $u \in \mathcal{D}om(\Pi)$, the **factor** $\mathcal{T}_u$ of $\mathcal{T}$ in u is the subgraph of $\mathcal{T}$ induced by the set

$$\{v \in \mathcal{D}om(T) \; : \; u \sqsubseteq v \text{ and } v \not\sqsubseteq u' \text{ for all successors } u' \text{ of } u \text{ in } \Pi\}$$

Example

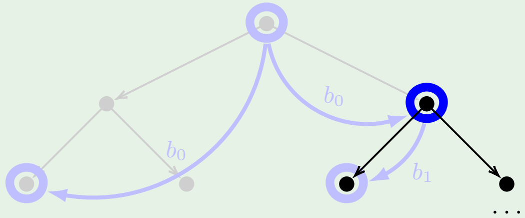


Definition (Factor)

For any $u \in \mathcal{D}om(\Pi)$, the **factor** $\mathcal{T}_u$ of $\mathcal{T}$ in u is the subgraph of $\mathcal{T}$ induced by the set

$$\left\{ v \in \mathcal{D}om(T) \ : \ u \sqsubseteq v \text{ and } v \sqsubseteq u' \text{ for all successors } u' \text{ of } u \text{ in } \Pi \right\}$$

Example



We also need to expand factors with information about the *edge labels* of the factorization Π:

Definition (Marked factor)

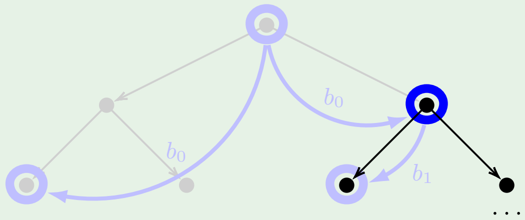
Given $u \in \mathcal{D}om(\Pi)$, the **marked factor** T_u^+ is obtained from $\mathcal{T}_u$ by *recoloring each leaf* u' with the *label* of the edge of Π that reaches u'.

We also need to expand factors with information about the *edge labels* of the factorization Π:

Definition (Marked factor)

Given $u \in \mathcal{D}om(\Pi)$, the **marked factor** T_u^+ is obtained from $\mathcal{T}_u$ by *recoloring each leaf* u' with the *label* of the edge of Π that reaches u'.

Example

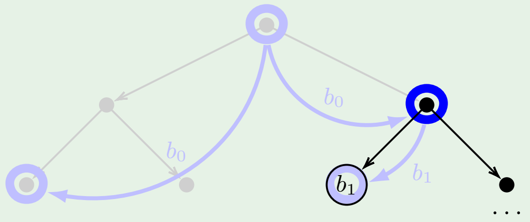


We also need to expand factors with information about the *edge labels* of the factorization Π:

Definition (Marked factor)

Given $u \in \mathcal{D}om(\Pi)$, the **marked factor** T_u^+ is obtained from $\mathcal{T}_u$ by *recoloring each leaf* u' with the *label* of the edge of Π that reaches u'.

Example

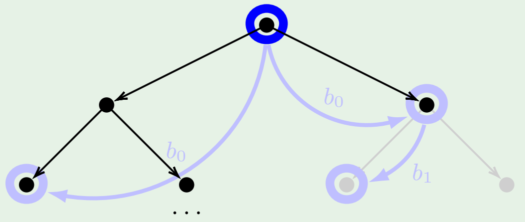


We also need to expand factors with information about the *edge labels* of the factorization Π:

Definition (Marked factor)

Given $u \in \mathcal{D}om(\Pi)$, the **marked factor** T_u^+ is obtained from $\mathcal{T}_u$ by *recoloring each leaf* u' with the *label* of the edge of Π that reaches u'.

Example

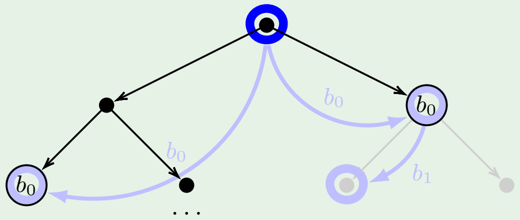


We also need to expand factors with information about the *edge labels* of the factorization Π:

Definition (Marked factor)

Given $u \in \mathcal{D}om(\Pi)$, the **marked factor** T_u^+ is obtained from $\mathcal{T}_u$ by *recoloring each leaf* u' with the *label* of the edge of Π that reaches u'.

Example

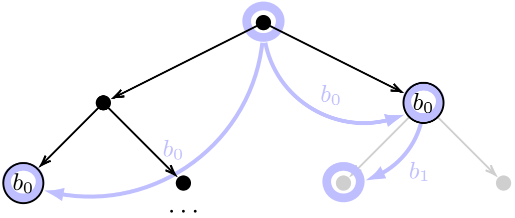


Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$,
the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_{\mathcal{A}}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \,:\, v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \,:\, \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.

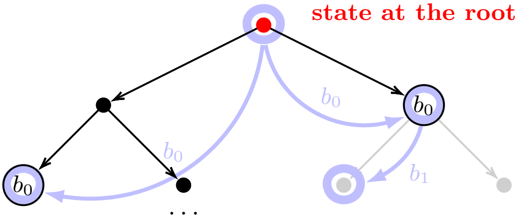


Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$,
the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_{\mathcal{A}}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \,:\, v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \,:\, \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.

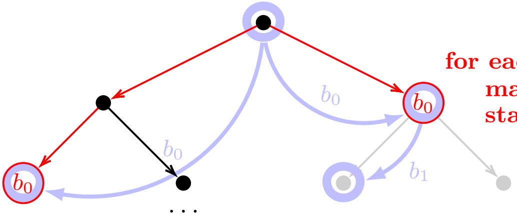


Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$, the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_\mathcal{A}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \ : \ v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \ : \ \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.



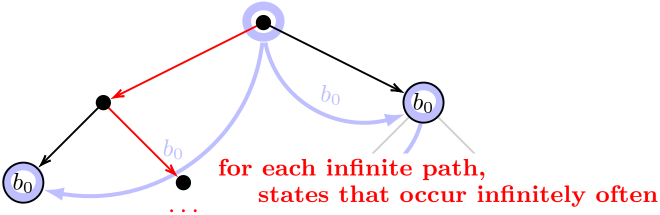
for each leaf, marker + state + states along access path

Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$,
the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_\mathcal{A}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \,:\, v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \,:\, \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.



Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$, the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_{\mathcal{A}}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \,:\, v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \,:\, \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.

There exist finitely many $\mathcal{A}$-types

Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$,
the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_\mathcal{A}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \,:\, v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \,:\, \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.

There exist finitely many $\mathcal{A}$-types

$\Rightarrow$ they induce an equivalence of finite index on the set of trees

Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$,
the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_\mathcal{A}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \,:\, v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \,:\, \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.

There exist finitely many $\mathcal{A}$-types

$\Rightarrow$ they induce an equivalence of finite index on the set of trees

$\Rightarrow$ we can see each $\mathcal{A}$-type as a color from a finite set

Definition ($\mathcal{A}$-Type)

Given an automaton M and a marked factor $\mathcal{T}_u^+$,
the $\mathcal{A}$-**type** $[\mathcal{T}_u^+]_{\mathcal{A}}$ is the *set of triples* of the form

$$\begin{pmatrix} \mathcal{R}(\varepsilon) \\ \{(\mathcal{T}(v), \mathcal{R}(v), \mathcal{I}mg(\mathcal{R}|\pi_v)) \,:\, v \in \mathcal{F}r(\mathcal{T})\} \\ \{\mathcal{I}nf(\mathcal{R}|\pi) \,:\, \pi \in \mathcal{B}ch(\mathcal{T})\} \end{pmatrix}$$

over all possible **partial runs** $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}_u^+$.

There exist finitely many $\mathcal{A}$-types

$\Rightarrow$ they induce an equivalence of finite index on the set of trees

$\Rightarrow$ we can see each $\mathcal{A}$-type as a color from a finite set

$\Rightarrow$ we can arrange the $\mathcal{A}$-types of the factors of a tree
in a tree-shaped colored structure called $\mathcal{A}$-contraction.

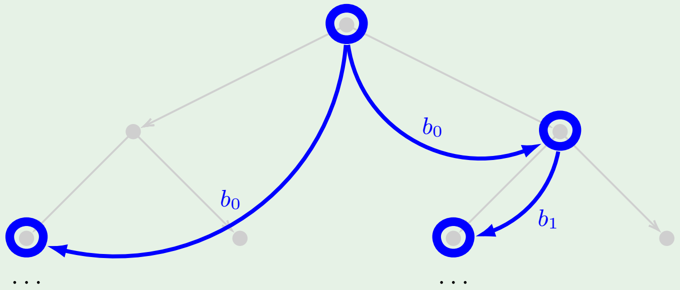
Definition ($\mathcal{A}$-contraction)

Given a tree $\mathcal{T}$, a factorization Π of $\mathcal{T}$, and an automaton $\mathcal{A}$, the $\mathcal{A}$-**contraction** $\vec{\mathcal{T}}$ of $\mathcal{T}$ is the tree obtained from Π by coloring its vertices u with the corresponding $\mathcal{A}$-type $[\mathcal{T}_u^+]_{\mathcal{A}}$.

Definition ($\mathcal{A}$-contraction)

Given a tree $\mathcal{T}$, a factorization Π of $\mathcal{T}$, and an automaton $\mathcal{A}$, the $\mathcal{A}$-**contraction** $\overrightarrow{\mathcal{T}}$ of $\mathcal{T}$ is the tree obtained from Π by coloring its vertices u with the corresponding $\mathcal{A}$-type $[\mathcal{T}_u^+]_{\mathcal{A}}$.

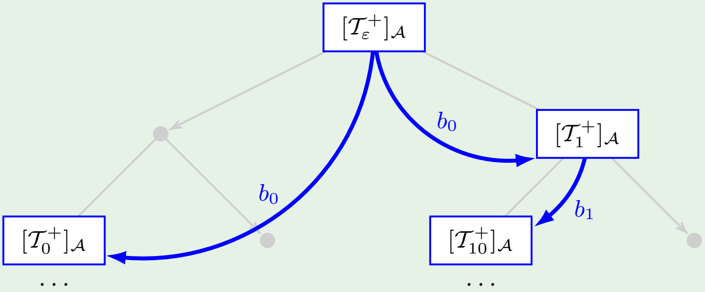
Example



Definition ($\mathcal{A}$-contraction)

Given a tree $\mathcal{T}$, a factorization Π of $\mathcal{T}$, and an automaton $\mathcal{A}$, the $\mathcal{A}$-**contraction** $\vec{\mathcal{T}}$ of $\mathcal{T}$ is the tree obtained from Π by coloring its vertices u with the corresponding $\mathcal{A}$-type $[\mathcal{T}_u^+]_{\mathcal{A}}$.

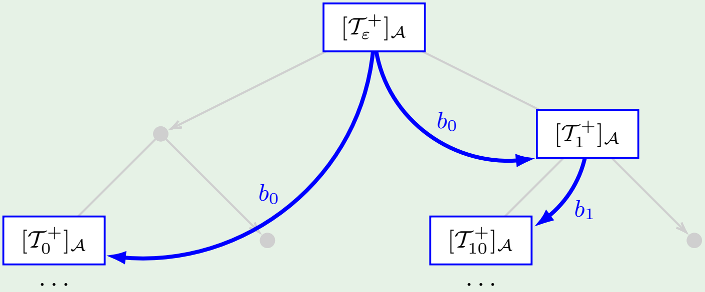
Example



Definition ($\mathcal{A}$-contraction)

Given a tree $\mathcal{T}$, a factorization Π of $\mathcal{T}$, and an automaton $\mathcal{A}$, the $\mathcal{A}$-**contraction** $\vec{\mathcal{T}}$ of $\mathcal{T}$ is the tree obtained from Π by coloring its vertices u with the corresponding $\mathcal{A}$-type $[\mathcal{T}_u^+]_\mathcal{A}$.

Example



Note: $\vec{\mathcal{T}}$ may have infinite out-degree ($\Rightarrow$ non-deterministic).

To reason on $\overrightarrow{\mathcal{T}}$ by tree automata, we must get rid of (tree) non-determinism.

To reason on $\vec{\mathcal{T}}$ by tree automata, we must get rid of (tree) non-determinism.

Idea

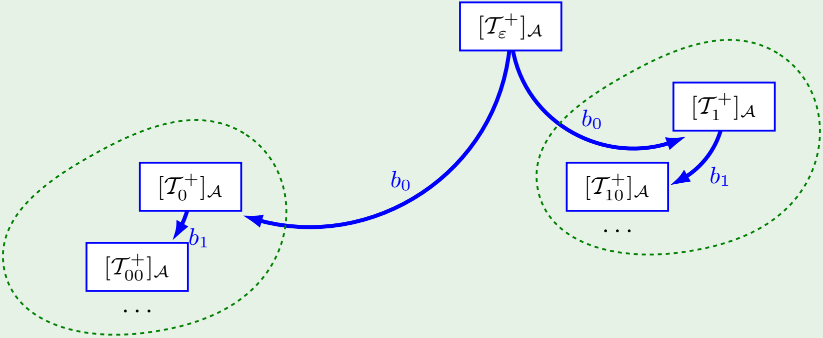
Collapse isomorphic subtrees in $\vec{\mathcal{T}}$. If this can be done for every pair of outgoing edges with the same label, then $\vec{\mathcal{T}}$ can be given the status of deterministic tree and we can give it in input to a tree automaton $\vec{\mathcal{A}}$.

To reason on $\overrightarrow{\mathcal{T}}$ by tree automata, we must get rid of (tree) non-determinism.

Idea

Collapse isomorphic subtrees in $\overrightarrow{\mathcal{T}}$. If this can be done for every pair of outgoing edges with the same label, then $\overrightarrow{\mathcal{T}}$ can be given the status of deterministic tree and we can give it in input to a tree automaton $\overrightarrow{\mathcal{A}}$.

Example

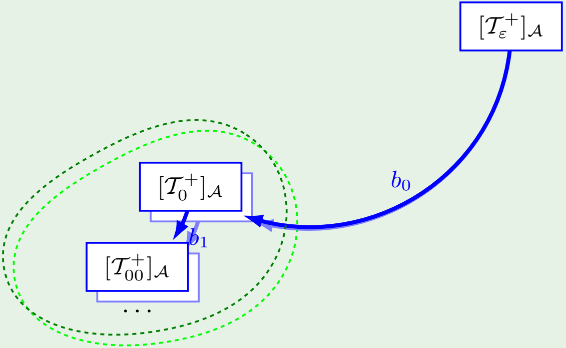


To reason on $\vec{\mathcal{T}}$ by tree automata, we must get rid of (tree) non-determinism.

Idea

Collapse isomorphic subtrees in $\vec{\mathcal{T}}$. If this can be done for every pair of outgoing edges with the same label, then $\vec{\mathcal{T}}$ can be given the status of deterministic tree and we can give it in input to a tree automaton $\vec{\mathcal{A}}$.

Example



Theorem (Montanari and Puppis '04)

If a tree $\mathcal{T}$ has a deterministic $\mathcal{A}$-contraction $\vec{\mathcal{T}}$ for any automaton $\mathcal{A}$, then one can build another Rabin tree automaton $\vec{\mathcal{A}}$, called **contraction automaton**, *such that*

$$\mathcal{T} \in \mathscr{L}(\mathcal{A}) \qquad \text{iff} \qquad \vec{\mathcal{T}} \in \mathscr{L}(\vec{\mathcal{A}})$$

Theorem (Montanari and Puppis '04)

If a tree $\mathcal{T}$ has a deterministic $\mathcal{A}$-contraction $\vec{\mathcal{T}}$ for any automaton $\mathcal{A}$, then one can build another Rabin tree automaton $\vec{\mathcal{A}}$, called **contraction automaton**, *such that*

$$\mathcal{T} \in \mathscr{L}(\mathcal{A}) \qquad \text{iff} \qquad \vec{\mathcal{T}} \in \mathscr{L}(\vec{\mathcal{A}})$$

Proof idea

- the input alphabet of $\vec{\mathcal{A}}$ consists of all $\mathcal{A}$-types plus a dummy symbol $\perp$ for missing b-labeled successors in $\vec{\mathcal{T}}$
- a transition of $\vec{\mathcal{A}}$ from a vertex colored by $[\mathcal{T}_u^+]_\mathcal{A}$ mimics a computation of $\mathcal{A}$ on the marked factor $\mathcal{T}_u^+$

  (note: this can be done since the $\mathcal{A}$-type $[\mathcal{T}_u^+]_\mathcal{A}$ is a finite object that completely characterizes the 'behavior' of $\mathcal{A}$ on the marked factor $\mathcal{T}_u^+$)

Corollary

Given a tree $\mathcal{T}$ and a Rabin tree automaton $\mathcal{A}$, if one can compute a (deterministic) $\mathcal{A}$-contraction $\overrightarrow{\mathcal{T}}$ with $Acc_{\overrightarrow{\mathcal{T}}}$ decidable (e.g., a regular $\mathcal{A}$-contraction), then $Acc_{\mathcal{T}}$ is decidable.

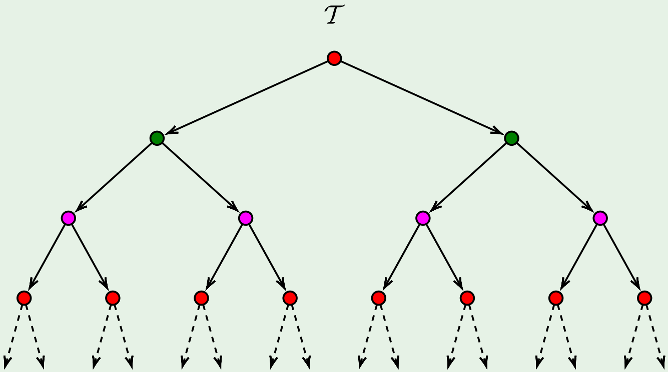
To summarize

to prove that the acceptance problem of a tree $\mathcal{T}$ is decidable:

1. provide a suitable factorization Π of $\mathcal{T}$
2. build the $\mathcal{A}$-contraction $\overrightarrow{\mathcal{T}}$ of $\mathcal{T}$ w.r.t. Π
3. show that $\overrightarrow{\mathcal{T}}$ is (bisimilar to) a deterministic tree
4. show that $Acc_{\overrightarrow{\mathcal{T}}}$ is decidable (e.g., show that $\overrightarrow{\mathcal{T}}$ is regular).

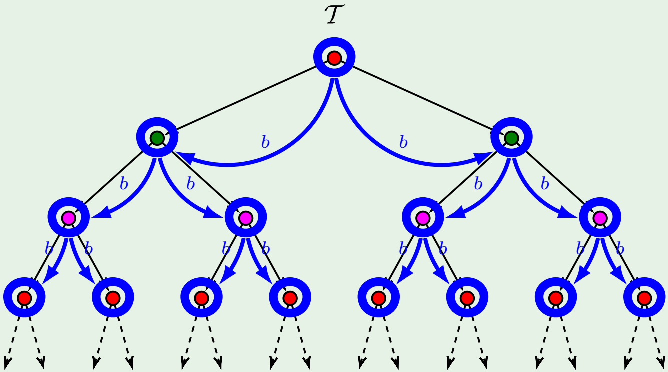
Example

Let $\mathcal{T}$ be a tree with homogeneously-colored levels



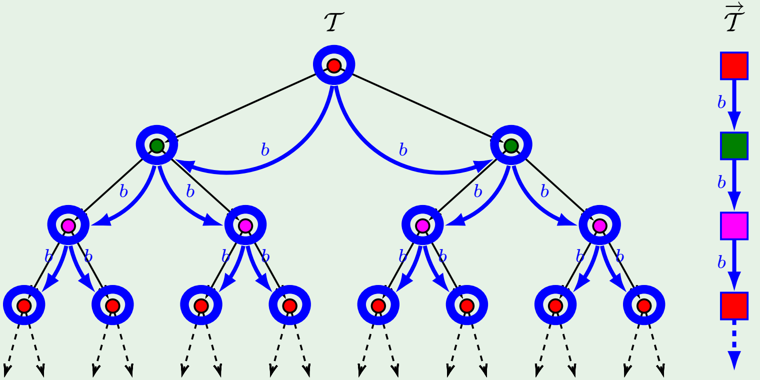
Example

Let $\mathcal{T}$ be a tree with homogeneously-colored levels
Π the factorization of $\mathcal{T}$ such that $\mathcal{D}om(\Pi) = \mathcal{D}om(\mathcal{T})$



Example

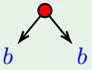
Let $\mathcal{T}$ be a tree with homogeneously-colored levels
Π the factorization of $\mathcal{T}$ such that $\mathcal{D}om(\Pi) = \mathcal{D}om(\mathcal{T})$
and $\vec{\mathcal{T}}$ a corresponding $\mathcal{A}$-contraction.

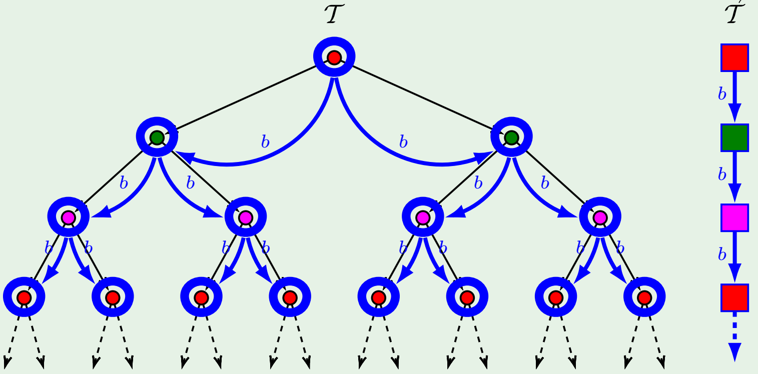


Example

Each vertex of the $\mathcal{A}$-contraction $\vec{\mathcal{T}}$

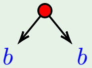
is colored with the $\mathcal{A}$-type of the corresponding factor



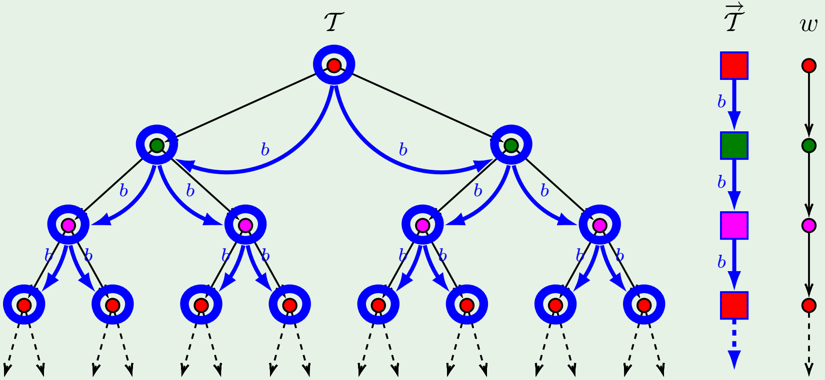


Example

Each vertex of the $\mathcal{A}$-contraction $\vec{\mathcal{T}}$ is colored with the $\mathcal{A}$-type of the corresponding factor

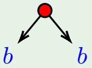


Acc_w decidable

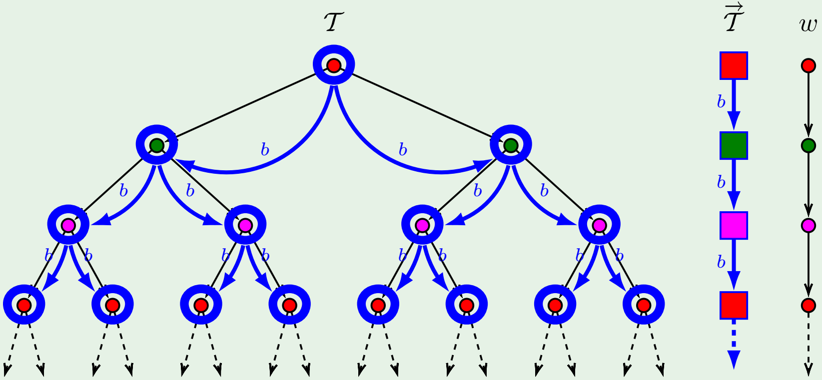


Example

Each vertex $\blacksquare$ of the $\mathcal{A}$-contraction $\vec{\mathcal{T}}$
is colored with the $\mathcal{A}$-type of the corresponding factor



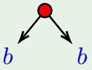
Acc_w decidable $\Rightarrow$ $Acc_{\vec{\mathcal{T}}}$ decidable



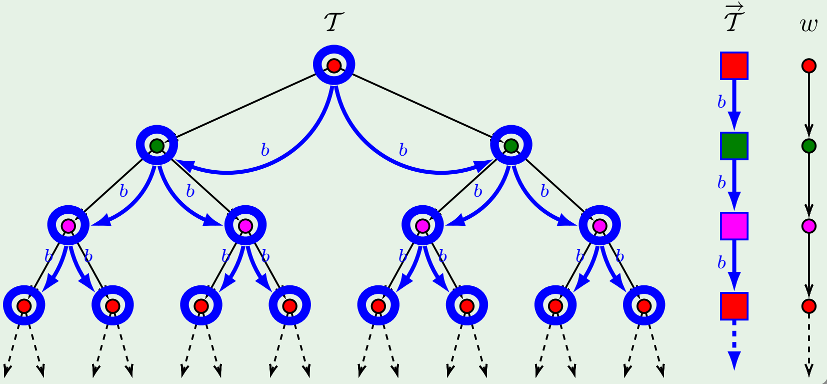
Example

Each vertex ■ of the $\mathcal{A}$-contraction $\vec{\mathcal{T}}$

is colored with the $\mathcal{A}$-type of the corresponding factor



Acc_w decidable $\Rightarrow$ $Acc_{\vec{\mathcal{T}}}$ decidable $\Rightarrow$ $Acc_{\mathcal{T}}$ decidable



Definition (Second-order tree substitution)

The **second-order tree substitution** $\mathcal{C}[\![\mathcal{T}/x]\!]$ is the replacement of each x-colored vertex in $\mathcal{C}$ with $\mathcal{T}$.

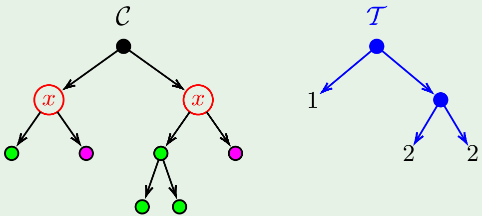
(*a suitable marking on the leaves of $\mathcal{T}$ is used to specify the attachment points for the subtrees rooted at the successors of a replacement occurrence*).

Definition (Second-order tree substitution)

The **second-order tree substitution** $\mathcal{C}[\![\mathcal{T}/x]\!]$ is the replacement of each x-colored vertex in $\mathcal{C}$ with $\mathcal{T}$.

(*a suitable marking on the leaves of* $\mathcal{T}$ *is used to specify the attachment points for the subtrees rooted at the successors of a replacement occurrence*).

Example

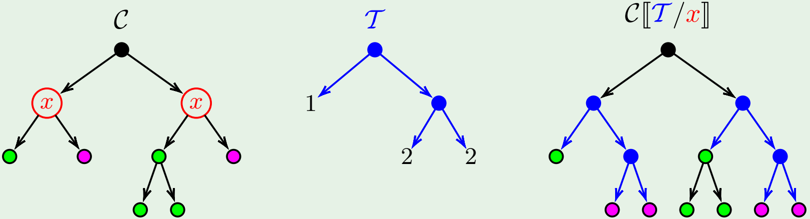


Definition (Second-order tree substitution)

The **second-order tree substitution** $\mathcal{C}[\![\mathcal{T}/x]\!]$ is the replacement of each x-colored vertex in $\mathcal{C}$ with $\mathcal{T}$.

(*a suitable marking on the leaves of $\mathcal{T}$ is used to specify the attachment points for the subtrees rooted at the successors of a replacement occurrence*).

Example



Theorem

Second-order tree substitutions respect the $\mathcal{A}$-types:

$$[\mathcal{T}]_{\mathcal{A}} = [\mathcal{T}']_{\mathcal{A}} \quad \Rightarrow \quad \left[\mathcal{C}[\![\mathcal{T}/x]\!]\right]_{\mathcal{A}} = \left[\mathcal{C}[\![\mathcal{T}'/x]\!]\right]_{\mathcal{A}}$$

Theorem

Second-order tree substitutions respect the $\mathcal{A}$-types:

$$[\mathcal{T}]_{\mathcal{A}} = [\mathcal{T}']_{\mathcal{A}} \quad \Rightarrow \quad \left[\mathcal{C}[\![\mathcal{T}/x]\!]\right]_{\mathcal{A}} = \left[\mathcal{C}[\![\mathcal{T}'/x]\!]\right]_{\mathcal{A}}$$

Corollary

Given an automaton $\mathcal{A}$ and a function γ such that

$$\gamma(\mathcal{T}) = \mathcal{C}[\![\mathcal{T}/x]\!] \qquad \textit{for any (marked) tree } \mathcal{T}$$

there is a well-defined (computable) function $\gamma_{\mathcal{A}}$ such that

$$\gamma_{\mathcal{A}}([\mathcal{T}]_{\mathcal{A}}) = \left[\gamma(\mathcal{T})\right]_{\mathcal{A}} \qquad \textit{for any (marked) tree } \mathcal{T}$$

Theorem

Second-order tree substitutions respect the $\mathcal{A}$-types:

$$[\mathcal{T}]_{\mathcal{A}} = [\mathcal{T}']_{\mathcal{A}} \quad \Rightarrow \quad \left[\mathcal{C}[\![\mathcal{T}/x]\!]\right]_{\mathcal{A}} = \left[\mathcal{C}[\![\mathcal{T}'/x]\!]\right]_{\mathcal{A}}$$

Corollary

Given an automaton $\mathcal{A}$ and a function γ such that

$$\gamma(\mathcal{T}) = \mathcal{C}[\![\mathcal{T}/x]\!] \qquad \text{for any (marked) tree } \mathcal{T}$$

there is a well-defined (computable) function $\gamma_{\mathcal{A}}$ such that

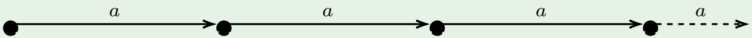
$$\gamma_{\mathcal{A}}([\mathcal{T}]_{\mathcal{A}}) = \left[\gamma(\mathcal{T})\right]_{\mathcal{A}} \qquad \text{for any (marked) tree } \mathcal{T}$$

The set of all functions $\gamma_{\mathcal{A}}$ with functional composition $\circ$ is a

finite monoid

Example

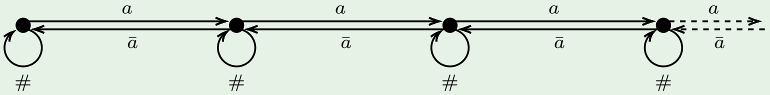
Let $\mathcal{L}$ be the semi-infinite line



Example

Let $\mathcal{L}$ be the semi-infinite line

we extend it with **backward edges** and **loops**



Example

Let $\mathcal{L}$ be the semi-infinite line

we extend it with **backward edges** and **loops**

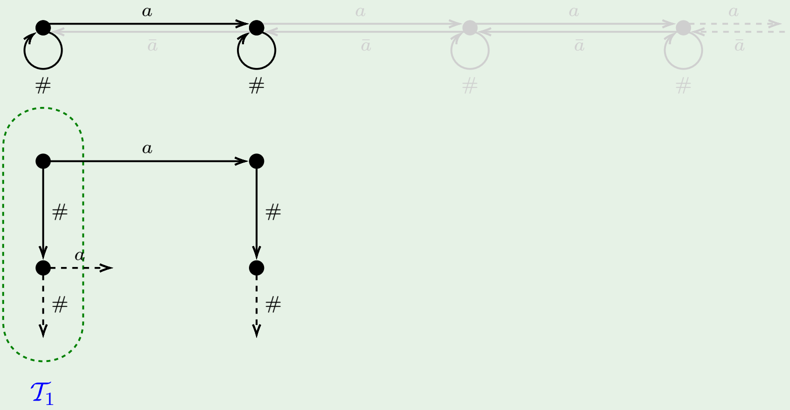
and we unfold the structure from the leftmost vertex.

Example

Let $\mathcal{L}$ be the semi-infinite line

we extend it with **backward edges** and **loops**

and we unfold the structure from the leftmost vertex.

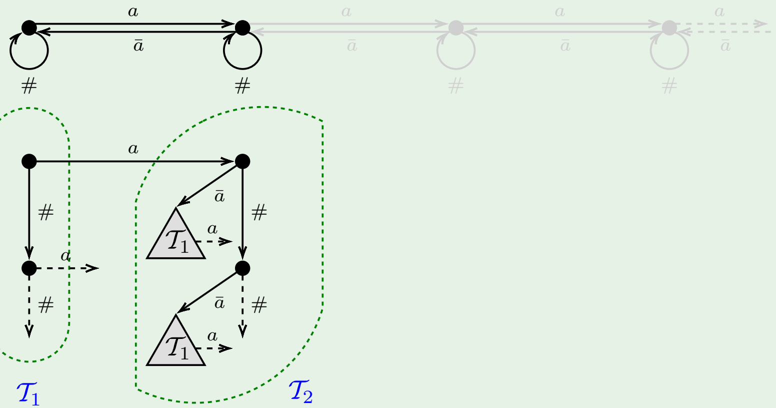


Example

Let $\mathcal{L}$ be the semi-infinite line

we extend it with **backward edges** and **loops**

and we unfold the structure from the leftmost vertex.

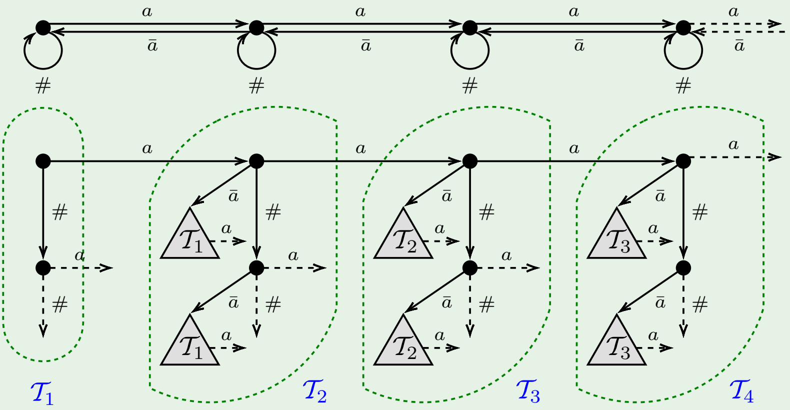


Example

Let $\mathcal{L}$ be the semi-infinite line

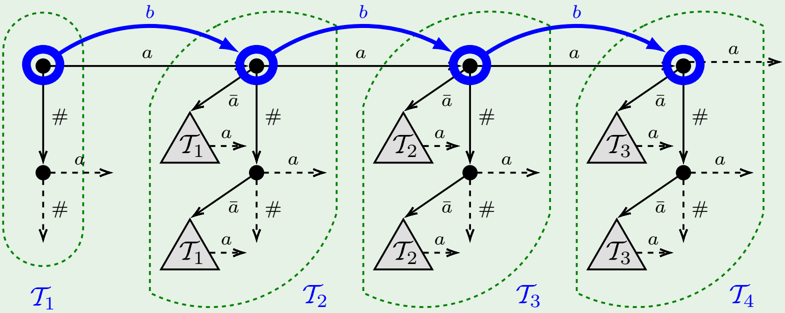
we extend it with **backward edges** and **loops**

and we unfold the structure from the leftmost vertex.



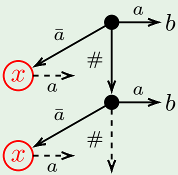
Example

We now define the following factorization:

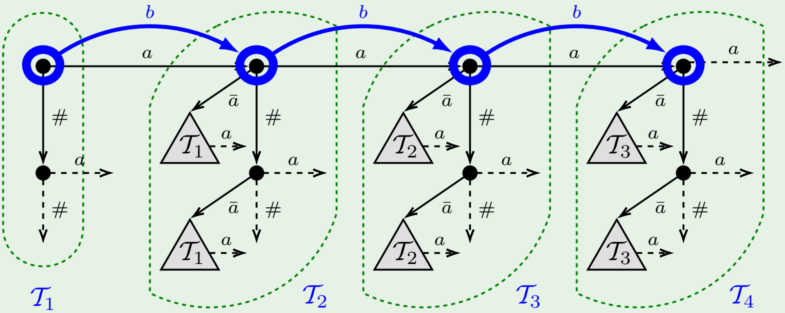


Example

Each factor $\mathcal{T}_{n+1}$ is obtained from the

second-order tree substitution in 

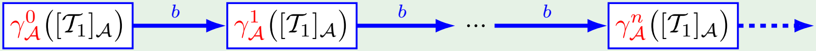
of x by $\mathcal{T}_n$.

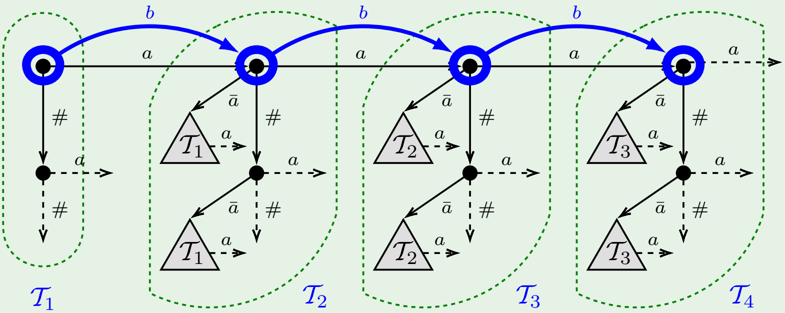


Example

We can write $\mathcal{T}_{n+1} = \gamma(\mathcal{T}_n)$, hence $[\mathcal{T}_{n+1}]_{\mathcal{A}} = \gamma^n_{\mathcal{A}}([\mathcal{T}_1]_{\mathcal{A}})$

$\Rightarrow$ the $\mathcal{A}$-contraction is a regular tree of the form





We just saw an example of a reduction to a regular contraction.

However, one can iterate reductions in order to show that the acceptance problem of a tree $\mathcal{T}$ is decidable ...

We just saw an example of a reduction to a regular contraction.

However, one can iterate reductions in order to show that the acceptance problem of a tree $\mathcal{T}$ is decidable ...

Example

Consider the problem of deciding if $\mathcal{T} \in \mathscr{L}(\mathcal{A})$:

If $\mathcal{T}$ has an $\mathcal{A}$-contraction $\vec{\mathcal{T}}$, and
$\vec{\mathcal{T}}$ has a *regular* $\vec{\mathcal{A}}$-contraction $\vec{\vec{\mathcal{T}}}$

Then we can decide if $\vec{\vec{\mathcal{T}}} \in \mathscr{L}(\vec{\vec{\mathcal{A}}})$, $\vec{\mathcal{T}} \in \mathscr{L}(\vec{\mathcal{A}})$, and $\mathcal{T} \in \mathscr{L}(\mathcal{A})$.

Definition (Reducible trees)

It comes natural to define a

hierarchy of recursively reducible trees:

- **rank** 0 **trees** := regular trees
- **rank** $n+1$ **trees** := trees enjoying a rank n $\mathcal{A}$-contraction,
  for any Rabin tree automaton $\mathcal{A}$.

Definition (Reducible trees)

It comes natural to define a

hierarchy of recursively reducible trees:

- **rank** 0 **trees** $:=$ regular trees
- **rank** $n+1$ **trees** $:=$ trees enjoying a rank n $\mathcal{A}$-contraction,
  for any Rabin tree automaton $\mathcal{A}$.

Theorem

The acceptance problem of any recursively reducible tree is decidable.

Closure properties of rank n trees

By exploiting the inductive structure of rank n trees, one can show that, for any $n \in \mathbb{N}$, rank n trees are closed under:

- **rational colorings**

  specified by regular path expressions (like rational restrictions and inverse rational mappings)

  (alternative specifications in terms of Mealy tree automata, namely, deterministic tree automata with an output function)

Closure properties of rank n trees

By exploiting the inductive structure of rank n trees, one can show that, for any $n \in \mathbb{N}$, rank n trees are closed under:

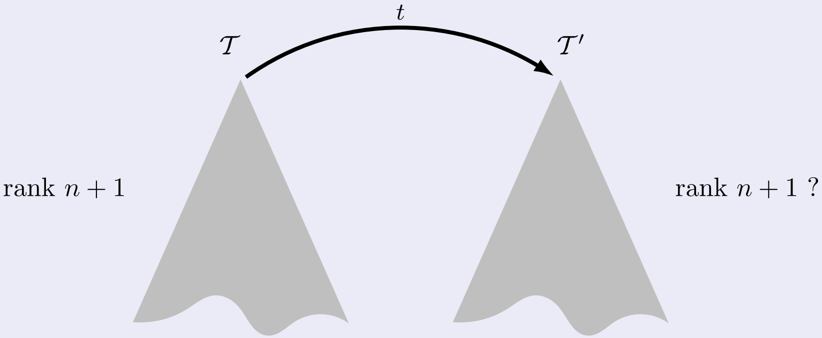
- **rational colorings**
  specified by regular path expressions (like rational restrictions and inverse rational mappings)
  (alternative specifications in terms of Mealy tree automata, namely, deterministic tree automata with an output function)
- **rational colorings with bounded lookahead**
  rational colorings extended with the facility of inspecting the subtree issued from the current position, up to a bounded depth

Closure properties of rank n trees

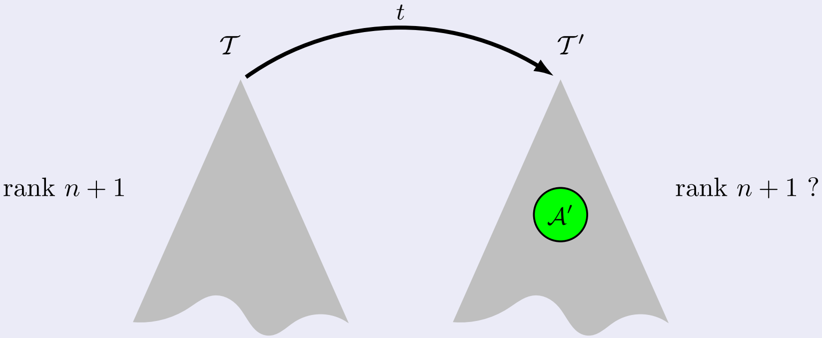
By exploiting the inductive structure of rank n trees, one can show that, for any $n \in \mathbb{N}$, rank n trees are closed under:

- **rational colorings**
  specified by regular path expressions (like rational restrictions and inverse rational mappings)
  (alternative specifications in terms of Mealy tree automata, namely, deterministic tree automata with an output function)
- **rational colorings with bounded lookahead**
  rational colorings extended with the facility of inspecting the subtree issued from the current position, up to a bounded depth
- **regular tree morphisms**
  specified by a tuple of regular trees $(\mathcal{T}_x)_{x \in X}$
  and mapping an input tree $\mathcal{C}$ to $\mathcal{C}[\![\mathcal{T}_x / x]\!]_{x \in X}$

Proof idea (closure properties of rank n trees)

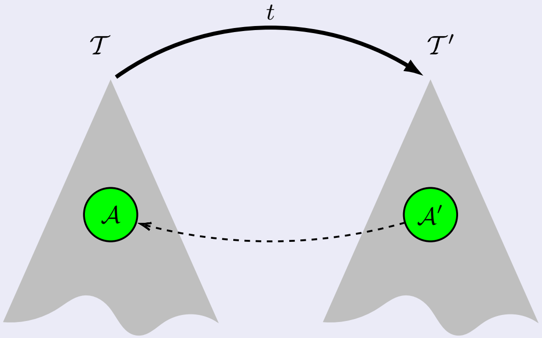


Proof idea (closure properties of rank n trees)



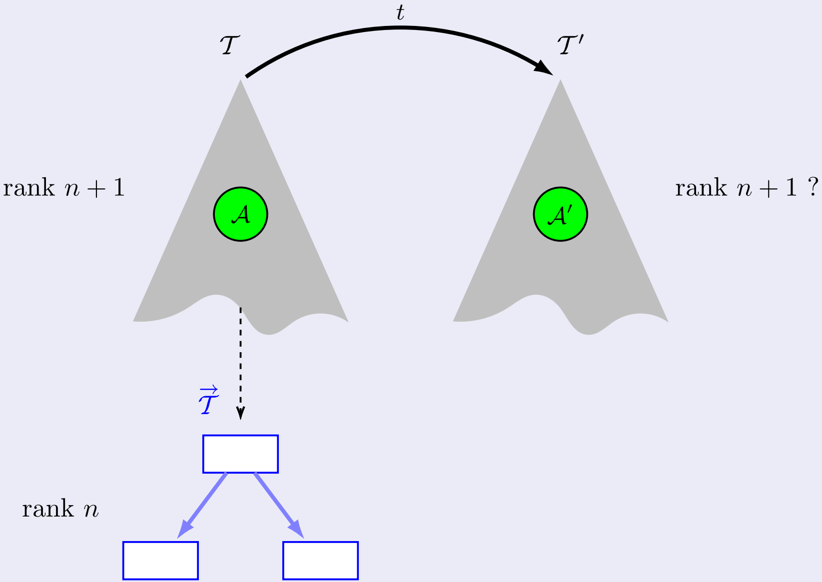
Proof idea (closure properties of rank n trees)

rank $n+1$

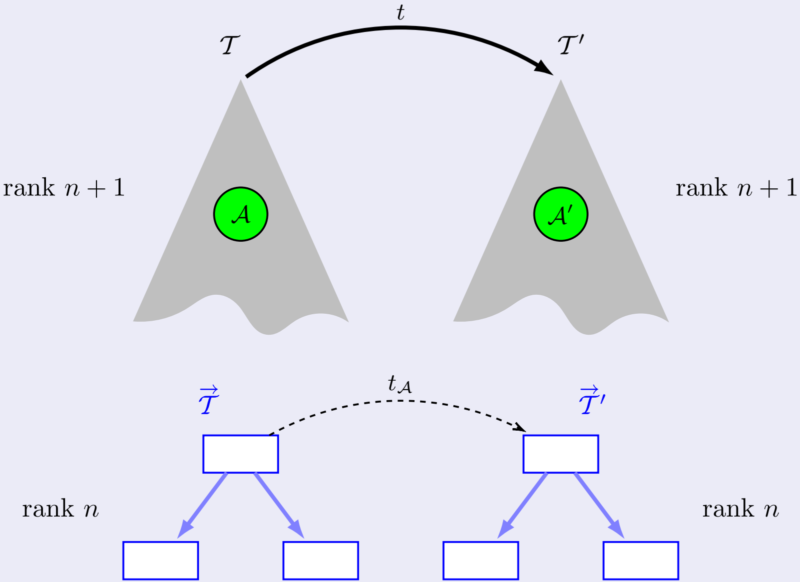


rank $n+1$?

Proof idea (closure properties of rank n trees)



Proof idea (closure properties of rank n trees)



Go