Synthesis of Transducers from Automatic Specifications

Christof Löding

RWTH Aachen University, Germany

based on joint work with Sarah Winter in the CASSTING project

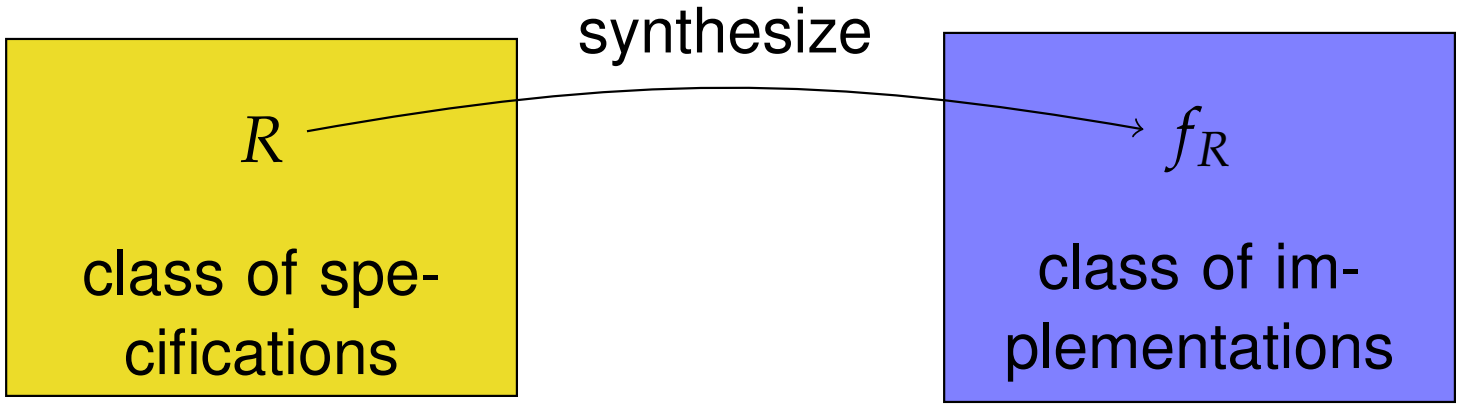
Trends in Tree Automata and Tree Transducers
ETAPS Workshop
London, April 18, 2015

General Problem Setting

Given a (binary) relation R (in this talk over words or trees), synthesize a function f_R

- that has the same domain as R, and
- whose graph is a subset of R.

We say that f_R implements (or realizes, or uniformizes) the specification R.



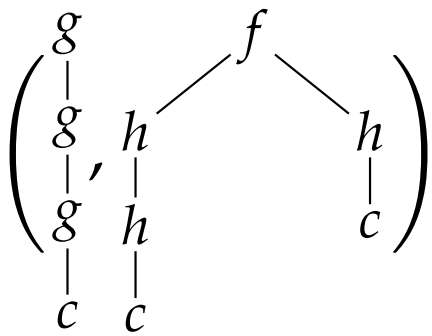
Specification: Relation over Finite Trees

Example:

Input alphabet: $g(), c, d$

Output alphabet: $f(,), h(), c, d$

$$\begin{array}{ll} & \{(g^n(c), f(h^{n-1}(c), h^m(c)) \mid n \geq 1, m \geq 0\} \\ \cup & \{(g^n(d), f(h^{n-1}(d), h^m(d)) \mid n \geq 1, m \geq 0\} \end{array}$$

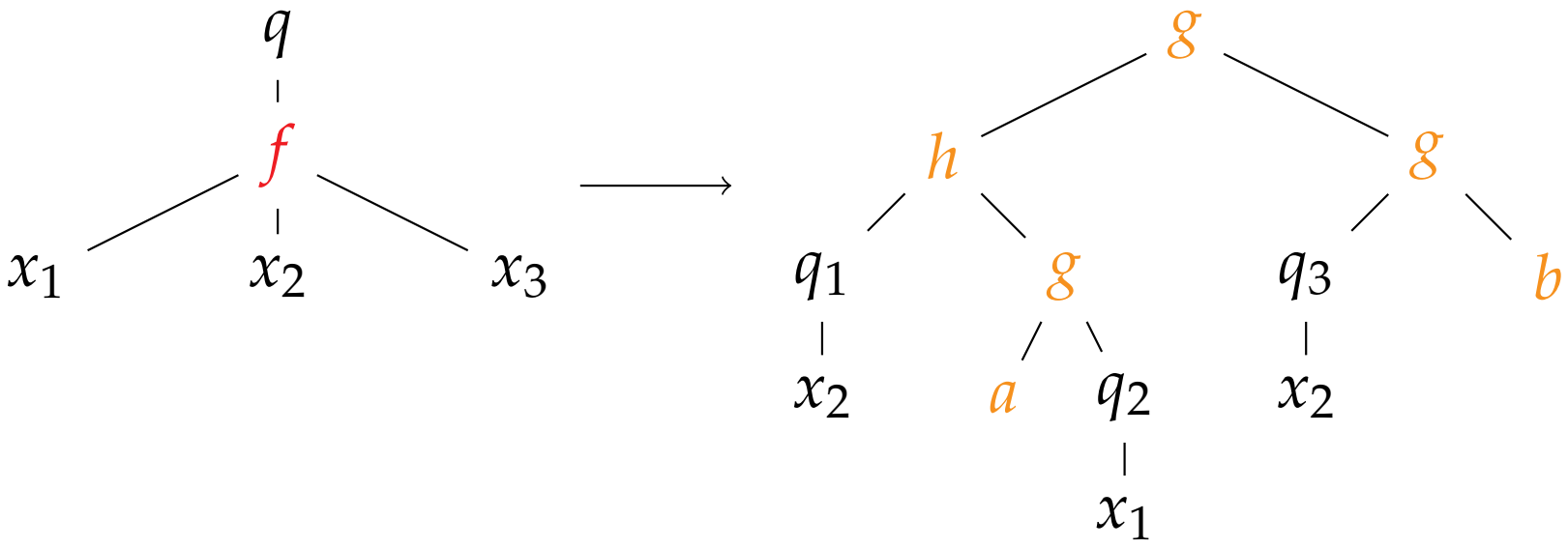


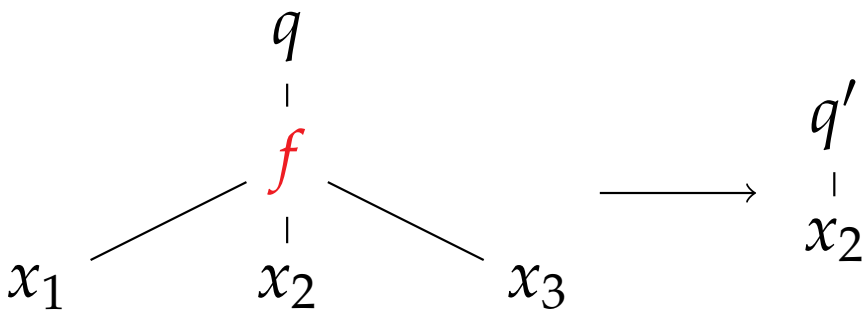
left component corresponds to input trees

right component to allowed output trees

Implementations: Deterministic Top-Down Tree Transducer

Transitions are of the following form (example):





Example

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

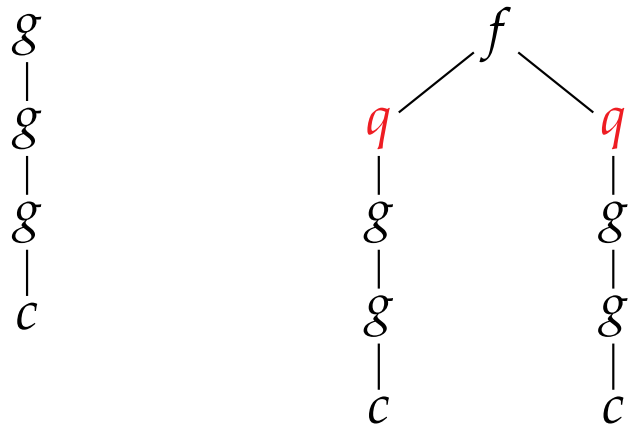
Execution:

Example

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:

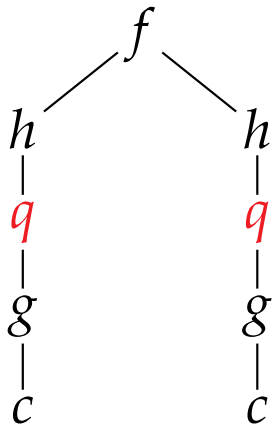


Example

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:

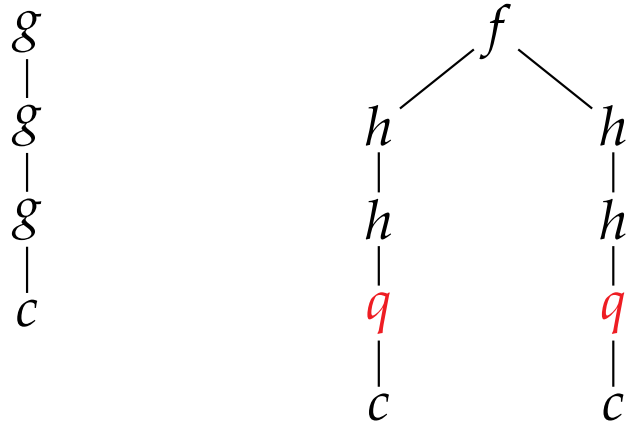


Example

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:

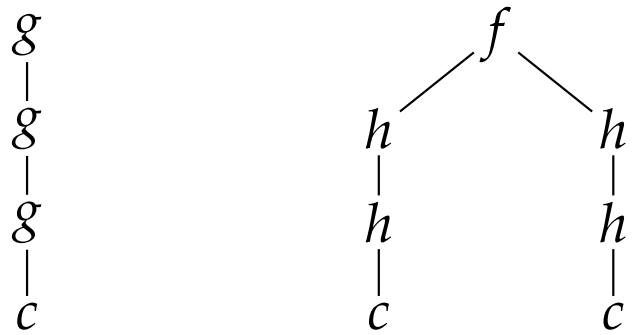


Example

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:

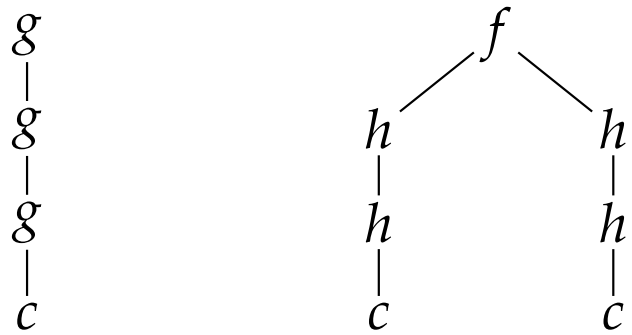


Example

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\to f(q(x_1), q(x_1)) \\ q(g(x_1)) &\to h(q((x_1)) \\ q(c) &\to c \\ q(d) &\to d \end{aligned}$$

Execution:



Implements

$$\begin{aligned} & \{(g^n(c), f(h^{n-1}(c), h^m(c)) \mid n \geq 1, m \geq 0\} \\ \cup \quad & \{(g^n(d), f(h^{n-1}(d), h^m(d)) \mid n \geq 1, m \geq 0\} \end{aligned}$$

Question

For which classes of specifications is the synthesis problem for DTDTT decidable?

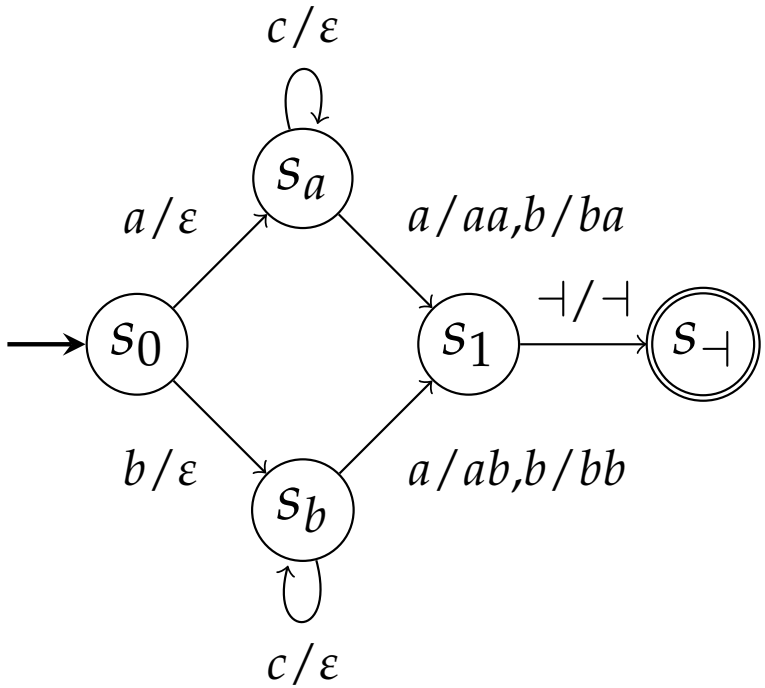
- Is it decidable if we take nondeterministic TDTT as specifications?
  No, already undecidable for words.
- We identify some decidable cases of the synthesis problem.

Outline

Implementations: Word Transducers

A deterministic asynchronous word transducer

- is deterministic on the input
- produces an output word (possibly empty) in each step
- uses $\dashv$ as end-of-word-marker

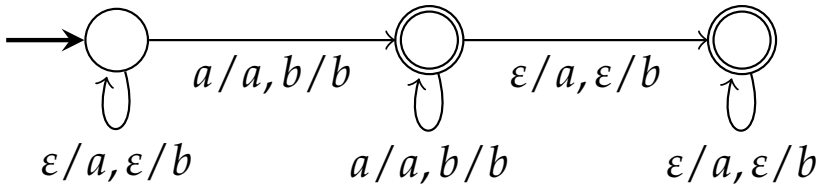


Specifications: Rational Relations

Nondeterministic asynchronous automata: transitions are labeled with pairs of words x/y or $\binom{x}{y}$

$\rightsquigarrow$ rational relations

A rational relation: $R = \{(v, w) \mid v \text{ is a non-empty infix of } w\}$



Undecidability for Rational Relations

Theorem. (Carayol/L.) For a given rational relation it is undecidable whether it can be realized by a deterministic asynchronous transducer.

Undecidability for Rational Relations

Theorem. (Carayol/L.) For a given rational relation it is undecidable whether it can be realized by a deterministic asynchronous transducer.

Idea: Reduction from PCP (proposed by E. Filiot and I. Jecker)

PCP instance $(u_1, \ldots, u_k)$ and $(v_1, \ldots, v_n)$

Specification:

input: $i_0 i_1 \cdots i_n \#^m A / B$

output: $w\#^m$

- If last letter of input is A, then $w = u_{i_1} \cdots u_{i_n}$
- If last letter of input is B, then $w \neq v_{i_1} \cdots v_{i_n}$

Undecidability for Rational Relations

Theorem. (Carayol/L.) For a given rational relation it is undecidable whether it can be realized by a deterministic asynchronous transducer.

Idea: Reduction from PCP (proposed by E. Filiot and I. Jecker)

PCP instance $(u_1, \ldots, u_k)$ and $(v_1, \ldots, v_n)$

Specification:

input: $i_0 i_1 \cdots i_n \#^m A / B$
output: $w\#^m$

- If last letter of input is A, then $w = u_{i_1} \cdots u_{i_n}$
- If last letter of input is B, then $w \neq v_{i_1} \cdots v_{i_n}$

Implementation:

- PCP has no solution: output $i_j \mapsto u_{i_j}$, $\# \mapsto \#$
- PCP has solution $i_0 \cdots i_n$: it is necessary to know the last input letter to produce a valid output sequence. $\rightsquigarrow$ no finite state transducer

Restricting the Specifications

Synchronous automata: transitions are labeled with pairs of letters a/b or $\binom{a}{b}$ (using a padding symbol for the shorter word).

$\rightsquigarrow$ automatic relations

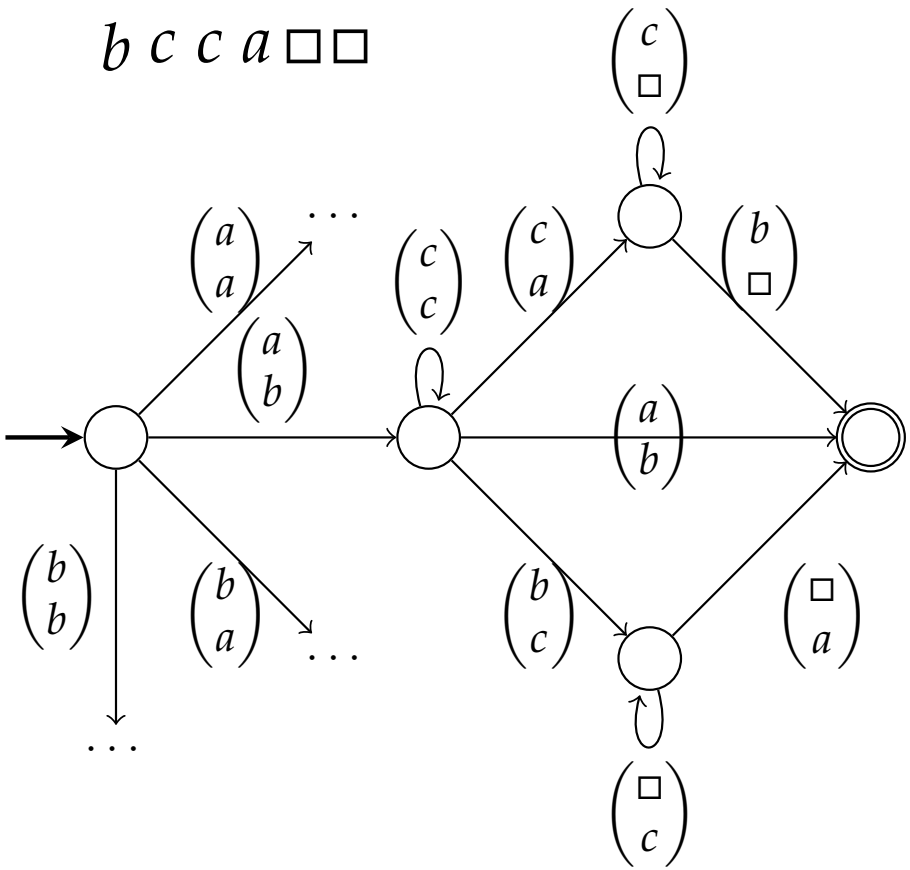
Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

"swap first and last letter"

$a\ c\ c\ c\ c\ b$

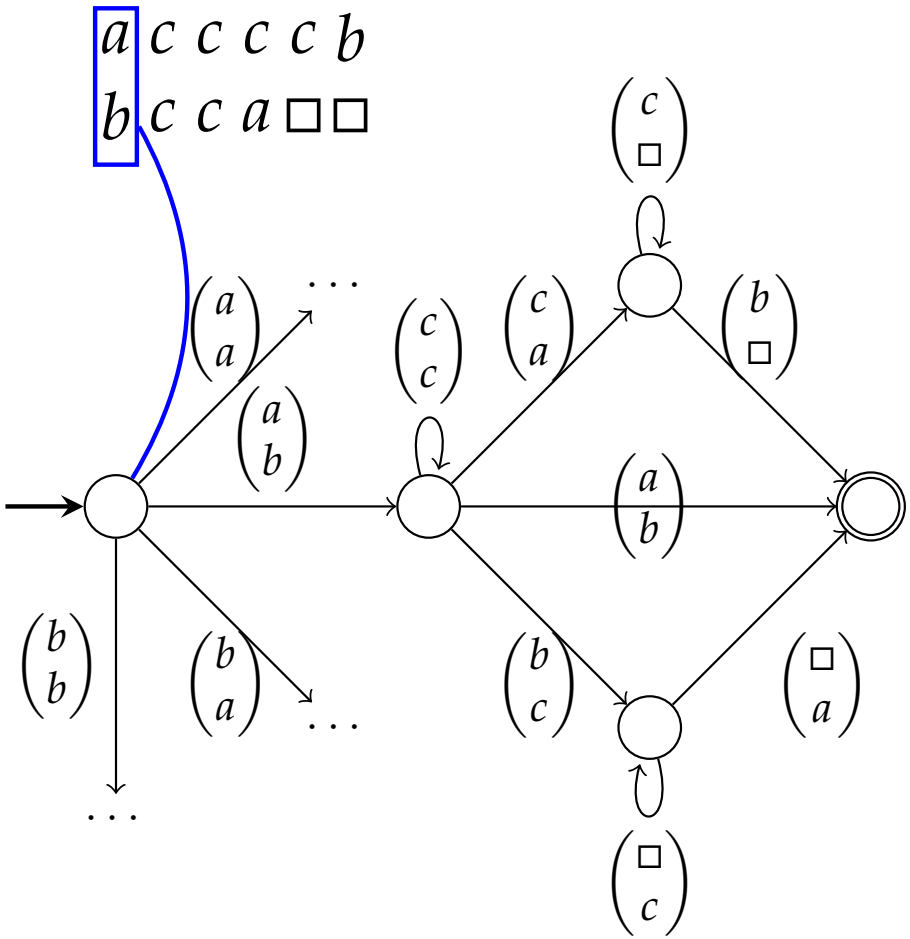
$b\ c\ c\ a\ \square\ \square$



Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

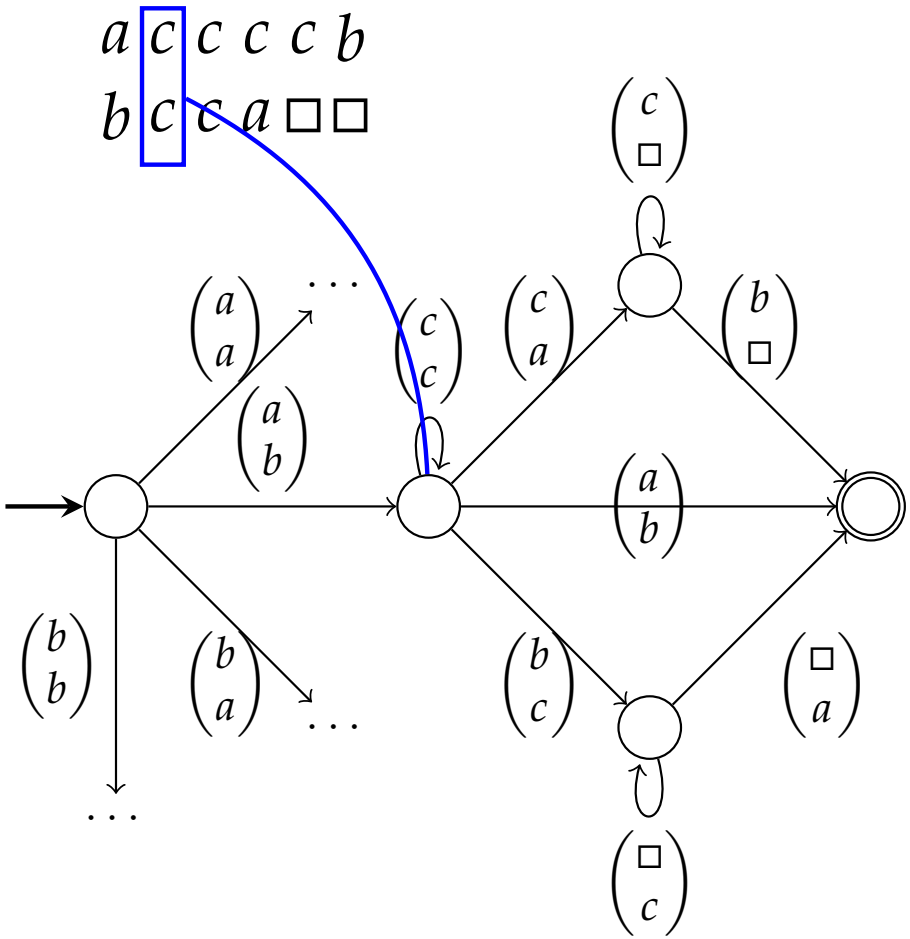
"swap first and last letter"



Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

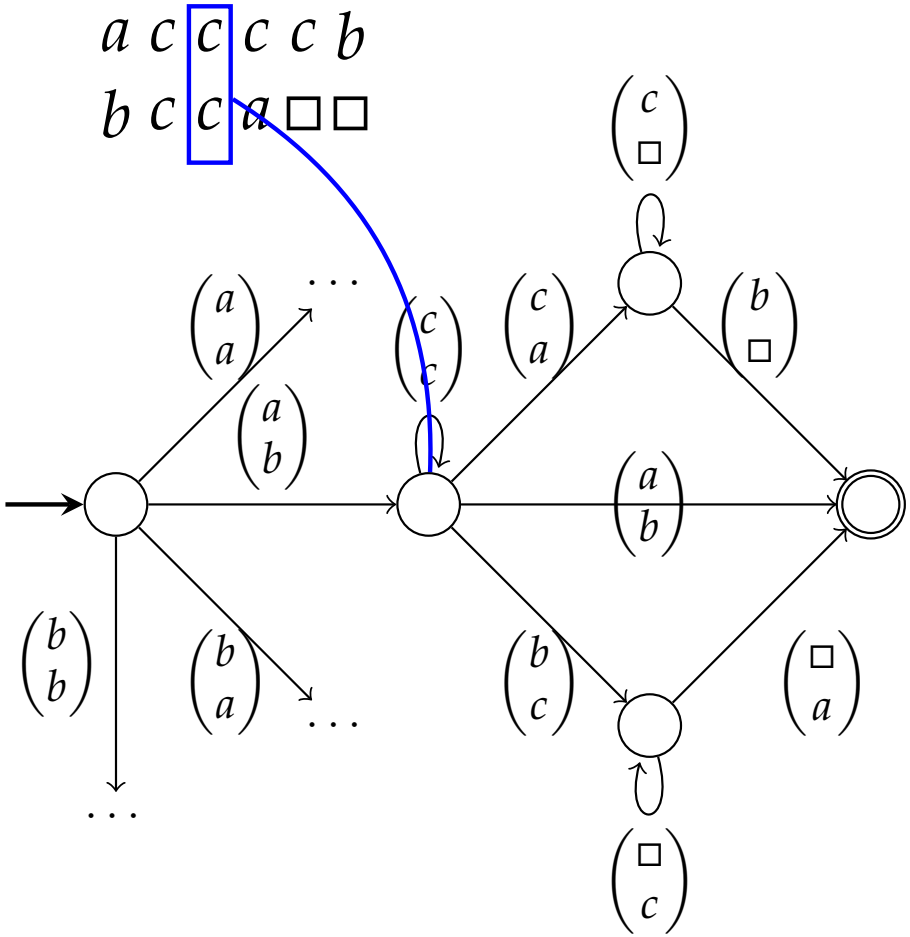
"swap first and last letter"



Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

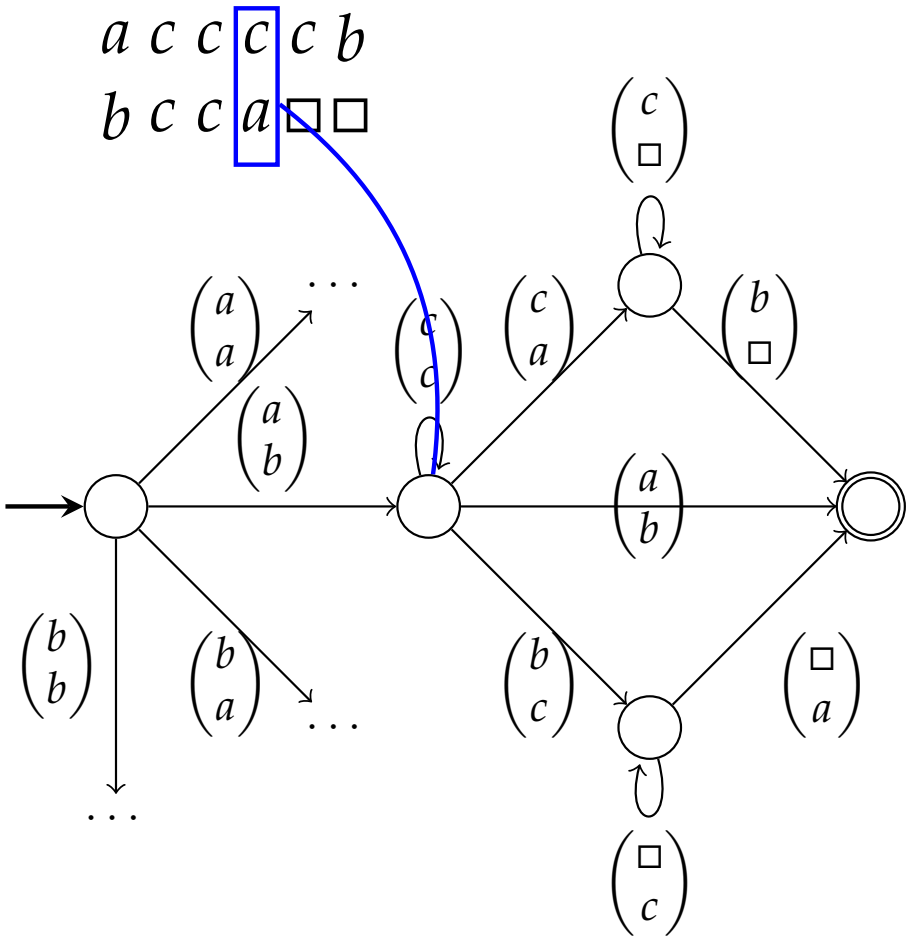
"swap first and last letter"



Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

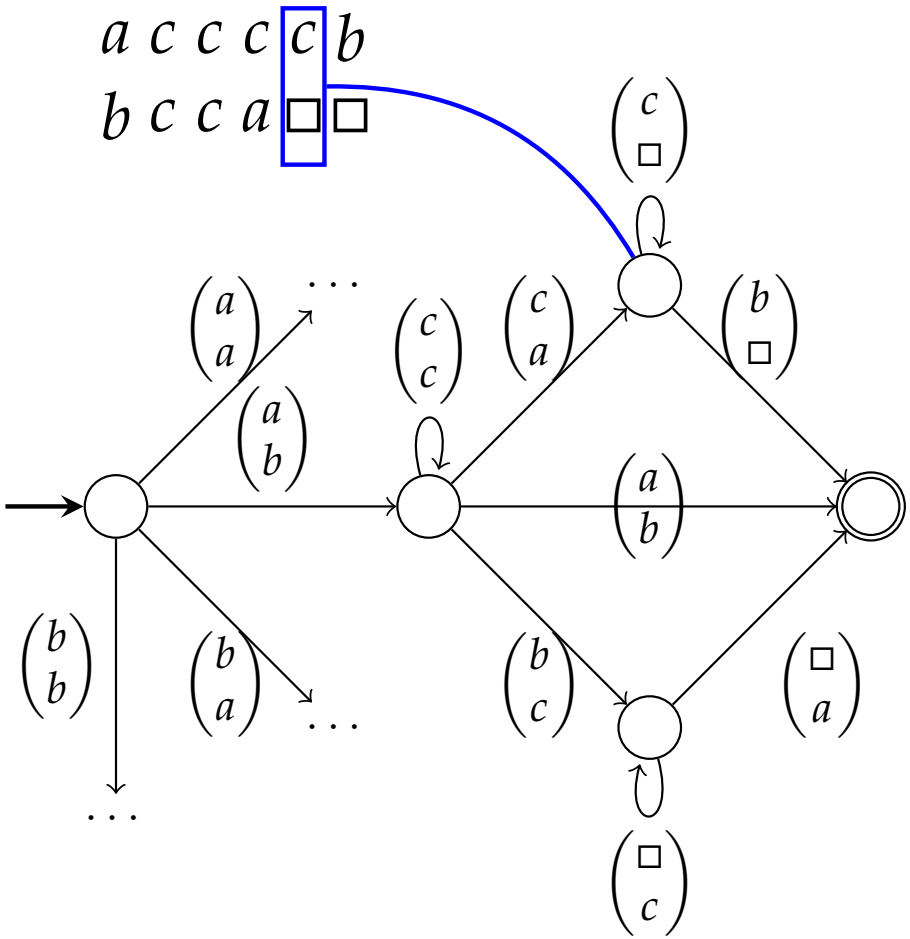
"swap first and last letter"



Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

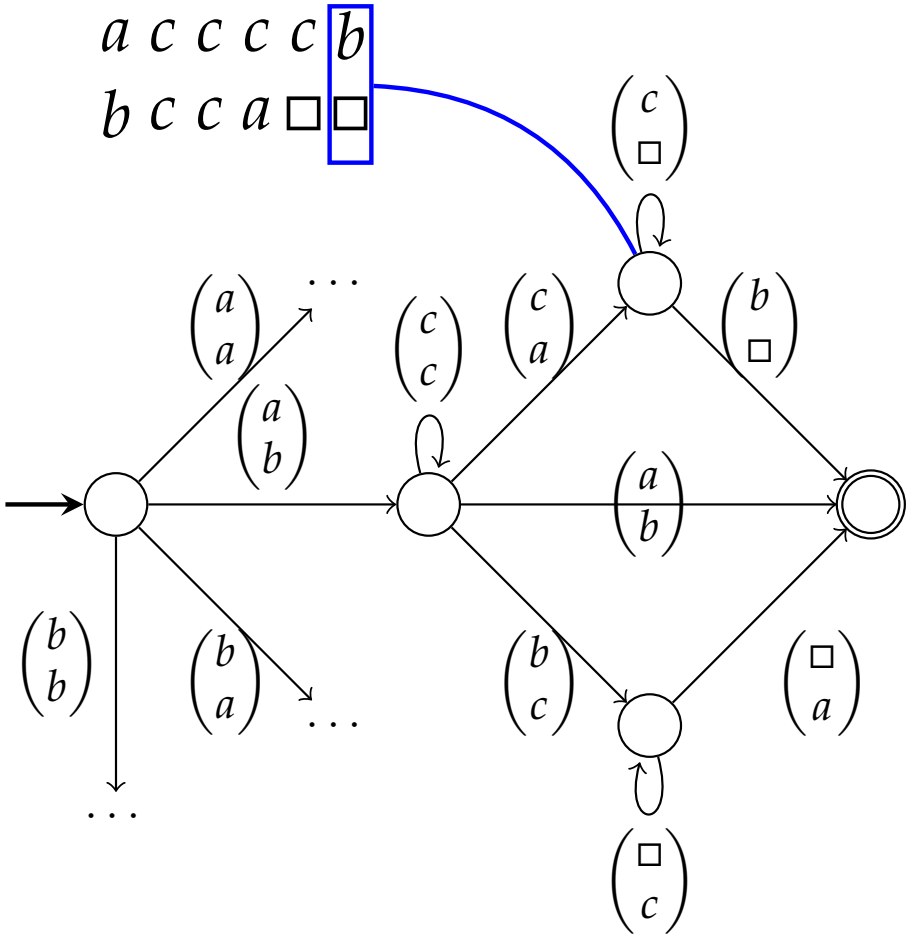
"swap first and last letter"



Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

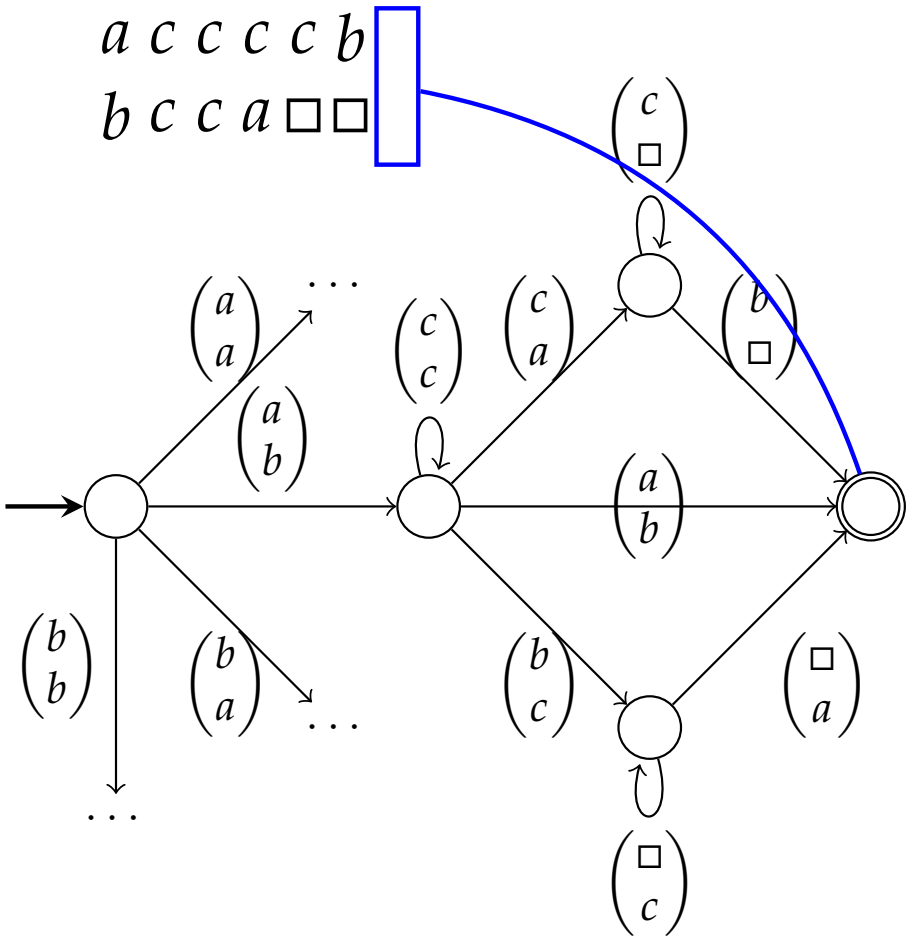
"swap first and last letter"



Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

"swap first and last letter"

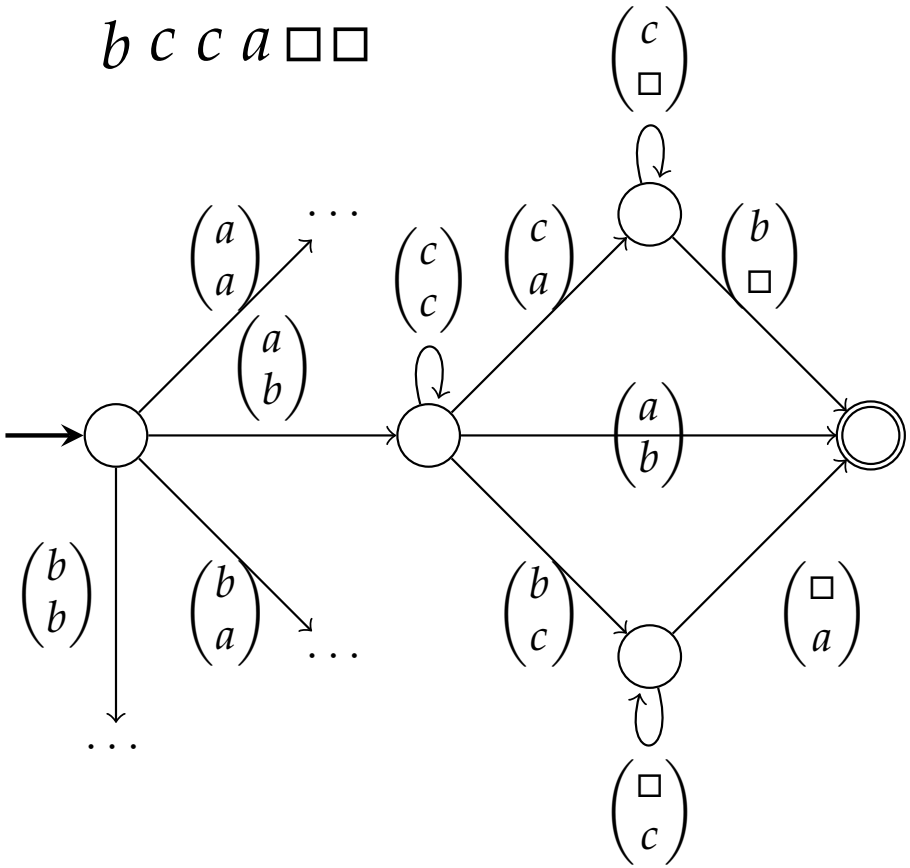


Example

$R = \{(xc^n y, yc^m x) \mid n, m \geq 0 \text{ and } x, y \in \{a, b\}\}$

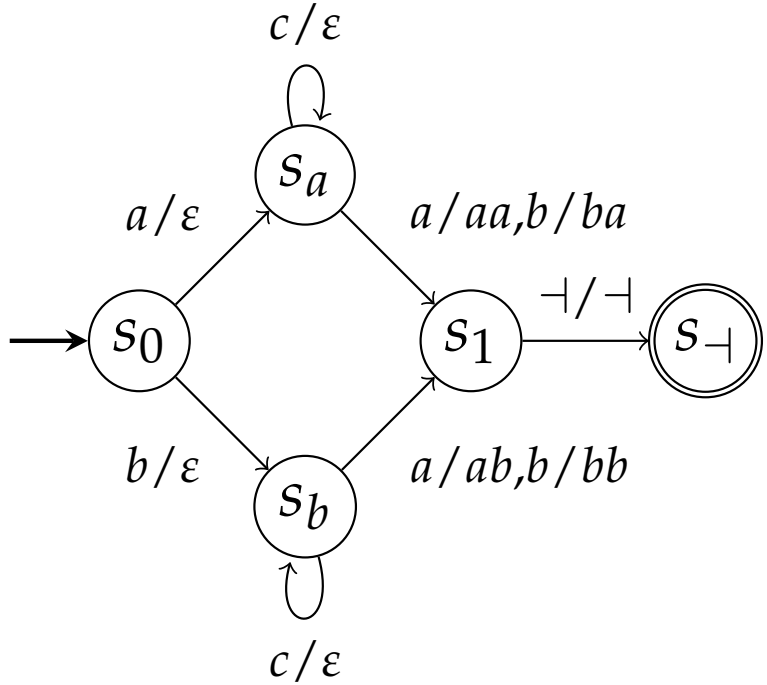
"swap first and last letter"

$a\ c\ c\ c\ c\ b$
$b\ c\ c\ a\ \square\ \square$



Realization by sequential transducer:

$xc^*y \mapsto yx$

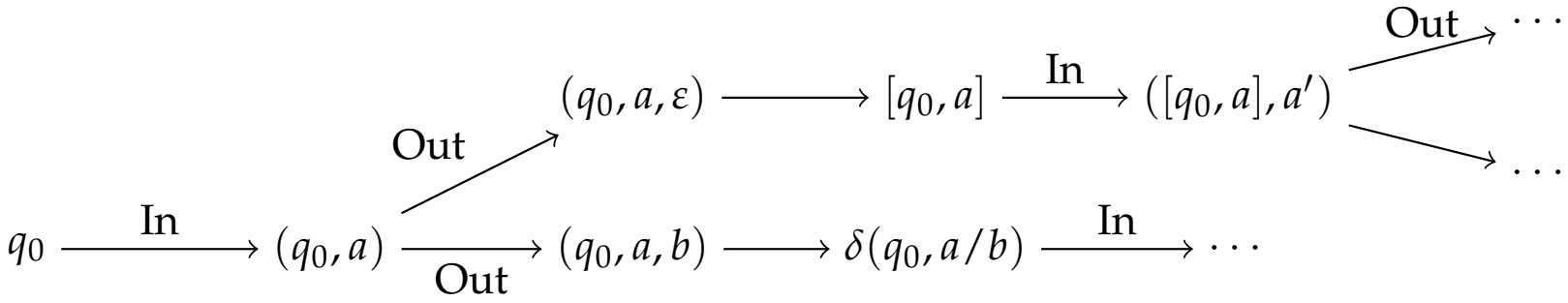


Decidability

Theorem (Carayol/L.). For an automatic specification it is decidable whether it can be realized by a deterministic asynchronous transducer.

Proof idea:

- Use a game between input and output, executing the transitions of the deterministic specification automaton along the moves:

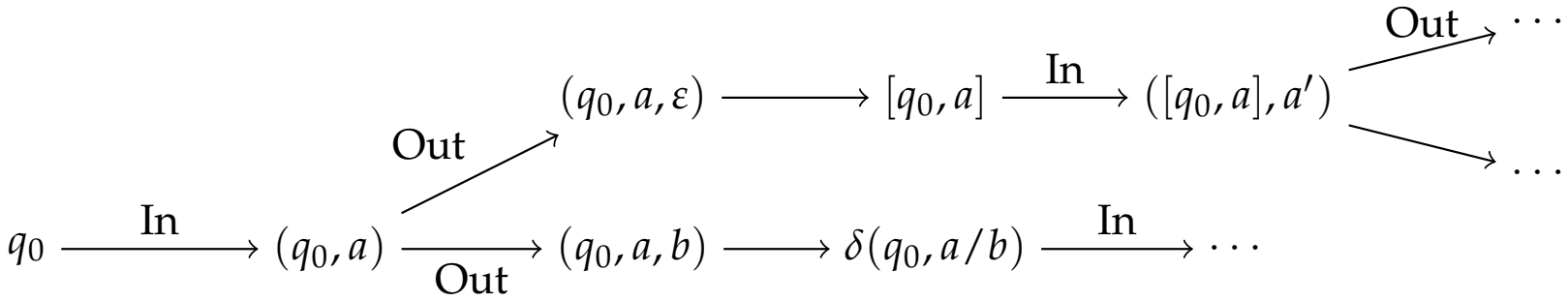


Decidability

Theorem (Carayol/L.). For an automatic specification it is decidable whether it can be realized by a deterministic asynchronous transducer.

Proof idea:

- Use a game between input and output, executing the transitions of the deterministic specification automaton along the moves:



- Problem: the delay cannot be bounded
- Solution: if the delay exceeds a certain bound K, then output can wait until the end, and play a last move based on some regular information of the input.

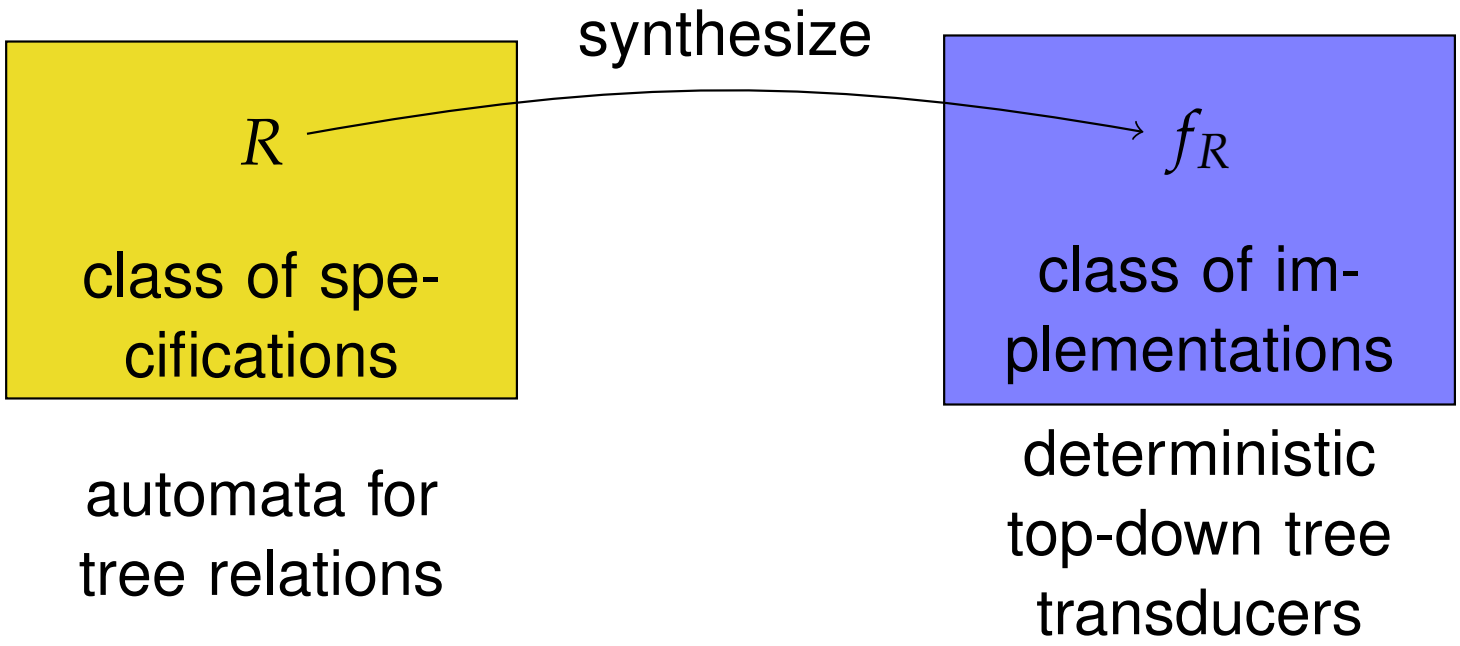
Outline

1. The String Case

2. Synthesis from Synchronous (Automatic) Specifications
   - Deterministic Specifications
   - Nondeterministic Specifications

3. Synthesis from Nondeterministic TDTT

Problem Setting for Trees



Undecidability in the General Case

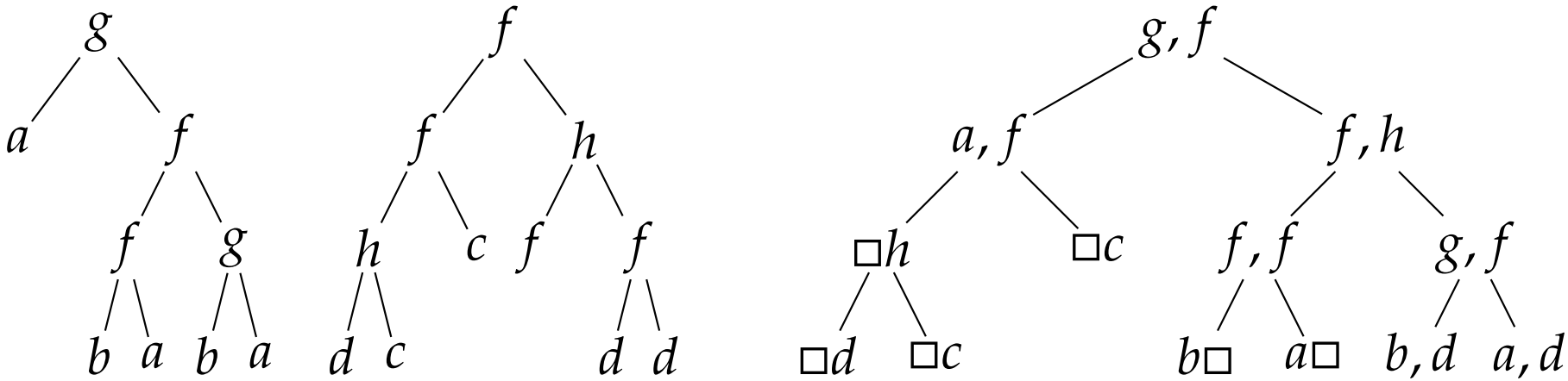
Corollary. It is undecidable whether a given NTDTT can be realized by a DTDTT.

- NTDTT restricted to words $\rightsquigarrow$ rational relations
- DTDTT restricted to words $\rightsquigarrow$ deterministic asynchronous word transducer

We analyze what happens if we restrict to automatic relations.

Automatic Relations of Finite Trees

How to combine trees?



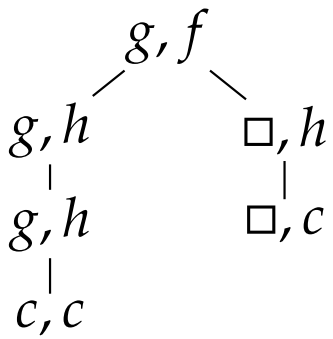
finite tree automata on such combined trees $\rightsquigarrow$ automatic relations over trees

Example

Input alphabet: $g(), c, d$

Output alphabet: $f(,), h(), c, d$

Relation :
$$\{(g^n(c), f(h^{n-1(c)}, h^m(c)) \mid n \geq 1, m \geq 0\}$$
$$\cup \quad \{(g^n(d), f(h^{n-1}(d), h^m(d)) \mid n \geq 1, m \geq 0\}$$



Can be accepted by a nondeterministic top-down automaton (needs to guess the leaf symbol).

Note: determinism ≠ nondeterminism for top-down automata

Possible Implementation

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

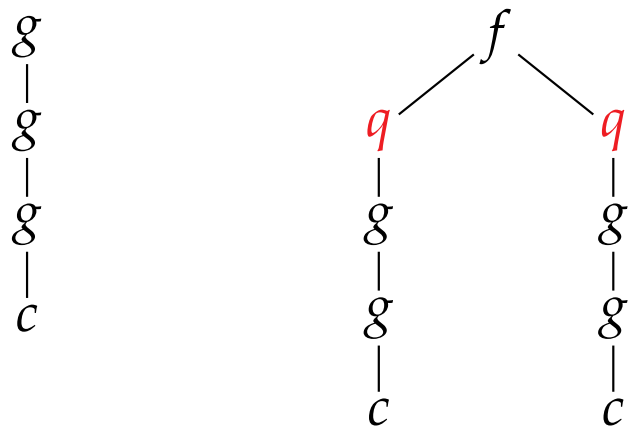
Execution:

Possible Implementation

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:

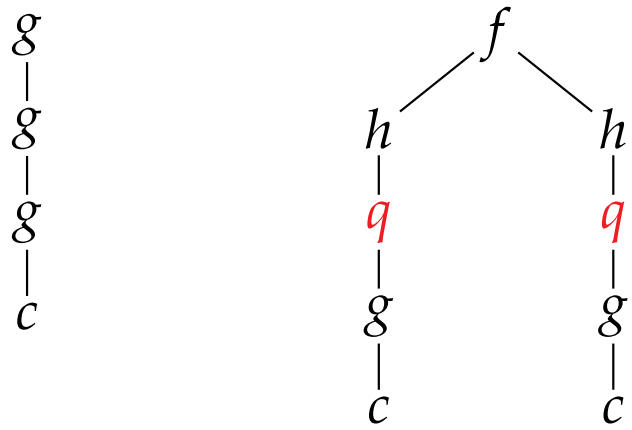


Possible Implementation

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\to f(q(x_1), q(x_1)) \\ q(g(x_1)) &\to h(q((x_1)) \\ q(c) &\to c \\ q(d) &\to d \end{aligned}$$

Execution:

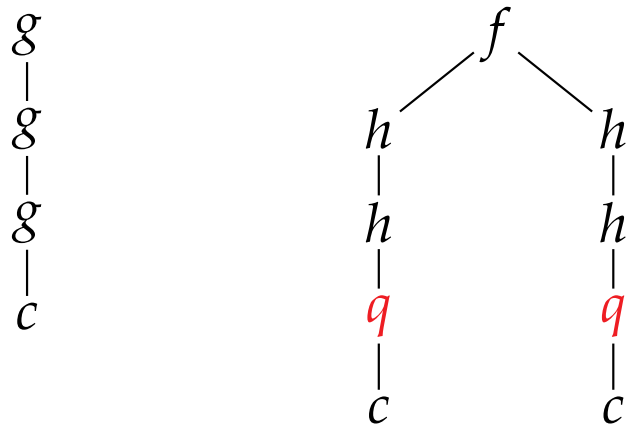


Possible Implementation

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:

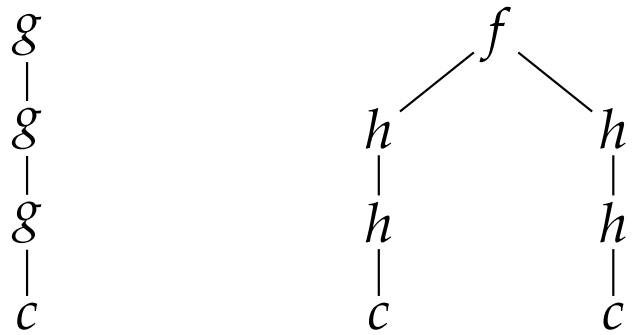


Possible Implementation

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:

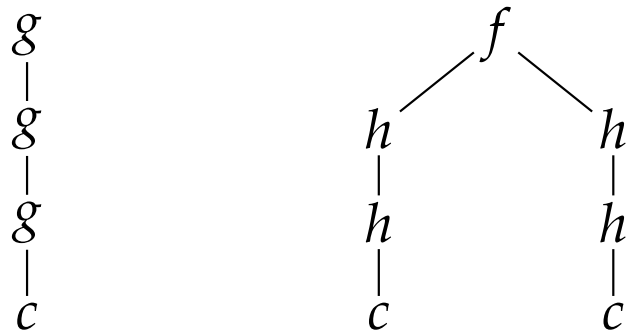


Possible Implementation

Transitions:

$$\begin{aligned} q_0(q(x_1)) &\rightarrow f(q(x_1), q(x_1)) \\ q(g(x_1)) &\rightarrow h(q((x_1)) \\ q(c) &\rightarrow c \\ q(d) &\rightarrow d \end{aligned}$$

Execution:



Needs to be non-linear because the two output leaf symbols have to be the same as the input leaf symbol.

Outline

Deterministic Specifications

Theorem (L./Winter'14). It is decidable, given an automatic tree relation defined by a deterministic top-down tree automaton, whether it can be implemented by a DTDTT.

Idea:

- Extend game methods from word case.
- If a DTDTT implementation exists, there is one that does not swap or copy subtrees.

Note: We assume that the transducer needs not to verify the domain of the relation. If an input tree does not have an image under the relation, then the transducer can produce arbitrary output.

The Game

- Player In and Out
- The game starts in the initial state of the specification automaton $\mathcal{A}$.
- From a state q of this automaton, the following moves are possible:
  $q \xrightarrow{\text{In}} (q, f) \xrightarrow{\text{Out}} (q, (f, g)) \xrightarrow{\text{In}} q_i$
  for the transition $(q, (f, g), q_1, \ldots, q_n)$ of $\mathcal{A}$
- A play corresponds to a path through the pair of input and output tree. Deterministic top-down languages are characterized by their paths.

The Game

- Player In and Out
- The game starts in the initial state of the specification automaton $\mathcal{A}$.
- From a state q of this automaton, the following moves are possible:
  $q \xrightarrow{\text{In}} (q, f) \xrightarrow{\text{Out}} (q, (f, g)) \xrightarrow{\text{In}} q_i$
  for the transition $(q, (f, g), q_1, \ldots, q_n)$ of $\mathcal{A}$
- A play corresponds to a path through the pair of input and output tree. Deterministic top-down languages are characterized by their paths.
- As in the word case, player Out can also skip (produce no output in this move).
- The “input buffer” resulting from the skip moves can be bounded similar to the word case.

Including Domain Verification

Relation:

t_a : only a at leafs

t : arbitrary tree

Implementation:

Top-down transducer only accepting valid input trees:

$$q_0(f(x_1, x_2)) \to f(q_a(x_2), q_b(x_1))$$

Swap and then copy subtrees, verifying the shape of the input.

Without swapping: Transducer cannot read the right subtree of the input.

Outline

Problems with Nondeterminism

Nondeterminism does not combine well with games:

$$q \xrightarrow{\text{In}} (q, f) \xrightarrow{\text{Out}} (q, (f, g)) \xrightarrow{\text{In}} q_i$$

for the transition $(q, (f, g), q_1, \ldots, q_n)$ of $\mathcal{A}$

The players would have to resolve the nondeterminism of $\mathcal{A}$; winning strategies do not correspond to implementations of the specifications anymore.

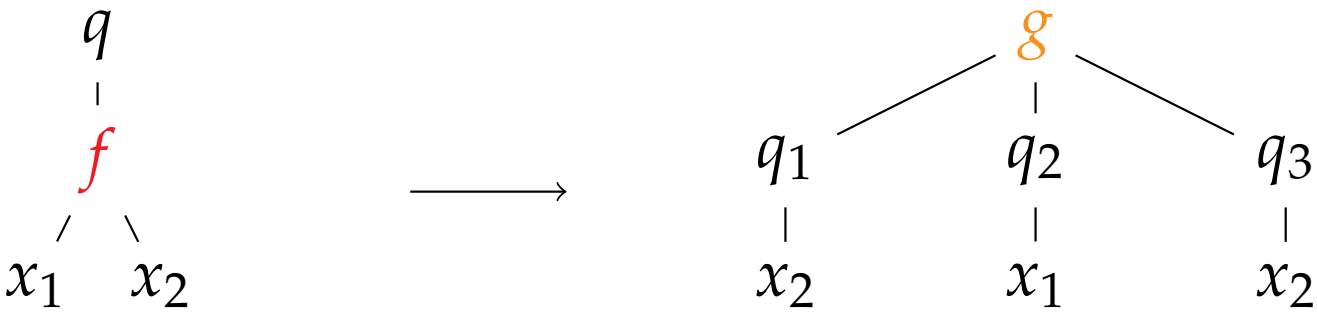
Restricted Cases

specification	implementation
deterministic	general DTDTT
finite unions of domain disjoint deterministic	synchronous DTDTT
nondeterministic	relabelling DTDTT

Restricted Cases

specification	implementation
deterministic	general DTDTT
finite unions of domain disjoint deterministic	synchronous DTDTT
nondeterministic	relabelling DTDTT

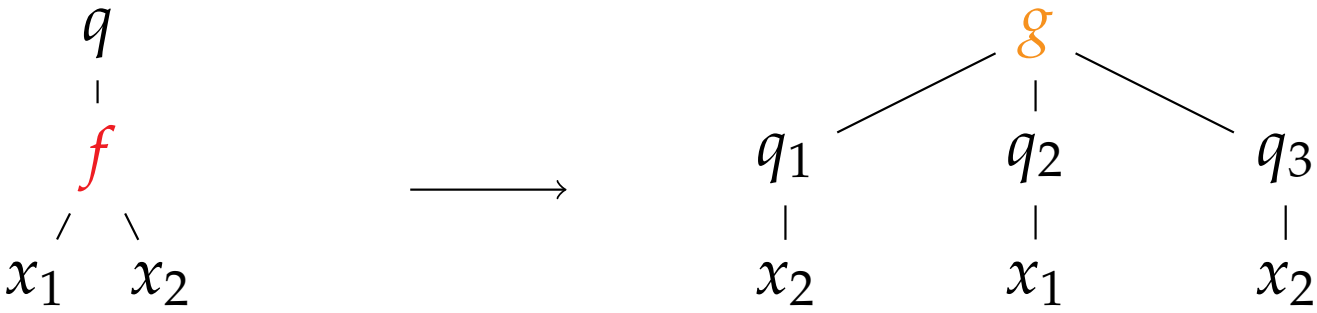
Synchronous top-down transducer: variables at depth 1



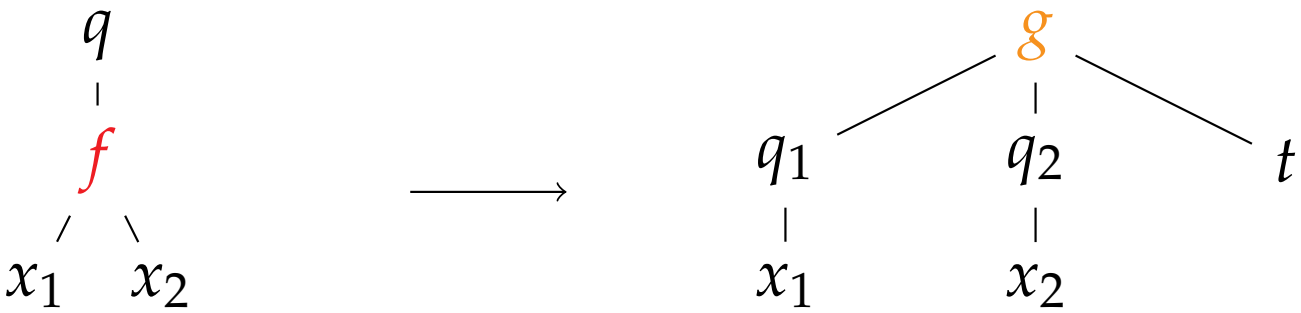
Restricted Cases

specification	implementation
deterministic	general DTDTT
finite unions of domain disjoint deterministic	synchronous DTDTT
nondeterministic	relabelling DTDTT

Synchronous top-down transducer: variables at depth 1



Relabelling top-down transducer: only relabels the nodes



General Nondeterministic Specifications

Theorem (L./Winter). It is decidable, given an automatic tree relation, whether it can be implemented by a relabelling DTDTT.

General Nondeterministic Specifications

Theorem (L./Winter). It is decidable, given an automatic tree relation, whether it can be implemented by a relabelling DTDTT.

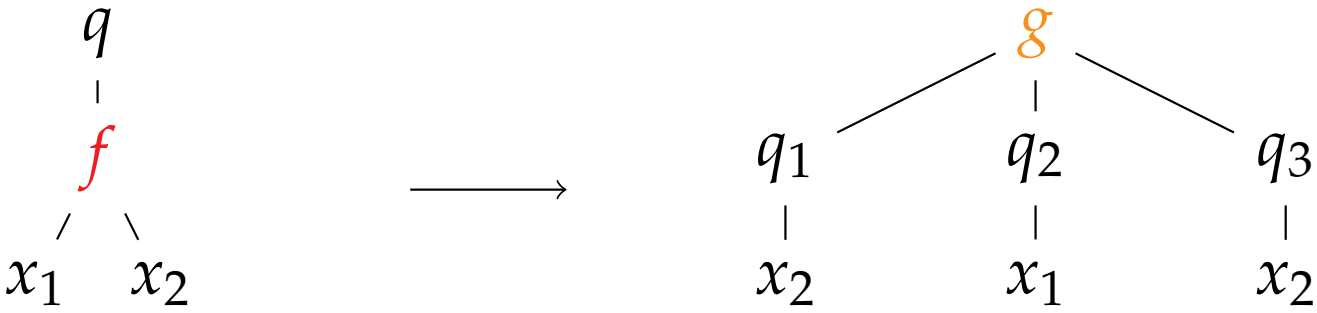
Idea: Replace deterministic specification automaton in the game by a guidable automaton, player Out choosing the transitions.

- A guidable tree automaton $\mathcal{A}$ can "simulate" all tree automata for a sub-language of $\mathcal{A}$.
- For every tree automaton there is an equivalent guidable tree automaton.
- If there is a relabelling DTDTT implementing $\mathcal{A}$, it can serve as a guide to define a winning strategy for Out in the game.

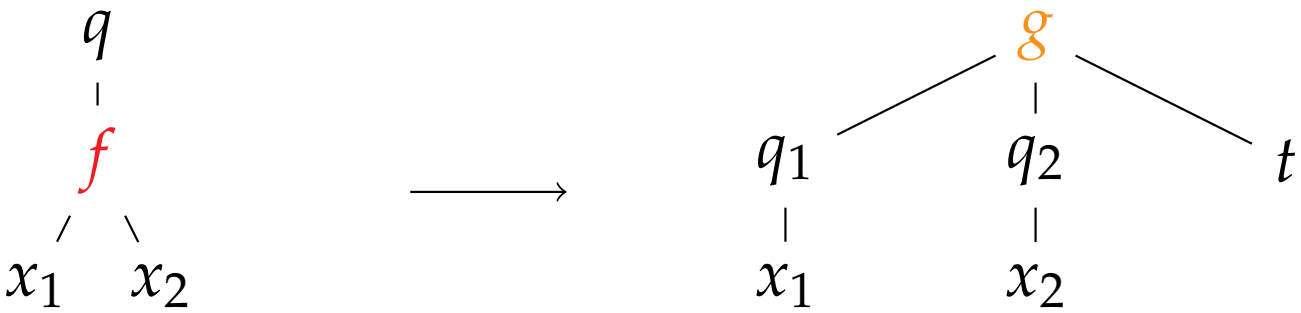
Restricted Cases

specification	implementation
deterministic	general DTDTT
finite unions of domain disjoint deterministic	synchronous DTDTT
nondeterministic	relabelling DTDTT

Synchronous top-down transducer: variables at depth 1



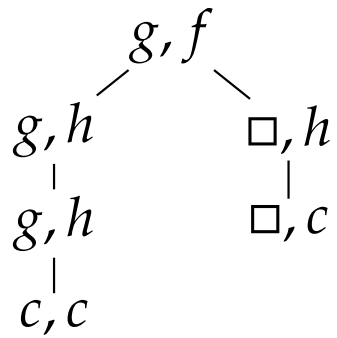
Relabelling top-down transducer: only relabels the nodes



Unions of Deterministic Specifications

$R = R_1 \cup \cdots \cup R_n$ with R_i top-down deterministic and $dom(R_i) \cap dom(R_j) = \emptyset$

Example:
$$\{(g^n(c), f(h^{n-1}(c), h^m(c)) \mid n \geq 1, m \geq 0\}$$
$$\cup \quad \{(g^n(d), f(h^{n-1}(d), h^m(d)) \mid n \geq 1, m \geq 0\}$$



Theorem (L./Winter). For a given specification from the class of relations that are finite union of deterministic top-down tree automatic relations with pairwise disjoint domains, it is decidable whether it can be implemented by a synchronous DTDTT.

Outline

Intuitive Reason for Undecidability

Specification:

input: $i_0 i_1 \cdots i_n \#^m A / B$

output: $w\#^m$

If last letter of input is A, then $w = u_{i_1} \cdots u_{i_n}$

If last letter of input is B, then $w \neq v_{i_1} \cdots v_{i_n}$

- The required implementation in the undecidability proof works in a completely different way as the specification automaton on the inputs ending with B.
- For these inputs, the position of the input heads in
  - an accepting run in the specification automaton, and
  - the run of the implementation

  can be at arbitrary distance when producing the same output symbol.

Preserving Origins

Consider a NTDTT $\mathcal{T}$.

- In a computation of $\mathcal{T}$, each output node u is produced while processing some input node v.
- Then v is called the origin of u in this computation (see Mikolaj's talk).

Preserving Origins

Consider a NTDTT $\mathcal{T}$.

- In a computation of $\mathcal{T}$, each output node u is produced while processing some input node v.
- Then v is called the origin of u in this computation (see Mikolaj's talk).

We say that a DTDTT $\mathcal{D}$ is a k-origin-realization of $\mathcal{T}$ if on each tree t, there is a computation of $\mathcal{T}$ on t

- that produces $\mathcal{D}(t)$, and
- for each output node u, the origin node is at distance at most k from the origin node of u in the computation $\mathcal{D}(t)$.

Preserving Origins

Consider a NTDTT $\mathcal{T}$.

- In a computation of $\mathcal{T}$, each output node u is produced while processing some input node v.
- Then v is called the origin of u in this computation (see Mikolaj's talk).

We say that a DTDTT $\mathcal{D}$ is a k-origin-realization of $\mathcal{T}$ if on each tree t, there is a computation of $\mathcal{T}$ on t

- that produces $\mathcal{D}(t)$, and
- for each output node u, the origin node is at distance at most k from the origin node of u in the computation $\mathcal{D}(t)$.

Work in progress:

Almost Theorem (Filiot/Jecker/L./Winter). For each k, the k-origin-realization problem for nondeterministic TDTT is decidable.

Conclusion

Decidability results:

specification	implementation
deterministic automatic	general DTDTT
finite unions of domain disjoint deterministic automatic	synchronous DTDTT
nondeterministic automatic	relabelling DTDTT
nondeterministic TDTT	k-uniformizing DTDTT

Perspectives:

- The general problem for nondeterministic automatic specifications
- Further decidable cases for NTDTT specifications
- Other transducer models