Validity of Tree Pattern Queries With Respect to Schema Information

Henrik Björklund [1] Wim Martens [2] Thomas Schwentick [3]

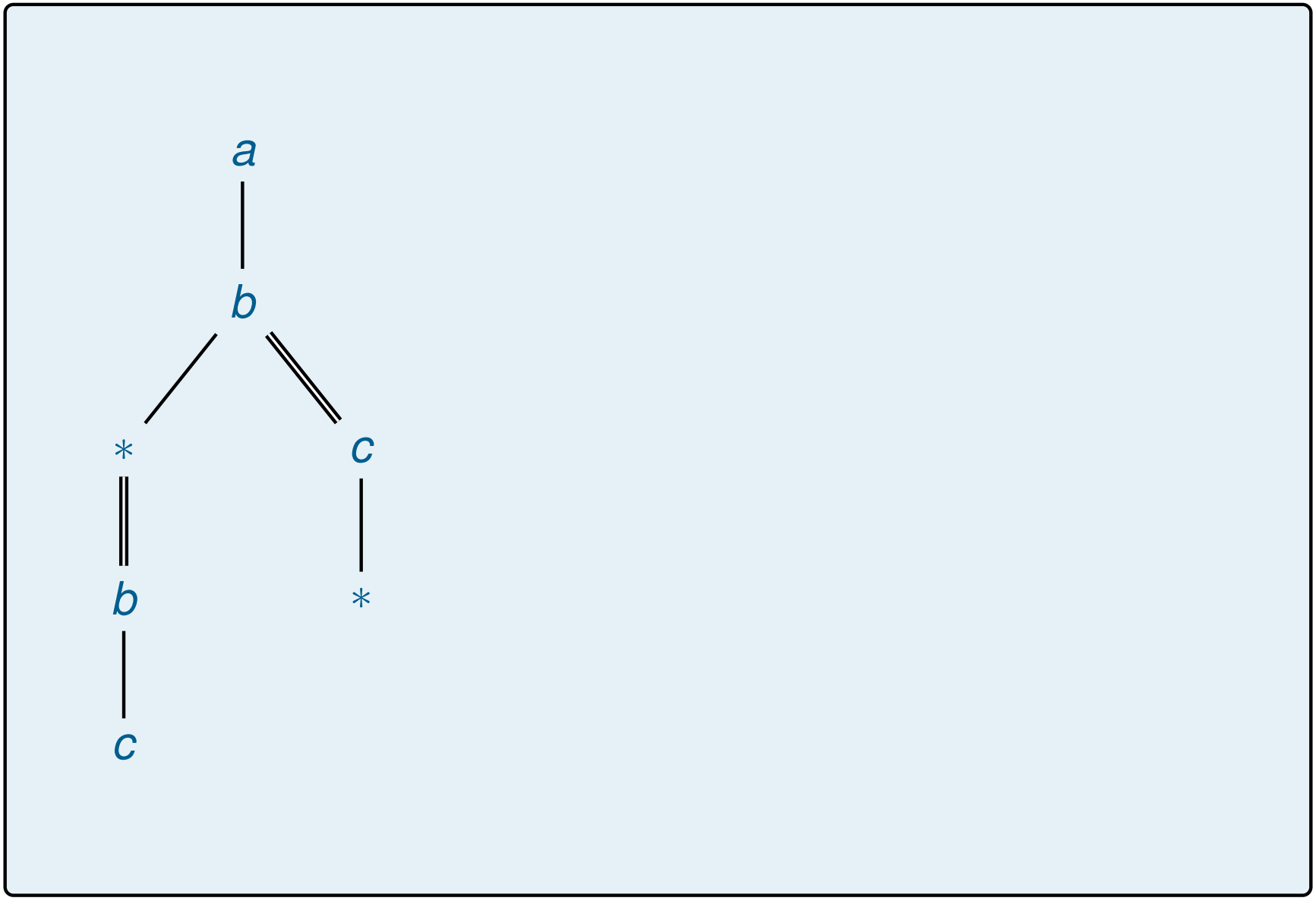
[1]Umeå University

[2]University of Bayreuth

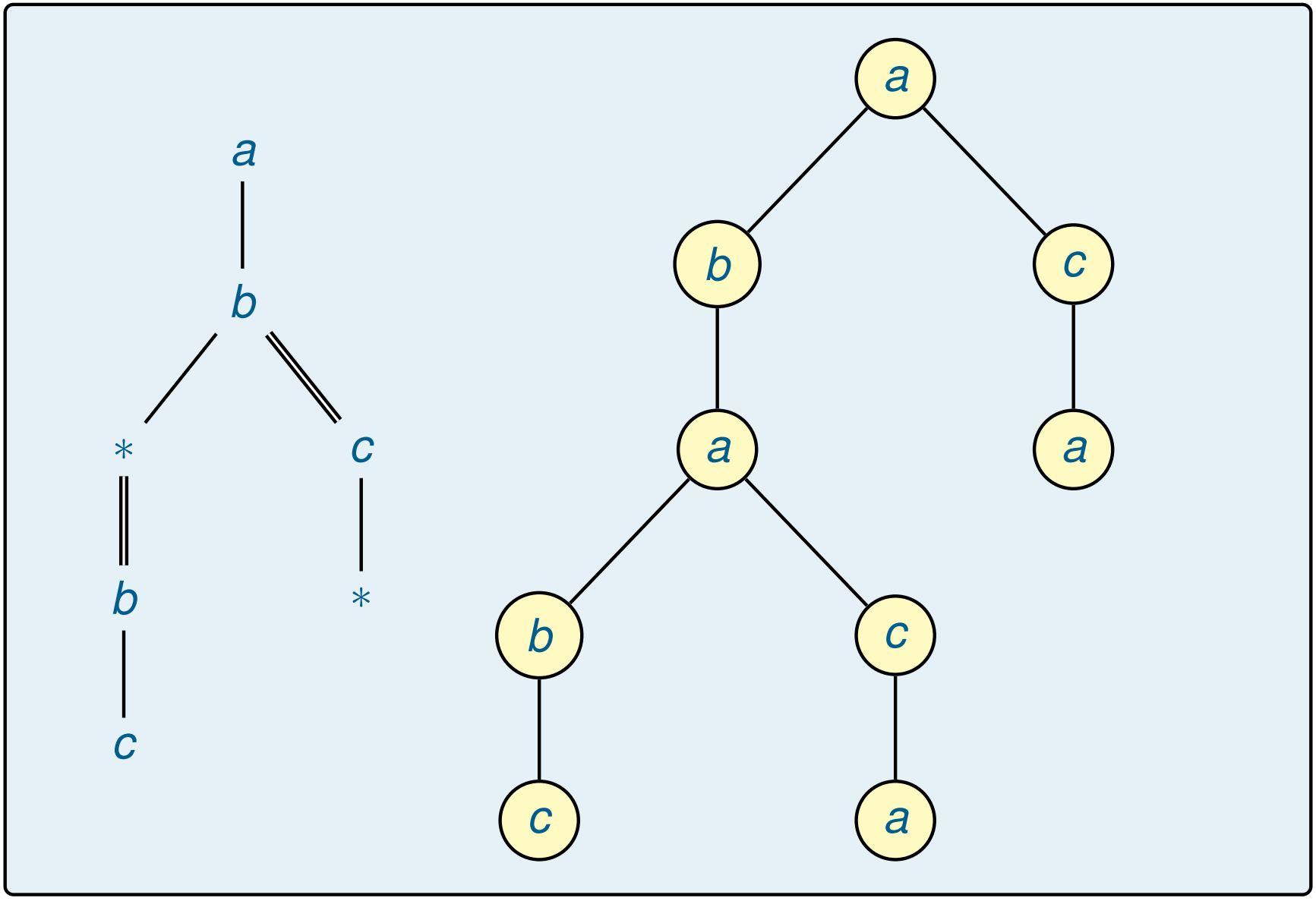
[3]TU Dortmund

MFCS 2013

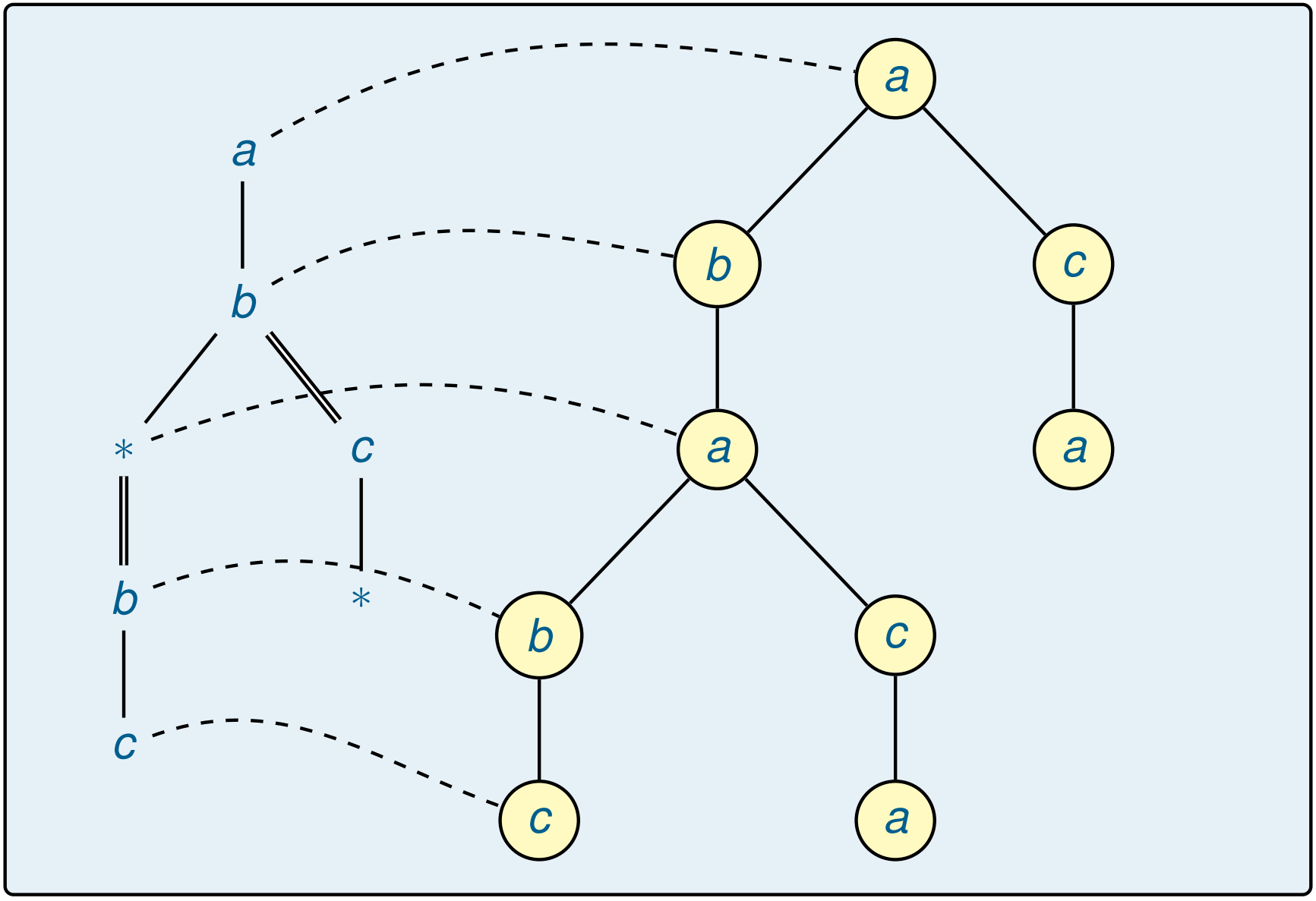
Tree pattern queries



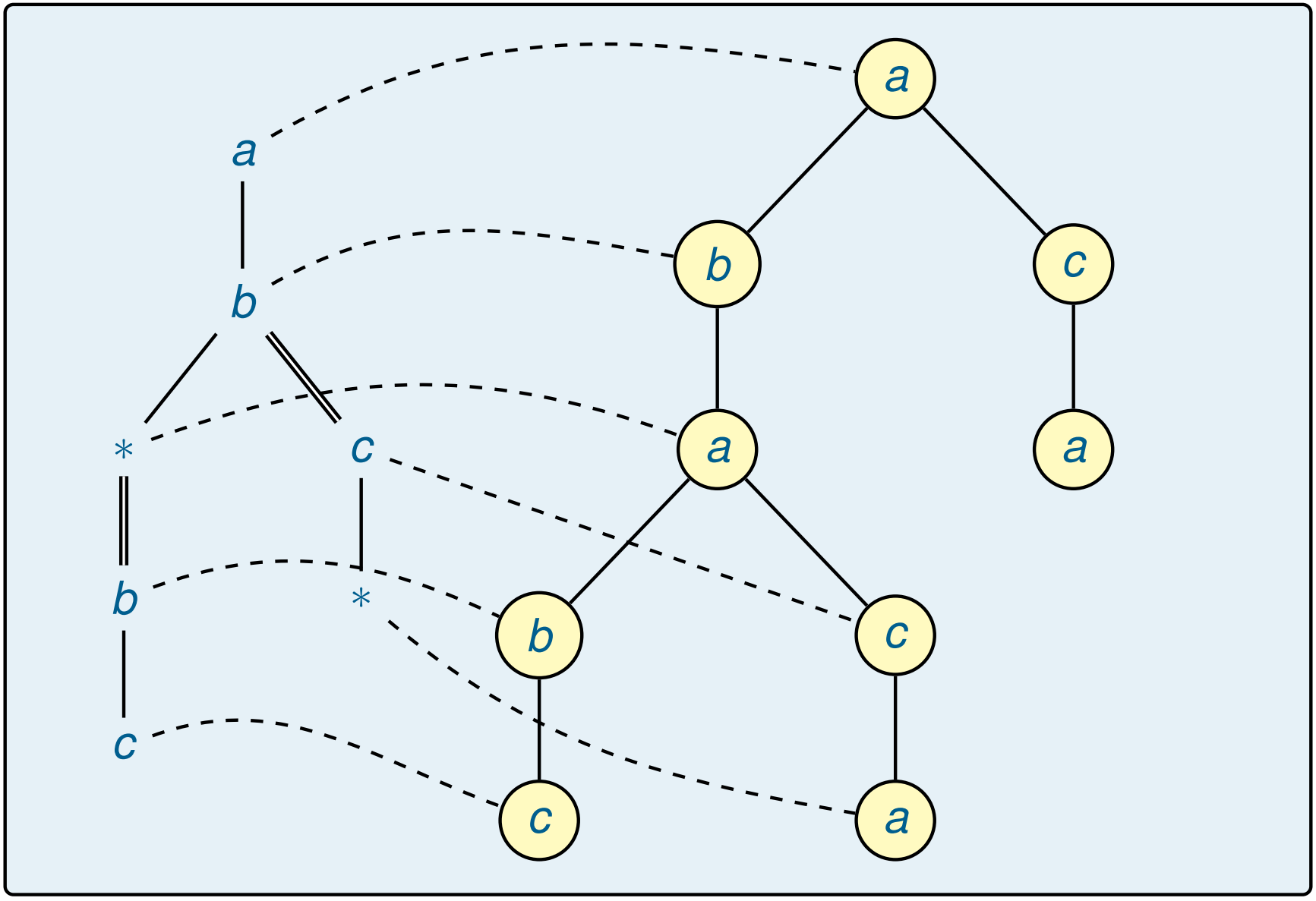
Tree pattern queries



Tree pattern queries



Tree pattern queries



Schemas

Schemas define tree languges (i.e., sets of documents).

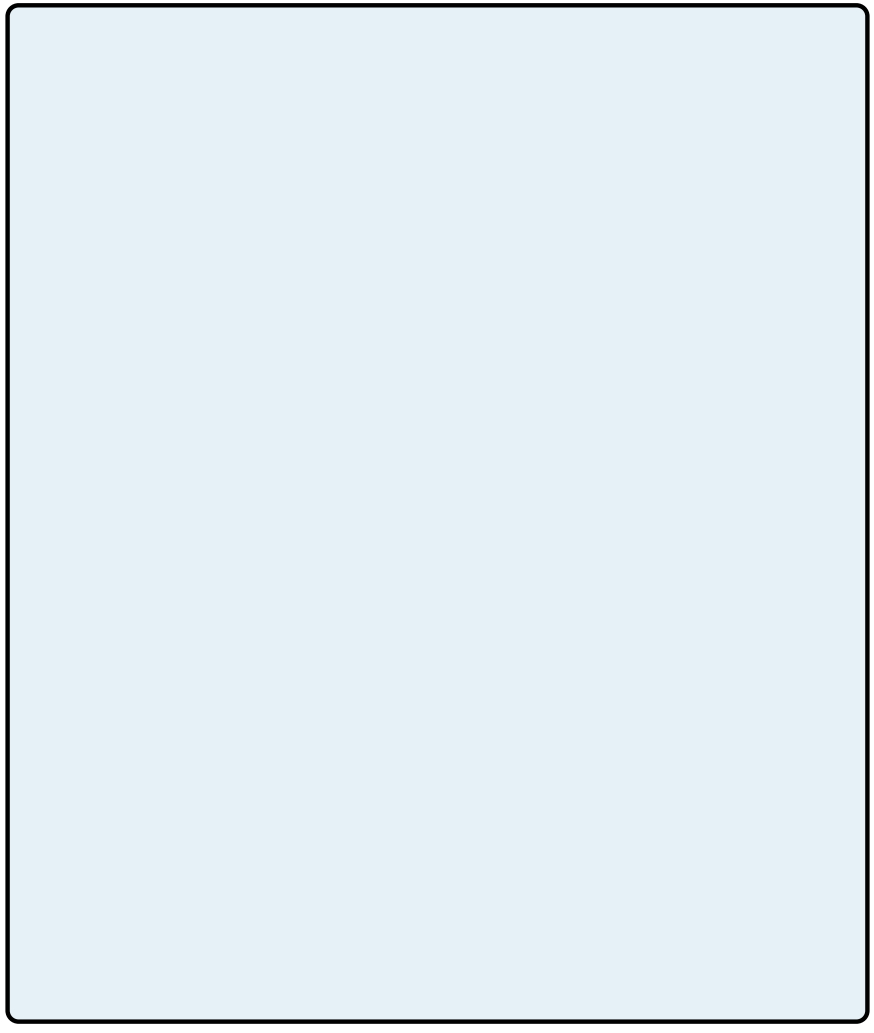
We consider three types of schemas:

- DTDs
- XSDs (corresponding to XML Schema Definitions)
- Tree automata (corresponding to Relax NG)

Validity with respect to a schema

Let Q be a TPQ.

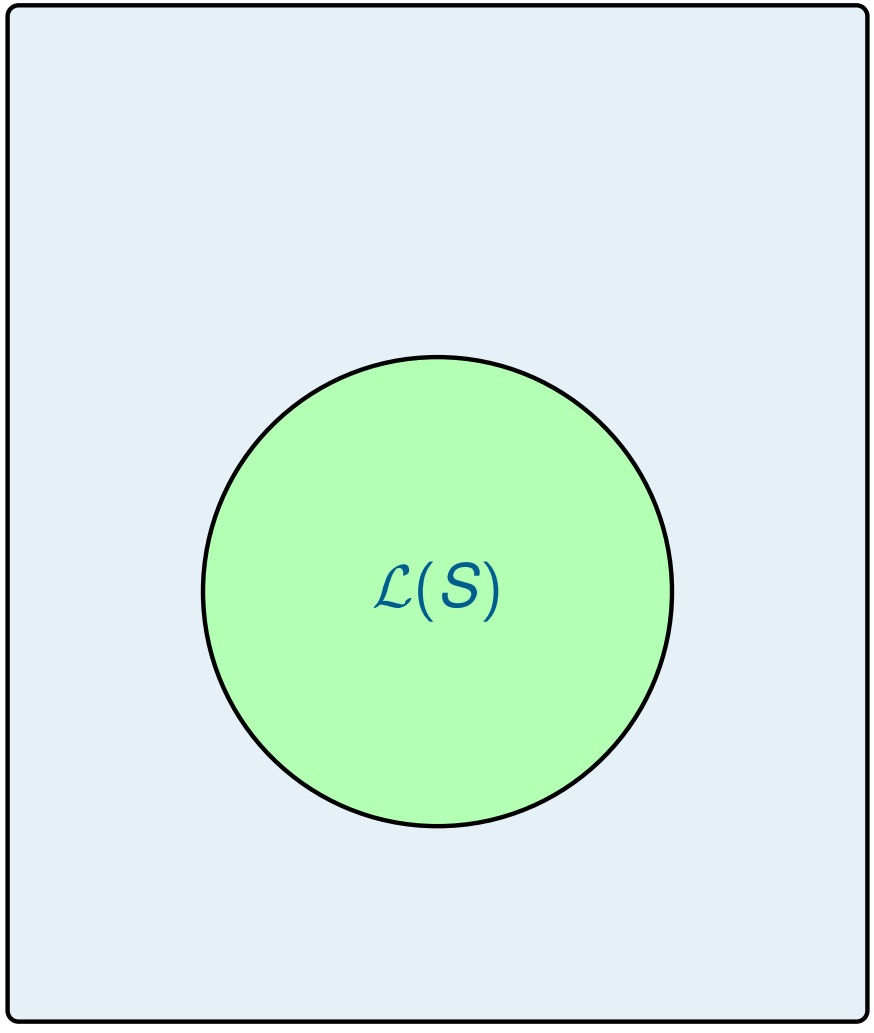
Let S be a schema.



Validity with respect to a schema

Let Q be a TPQ.

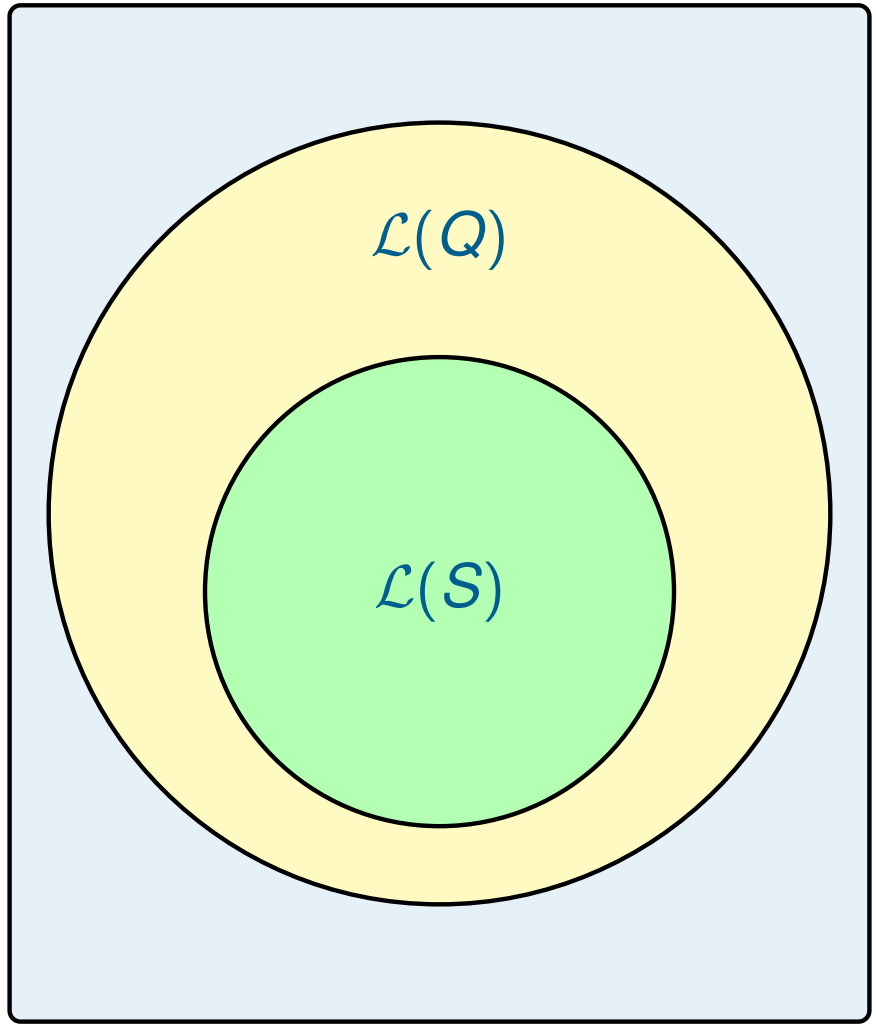
Let S be a schema.



Validity with respect to a schema

Let Q be a TPQ.

Let S be a schema.



Some known results

Theorem (Hashimoto et al., 2011)

Validity of path queries w.r.t. DTDs is in PTIME.

Some known results

Theorem (Hashimoto et al., 2011)

Validity of path queries w.r.t. DTDs is in PTIME.

Theorem (Benedikt et al. 2012)

Validity of unions of conjunctive queries w.r.t. tree automata is in EXPTIME.

Our results (1)

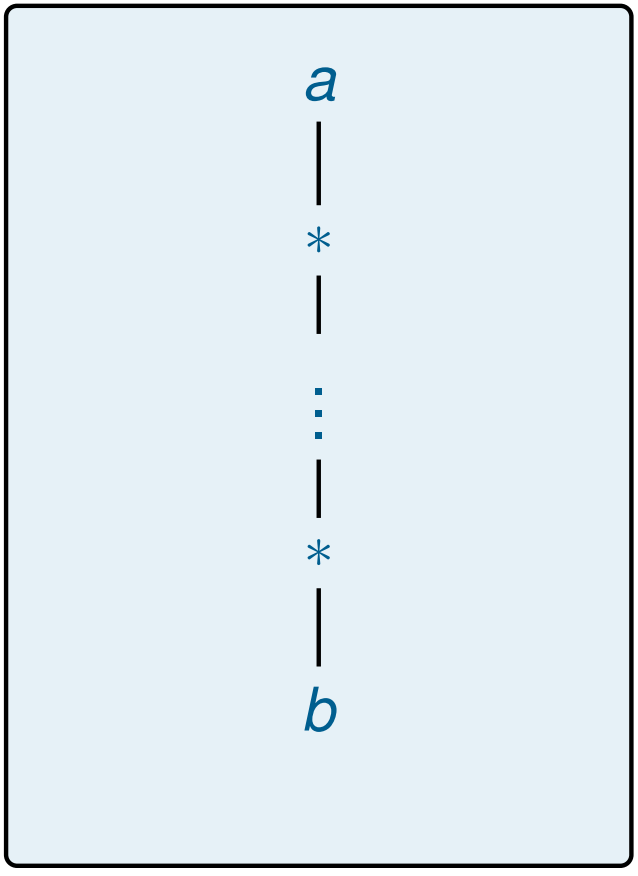
Theorem

Validity of TPQs w.r.t. tree automata is EXPTIME-hard.

Our results (1)

Theorem
Validity of TPQs w.r.t. tree automata is EXPTIME-hard.

This holds even if the schema has only 3 symbols and the query looks like this.



Our results (2)

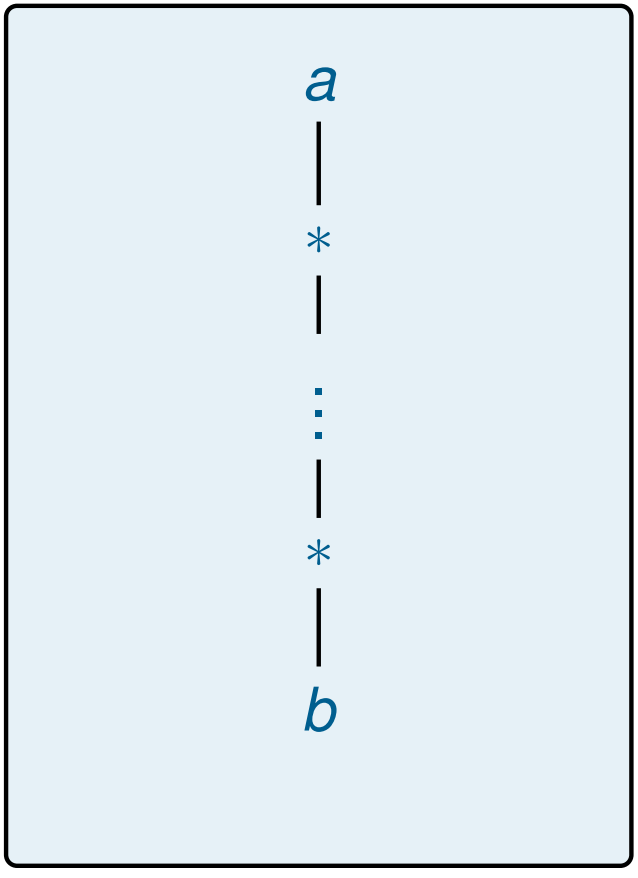
Theorem

Validity of TPQs w.r.t. XSDs is EXPTIME-hard.

Our results (2)

Theorem
Validity of TPQs w.r.t. XSDs is EXPTIME-hard.

This holds even if the schema has only 4 symbols and the query looks like this.



Our results (3)

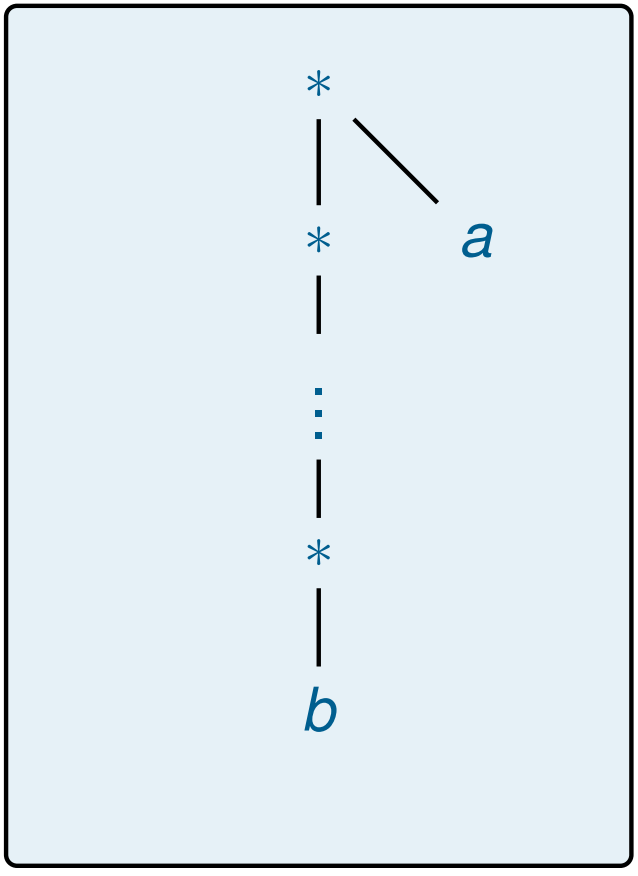
Theorem

Validity of TPQs w.r.t. DTDs is EXPTIME-hard.

Our results (3)

Theorem
Validity of TPQs w.r.t. DTDs is EXPTIME-hard.

This holds even if query looks like this.



Our results (4)

The string warmup version:

Theorem

Inclusion of a DFA language in a regexp language is PSPACE-complete.

Our results (4)

The string warmup version:

Theorem

Inclusion of a DFA language in a regexp language is PSPACE-complete.

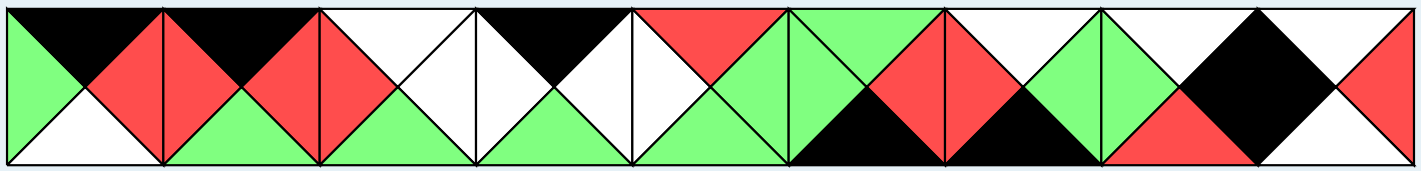
This holds even if the regexp looks like this.

$$\Sigma^* \cdot a \cdot \Sigma^n \cdot b \cdot \Sigma^*$$

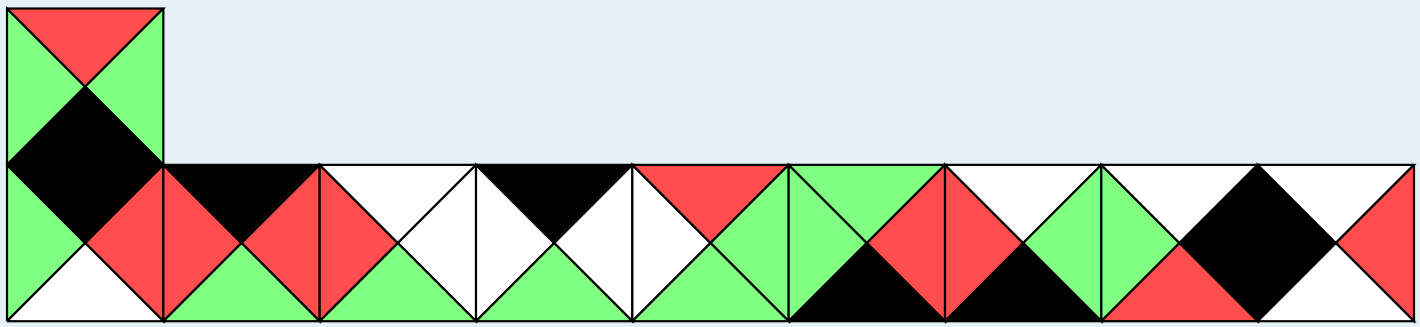
Recommended reading

B. S. Chlebus. Domino-Tiling Games. Journal of Computer and System Sciences, 32(3):374-392, 1986.

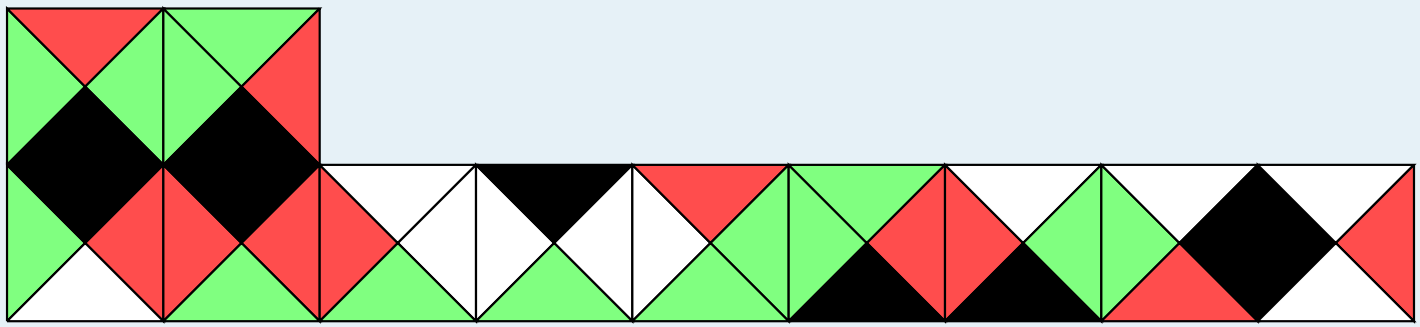
Tiling systems



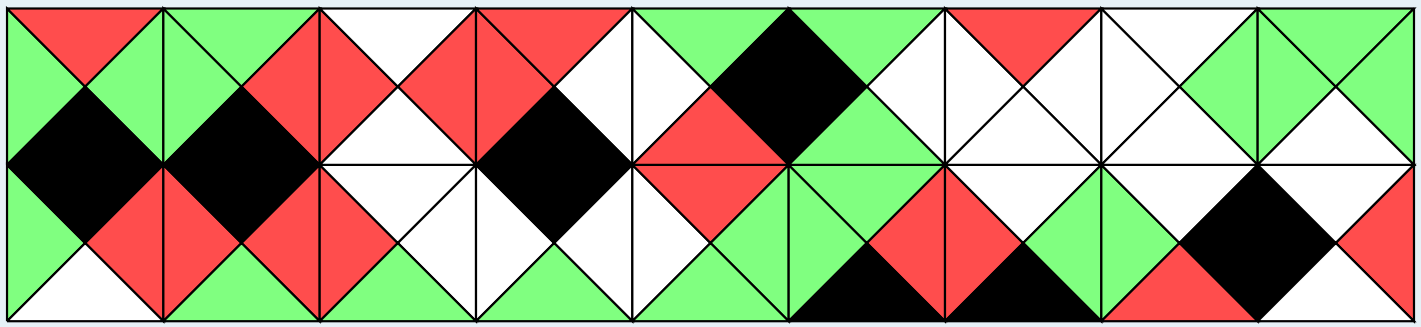
Tiling systems



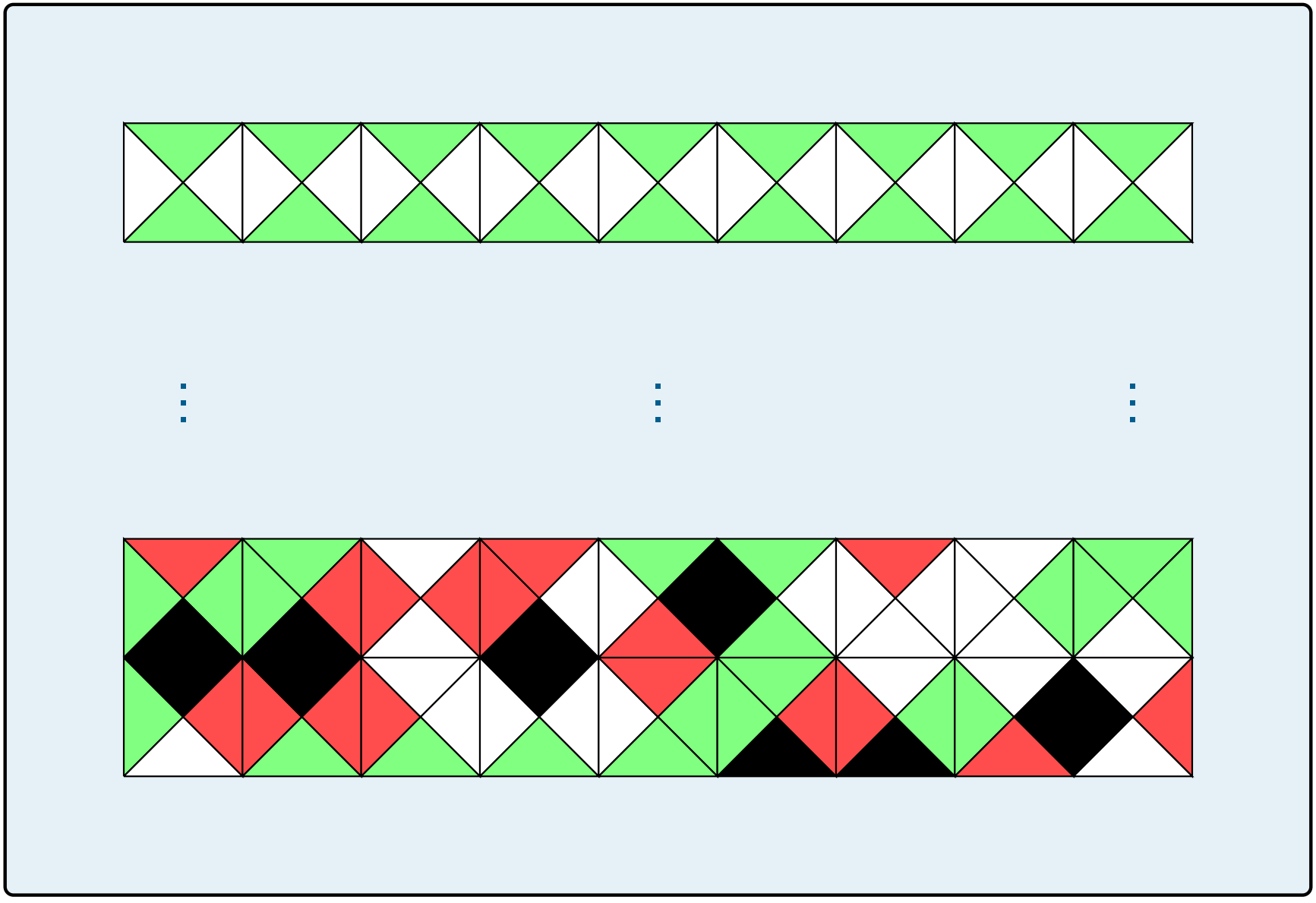
Tiling systems



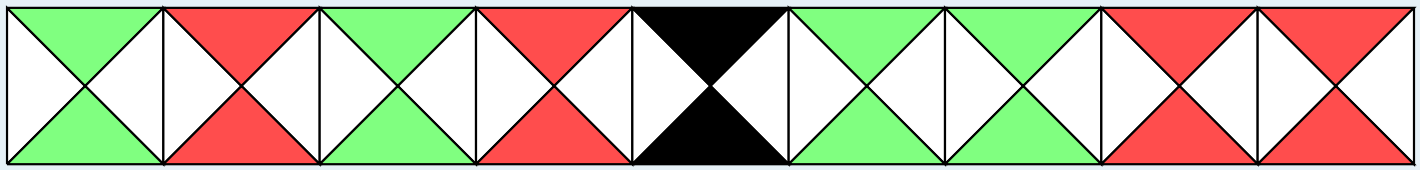
Tiling systems



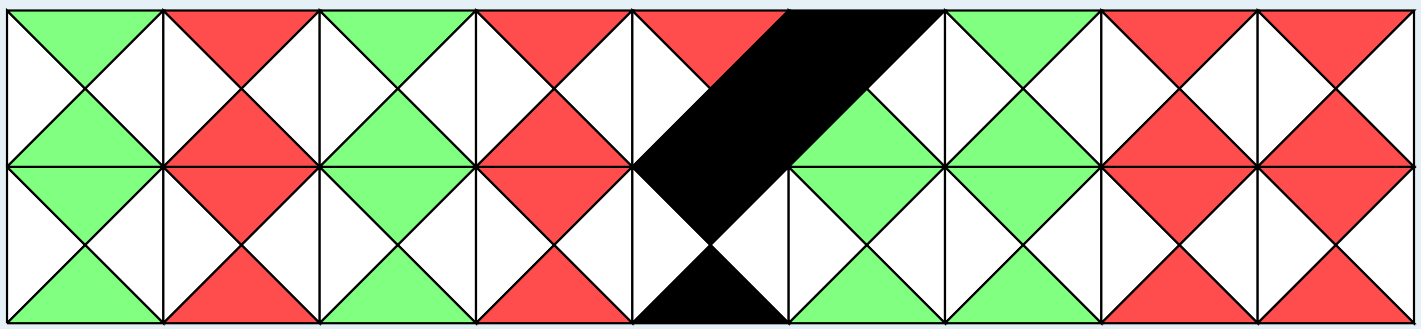
Tiling systems



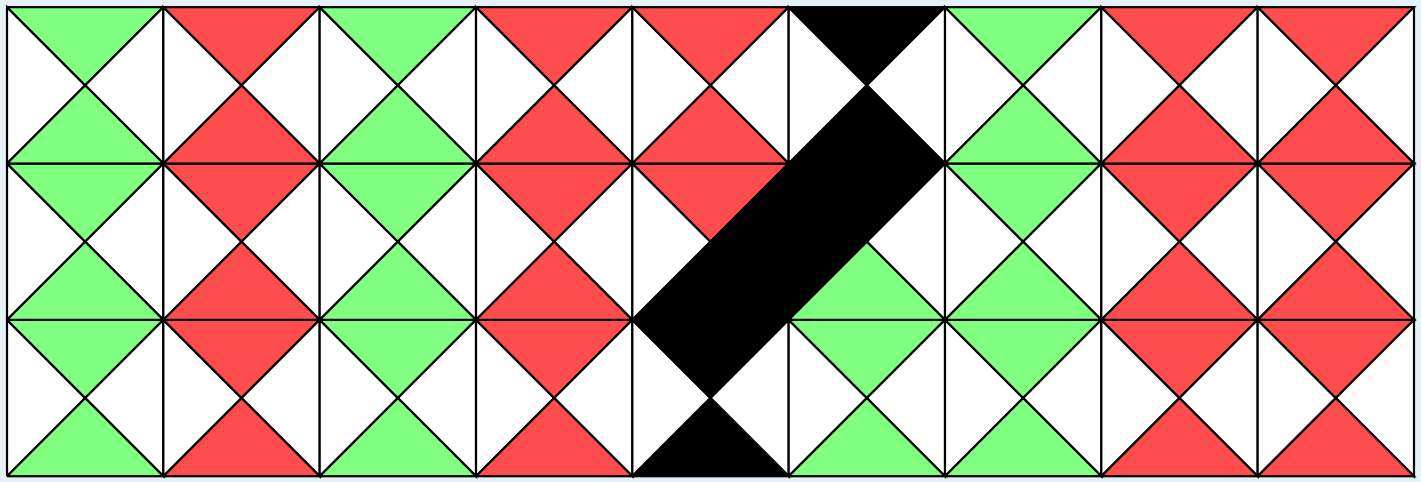
Tilings and TMs



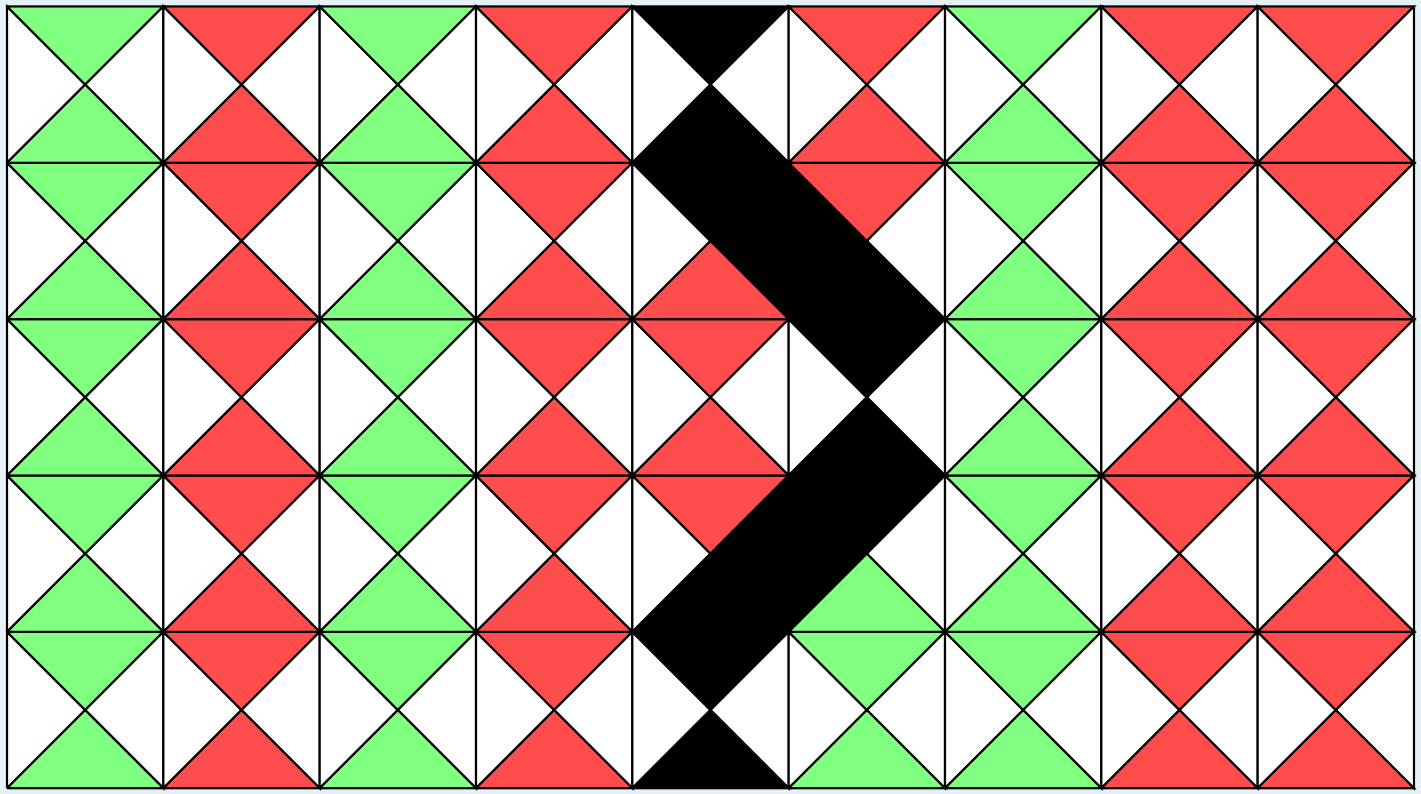
Tilings and TMs



Tilings and TMs



Tilings and TMs



Tilings and complexity

Theorem (Chlebus, 1986)

- *The corridor tiling problem is PSPACE-complete.*
- *The corridor tiling game problem is EXPTIME-complete.*

Tilings and complexity

Theorem (Chlebus, 1986)

- *The corridor tiling problem is PSPACE-complete.*
- *The corridor tiling game problem is EXPTIME-complete.*
- *There is a tiling system S such that Tilings(S) is PSPACE-complete.*
- *There is a tiling system S such that TilingWinner(S) is EXPTIME-complete.*

- Tilings(S): the set of initial rows for which there is a solution S.
- TilingWinner(S): the set of initial rows for which Player 1 has a winning strategy in the tiling game on S.

Restricted tiling systems

A tiling system is restricted if

- it has even length,
- at each point, there are exactly two continuations such that the row can be completed, and
- whenever player 2 is to move, there are exactly two legal moves.

Restricted tiling systems

A tiling system is restricted if

- it has even length,
- at each point, there are exactly two continuations such that the row can be completed, and
- whenever player 2 is to move, there are exactly two legal moves.

Theorem

There is a restricted tiling system S such that TilingWinner(S) is EXPTIME-complete.

Theorem

Inclusion of a DFA language in a regexp language is PSPACE-hard, even for expressions of the form

$$\Sigma^* \cdot a \cdot \Sigma^n \cdot b \cdot \Sigma^*.$$

Strings

Theorem

Inclusion of a DFA language in a regexp language is PSPACE-hard, even for expressions of the form

$$\Sigma^* \cdot a \cdot \Sigma^n \cdot b \cdot \Sigma^*.$$

First step: consider tilings as strings.

A DFA

Given tiling system S and initial row w, construct a DFA A that accepts strings that

A DFA

Given tiling system S and initial row w, construct a DFA A that accepts strings that

1. have prefix w,

A DFA

Given tiling system S and initial row w, construct a DFA A that accepts strings that

1. have prefix w,
2. have length divisible by $|w|$,

A DFA

Given tiling system S and initial row w, construct a DFA A that accepts strings that

1. have prefix w,
2. have length divisible by $|w|$,
3. respect all horizontal constraints, and

A DFA

Given tiling system S and initial row w, construct a DFA A that accepts strings that

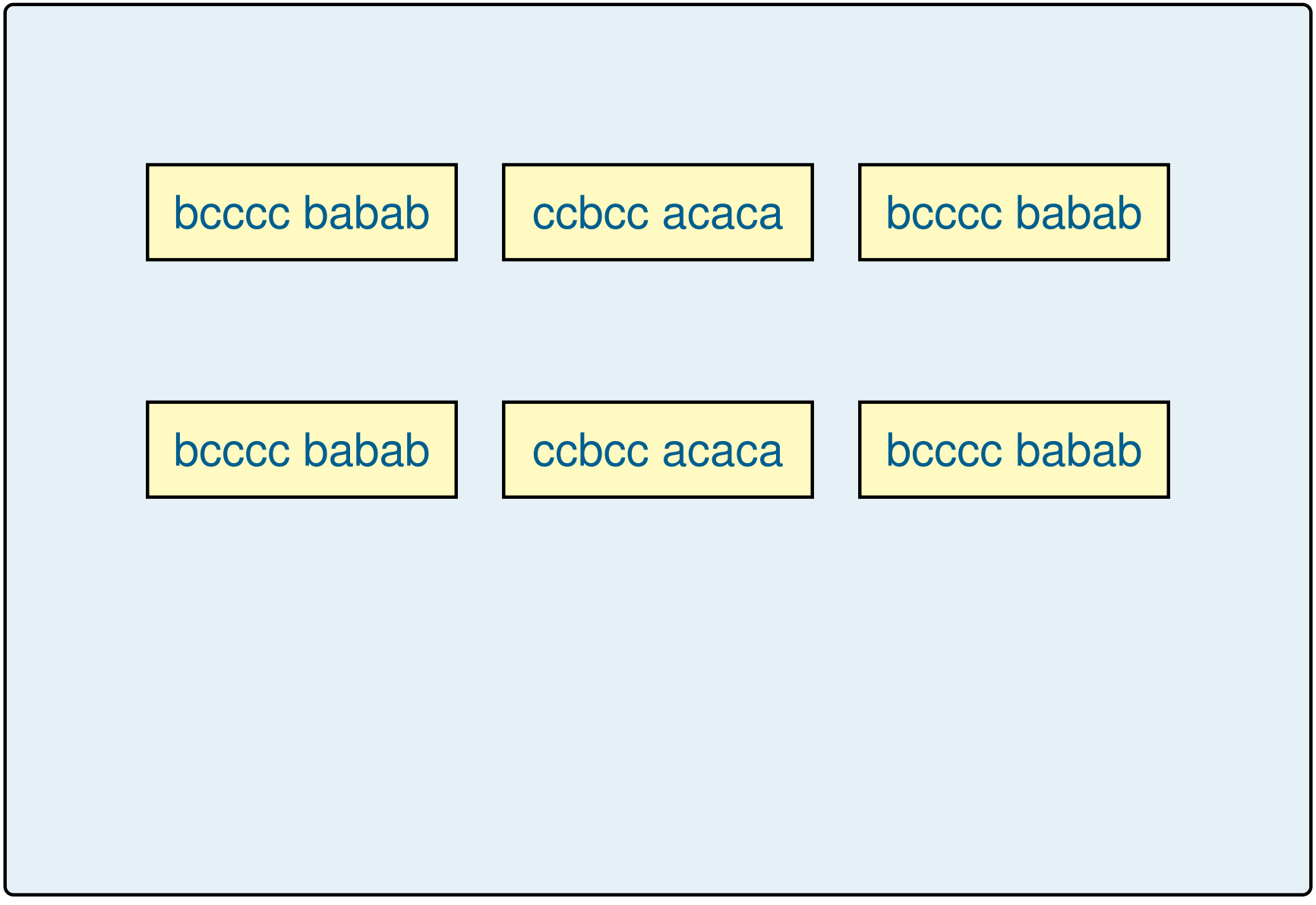
1. have prefix w,
2. have length divisible by $|w|$,
3. respect all horizontal constraints, and
4. have the final row as suffix.

Encoding tiles

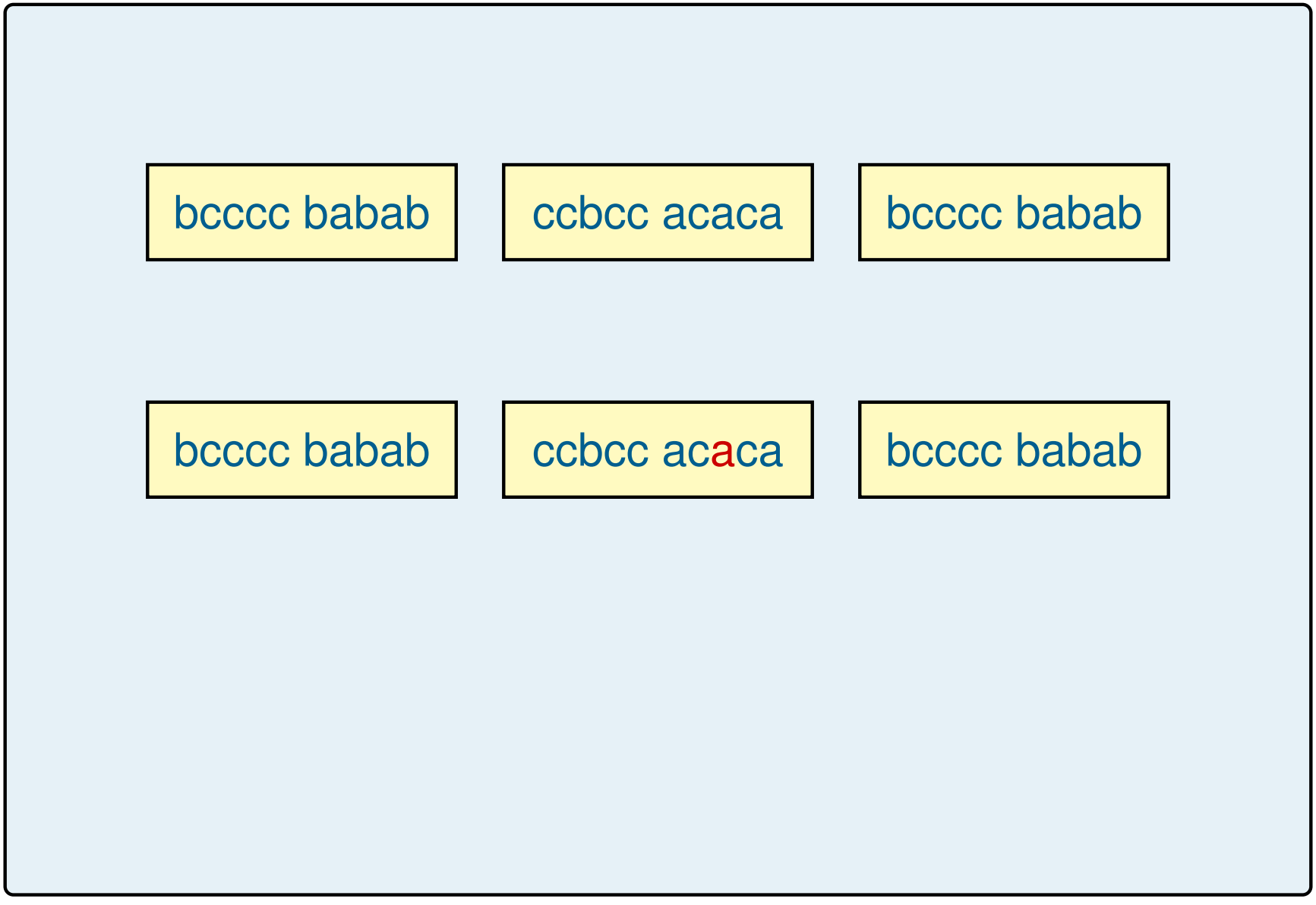
We will encode tiles by strings of length $2k$, where k is the number of tiles in S.
Suppose S has 5 tiles and that tiles 2 and 4 are the only ones allowed on top of tile 3. Then we encode tile 3 by

ccbcc acaca

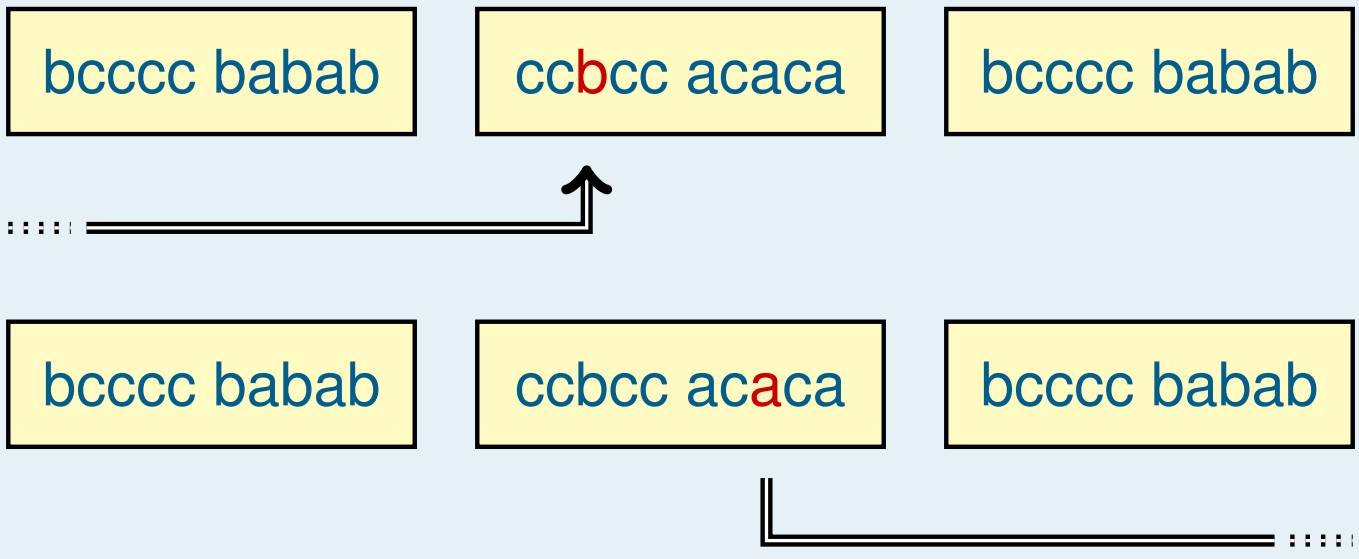
Detecting vertical errors



Detecting vertical errors



Detecting vertical errors



$(2n-1)k = 5 \cdot 5 = 25$ steps

Putting the pieces together

We are given a tiling system S with k tiles and an initial row w of length n.

Putting the pieces together

We are given a tiling system S with k tiles and an initial row w of length n.

We can construct a DFA A and a regular expression $r = \Sigma^* \cdot a \cdot \Sigma^{(2n-1)k} \cdot b \cdot \Sigma^*$ such that if there is no solution to the tiling problem, then r will match every string that is accepted by A.

Putting the pieces together

We are given a tiling system S with k tiles and an initial row w of length n.

We can construct a DFA A and a regular expression $r = \Sigma^* \cdot a \cdot \Sigma^{(2n-1)k} \cdot b \cdot \Sigma^*$ such that if there is no solution to the tiling problem, then r will match every string that is accepted by A.

Thus we have reduced the complement of the tiling problem to inclusion of a DFA language in a regexp language.

Putting the pieces together

We are given a tiling system S with k tiles and an initial row w of length n.

We can construct a DFA A and a regular expression $r = \Sigma^* \cdot a \cdot \Sigma^{(2n-1)k} \cdot b \cdot \Sigma^*$ such that if there is no solution to the tiling problem, then r will match every string that is accepted by A.

Thus we have reduced the complement of the tiling problem to inclusion of a DFA language in a regexp language.

Since PSPACE is closed under complement, this gives us what we need.

Back to tree pattern queries

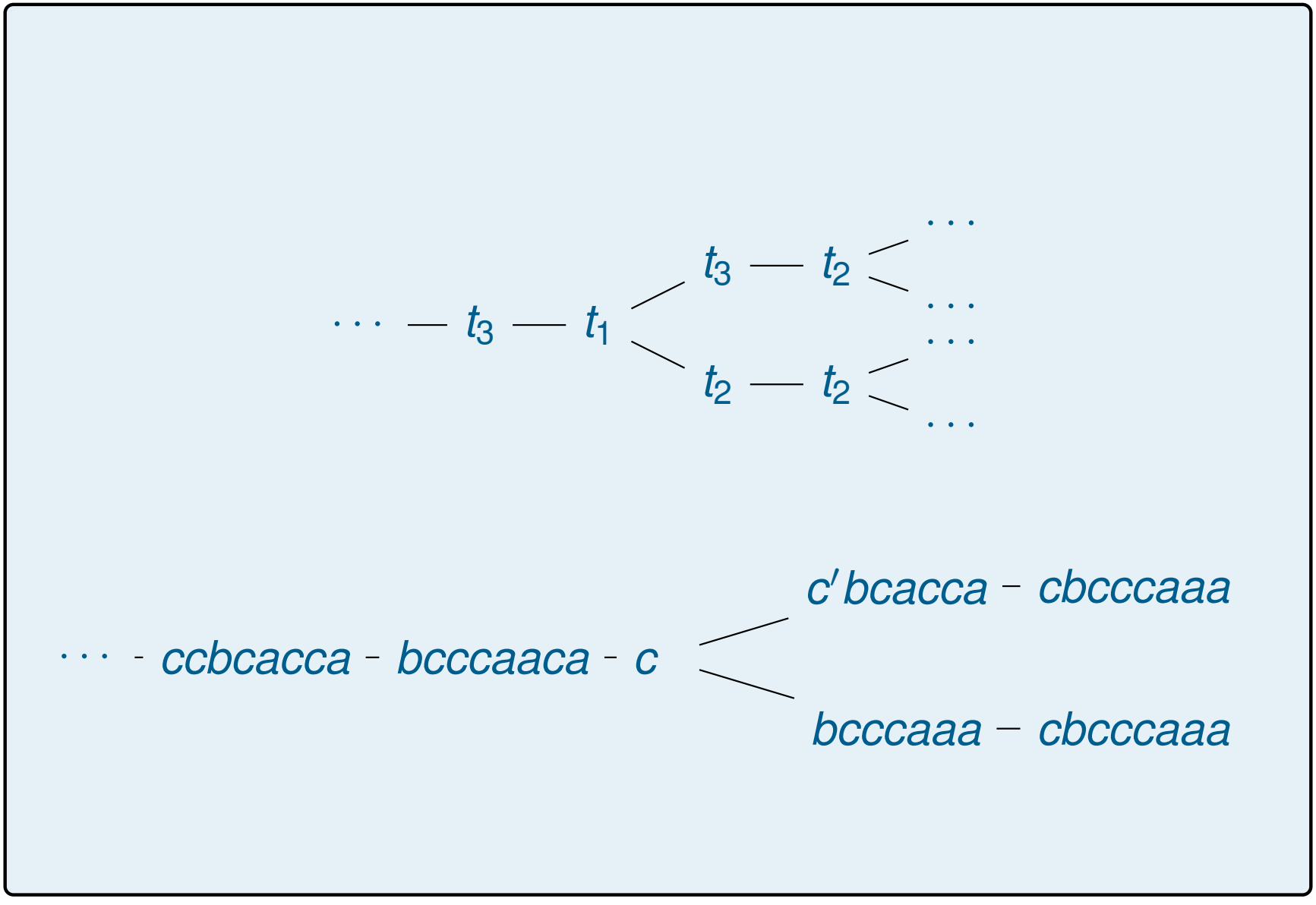
The EXPTIME-hardness results for tree pattern queries are obtained in a very similiar way.

Back to tree pattern queries

The EXPTIME-hardness results for tree pattern queries are obtained in a very similiar way.

We reduce from tiling games and use strategy trees where each branch encodes a possible tiling.

Encoding strategy trees



Result repetition (1)

Theorem

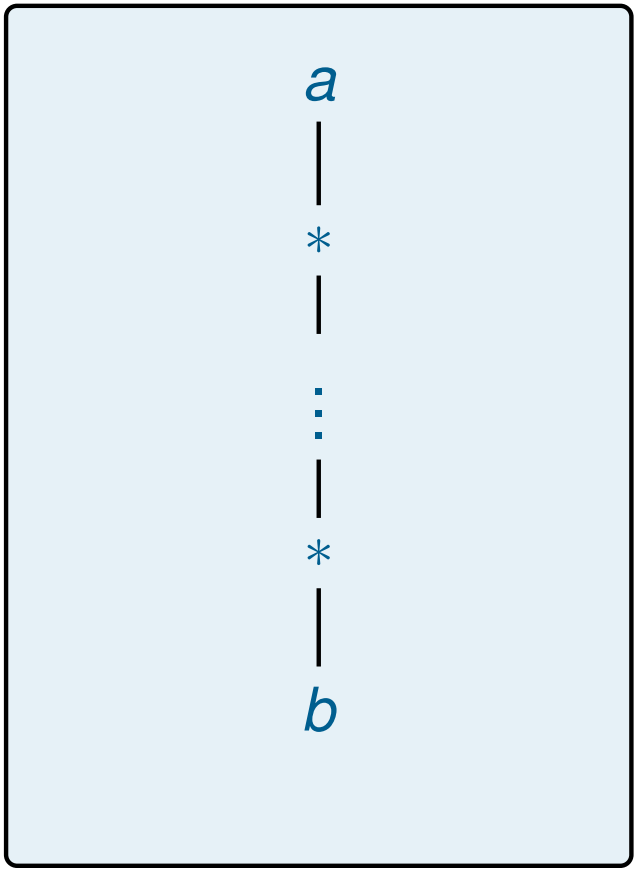
Validity of TPQs w.r.t. tree automata is EXPTIME-hard.

Result repetition (1)

Theorem

Validity of TPQs w.r.t. tree automata is EXPTIME-hard.

This holds even if the schema has only 3 symbols and the query looks like this.



Result repetition (2)

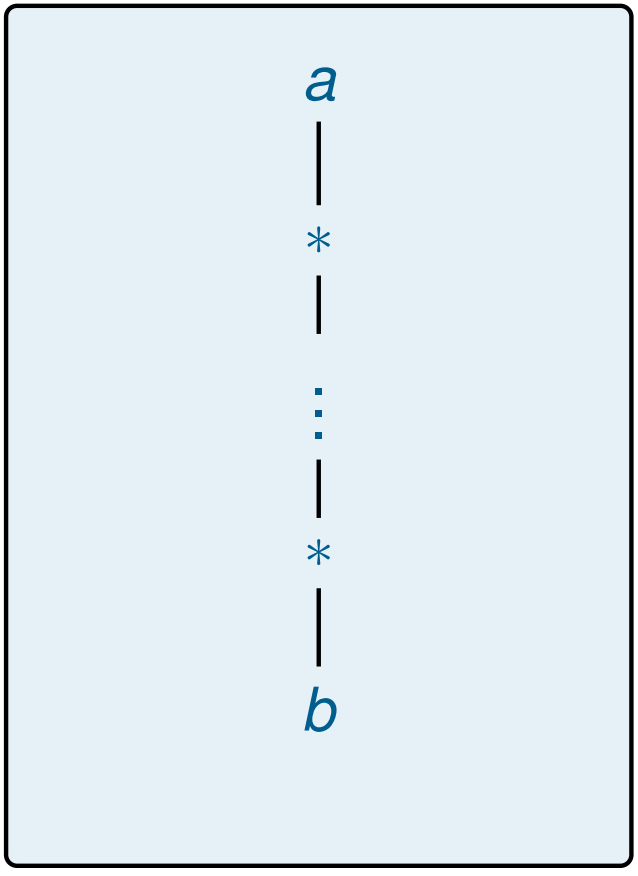
Theorem

Validity of TPQs w.r.t. XSDs is EXPTIME-hard.

Result repetition (2)

Theorem
Validity of TPQs w.r.t. XSDs is EXPTIME-hard.

This holds even if the schema has only 4 symbols and the query looks like this.



Result repetition (3)

Theorem

Validity of TPQs w.r.t. DTDs is EXPTIME-hard.

Result repetition (3)

Theorem

Validity of TPQs w.r.t. DTDs is EXPTIME-hard.

This holds even if query looks like this.