Logics on words and trees with data

Diego Figueira & Ranko Lazić

ESSLLI 2016 - Bolzano

Logics on words and trees with data

Diego Figueira & Ranko Lazić

ESSLLI 2016 - Bolzano

Agenda

- Monday: Introduction, motivation
- Tuesday: Data words, first-order logic
- Wednesday: Data words, temporal logics
- Thursday: Data trees, path-based logics
- Friday: Data trees, other formalisms for data trees

Patterns!

Data Tree Patterns

- Defined is a **tree**
  - Unordered, unranked
  - Nodes can have labels
    - Edges are of type 'child' or 'descendant'
    - Additional equality and inequality constraints between nodes
- A data tree "satisfies" a pattern if there is an morphism

$$f : \text{pattern} \longrightarrow \text{tree}$$

that verifies the constraints of the pattern.

Data Tree Patterns

A **data tree pattern** is a tree with

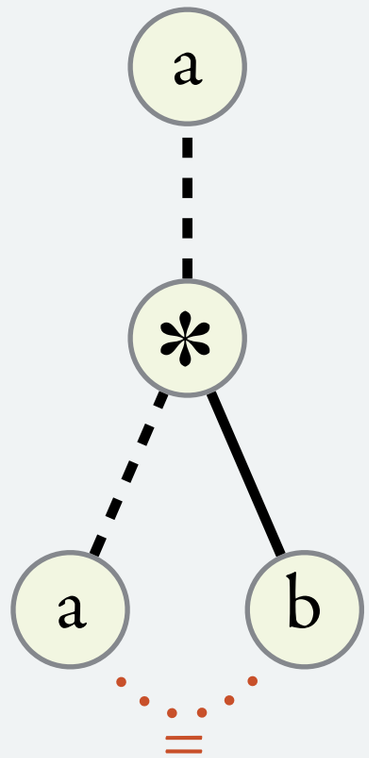
- child/descendant edges,

- labelled nodes, and

- data equality/ inequality constraints between nodes.

Data Tree Patterns

Example:



data pattern

Data Tree Patterns

A **data tree pattern** P is "satisfied" by a data tree T if

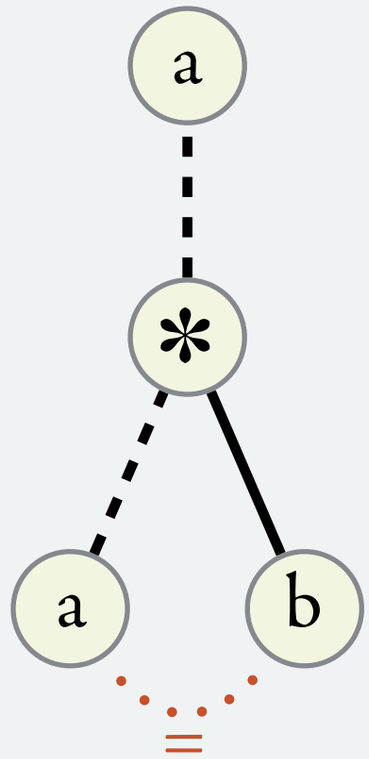
- there is an injective function

  $f : \text{nodes}(P) \longrightarrow \text{nodes}(T)$ so that:

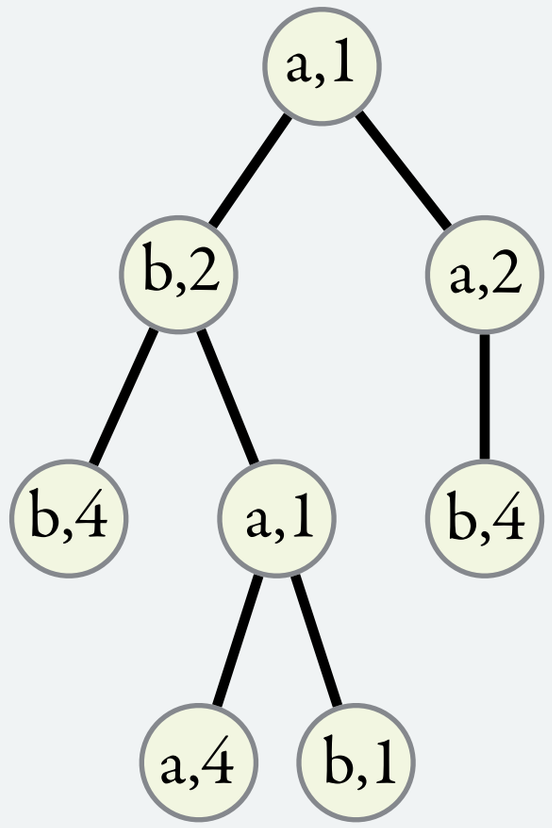
- if $v \ni \text{nodes}(P)$ labeled with **a** $\Rightarrow f(v) \ni \text{nodes}(T)$ labeled with **a**
- (v,v') related by a 'child' edge $\Rightarrow f(v')$ is a child of $f(v)$
- (v,v') related by a 'descendant' edge $\Rightarrow f(v')$ is descendant of $f(v)$
- (v,v') at 'incomparable' positions $\Rightarrow f(v)$, $f(v')$ incomparable
- (v,v') are related by a '=' edge (resp. '≠'), then $f(v)$, $f(v')$ have the same (resp. different) data value.

Data Tree Patterns

Example 1:



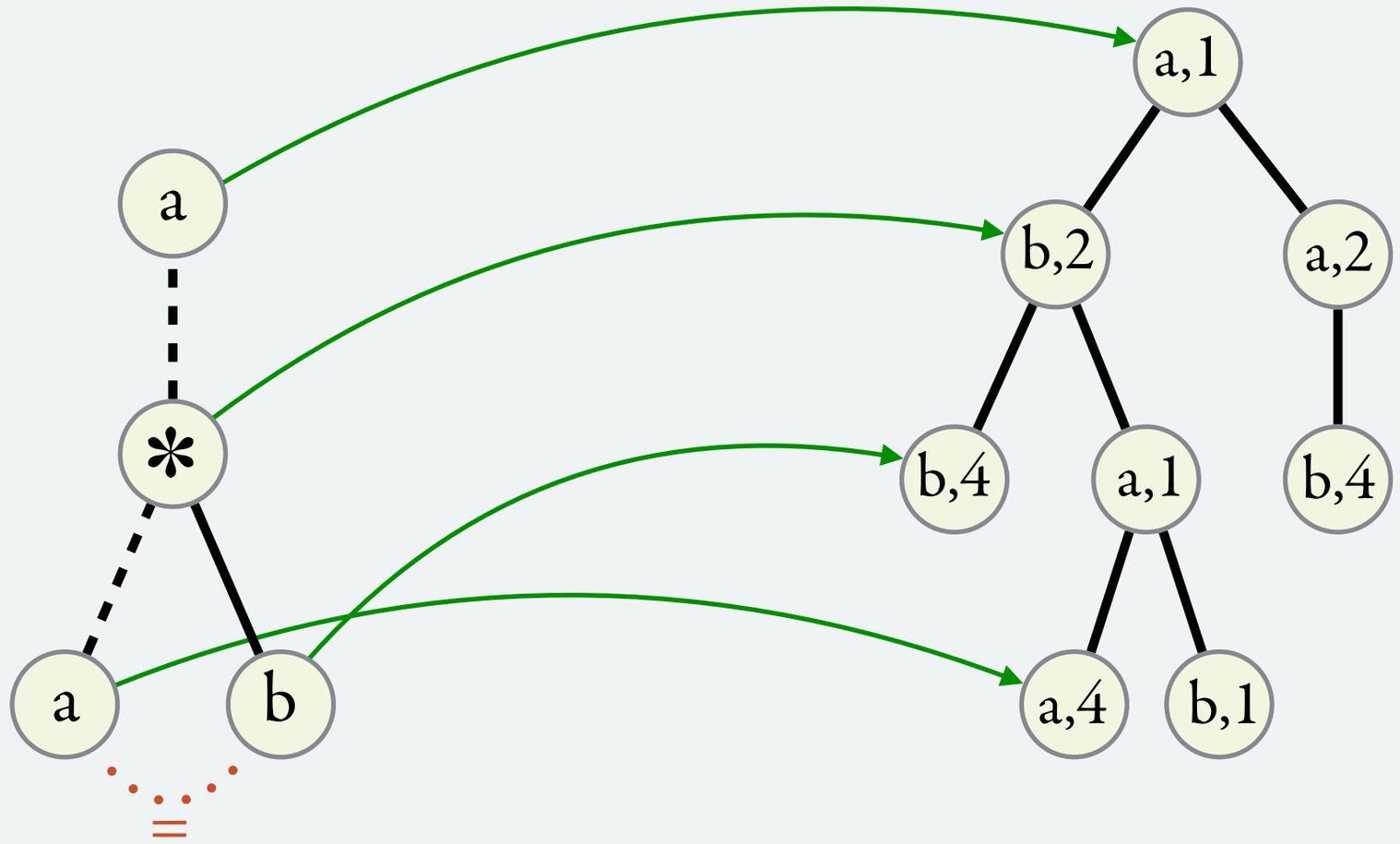
data pattern



data tree

Data Tree Patterns

Example 1:

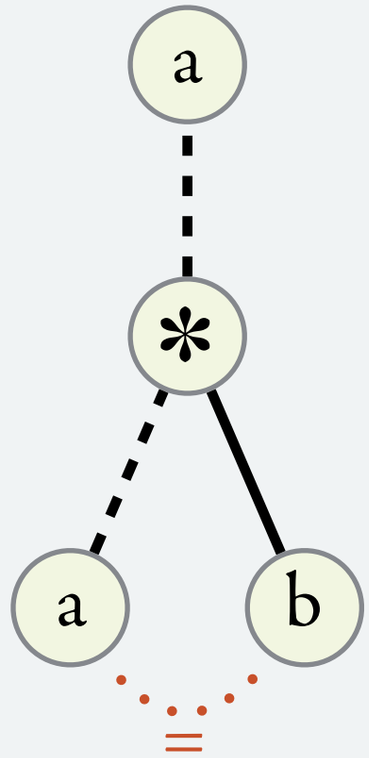


data pattern

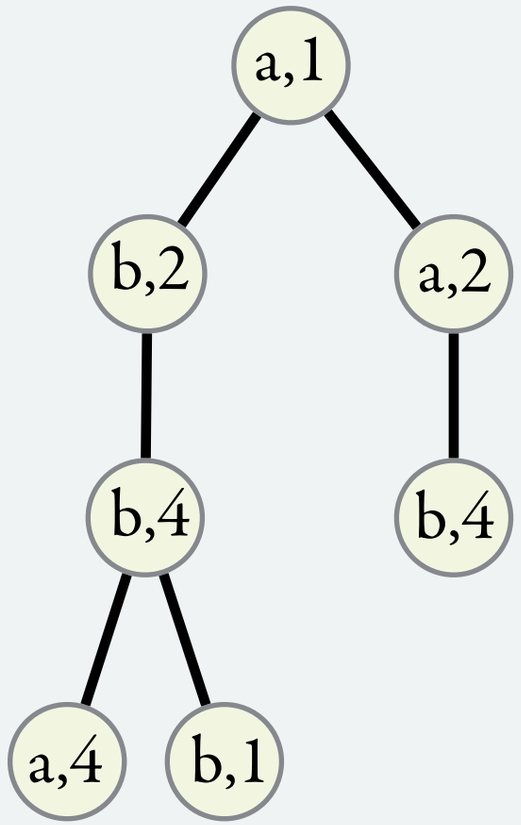
data tree

Data Tree Patterns

Example 2:



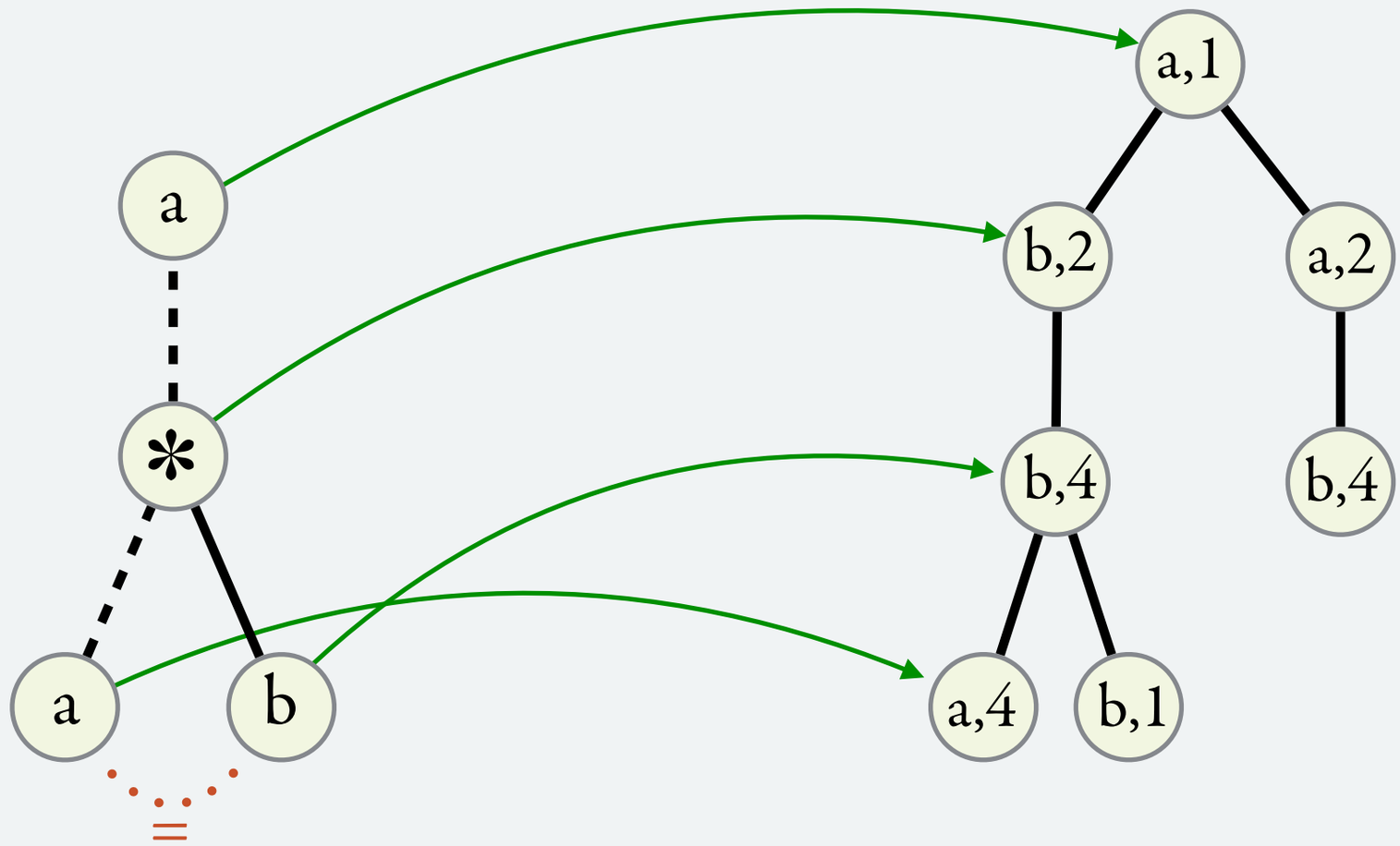
data pattern



data tree

Data Tree Patterns

Example 2:

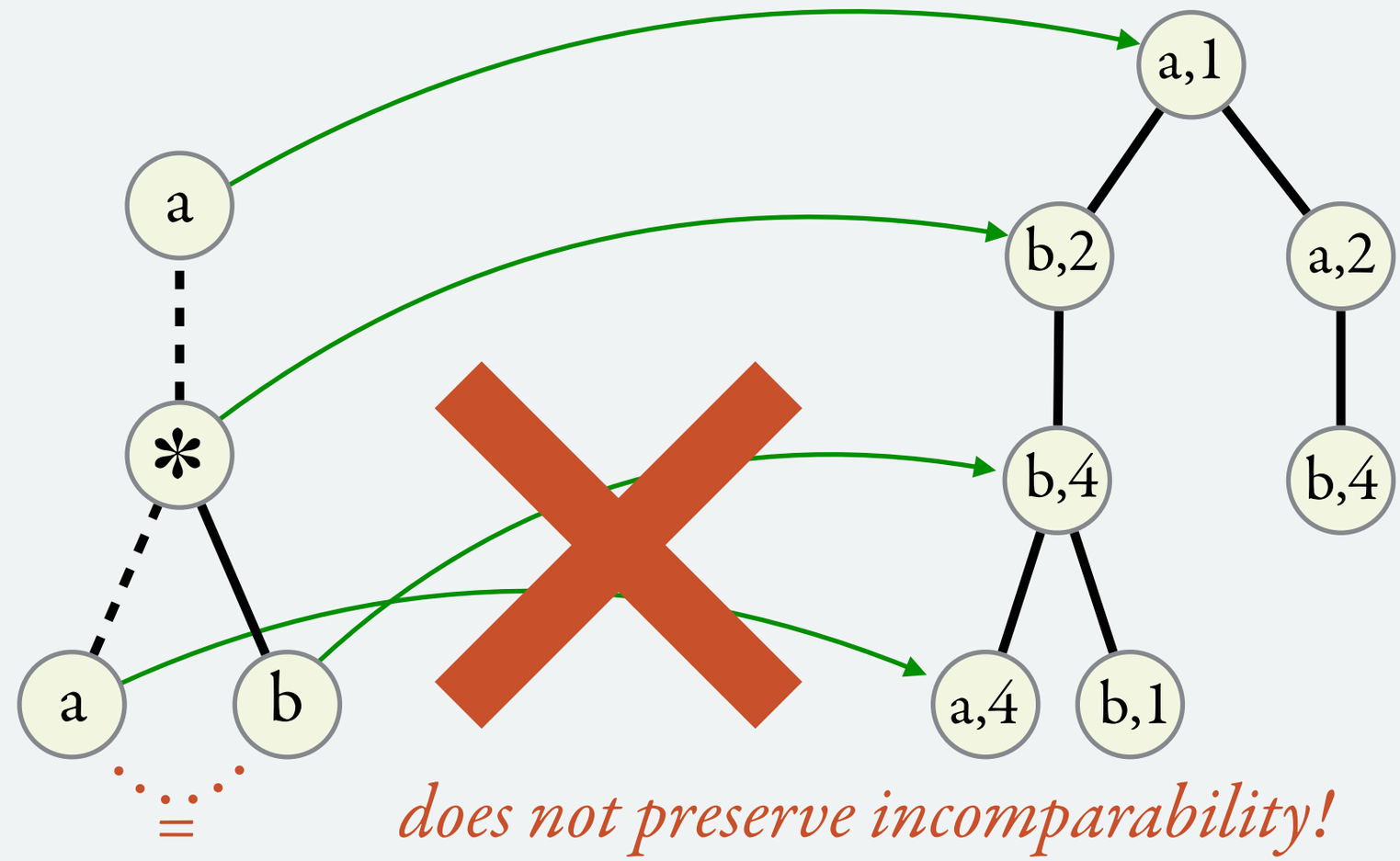


data pattern

data tree

Data Tree Patterns

Example 2:

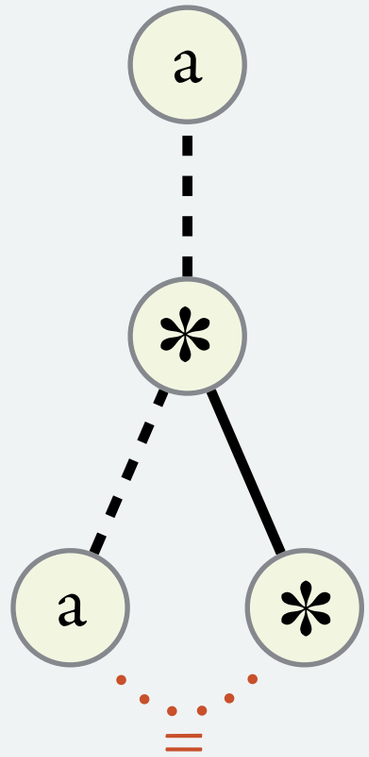


data pattern

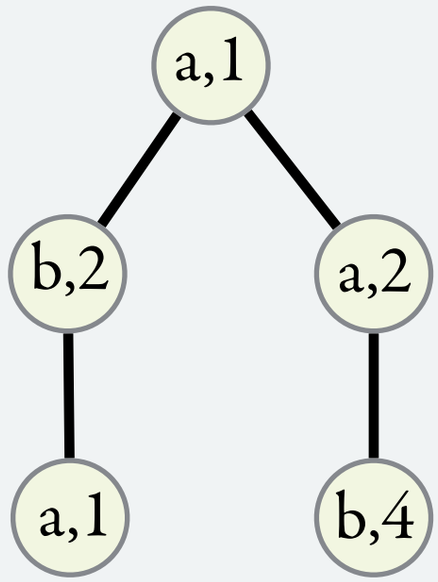
data tree

Data Tree Patterns

Example 3:



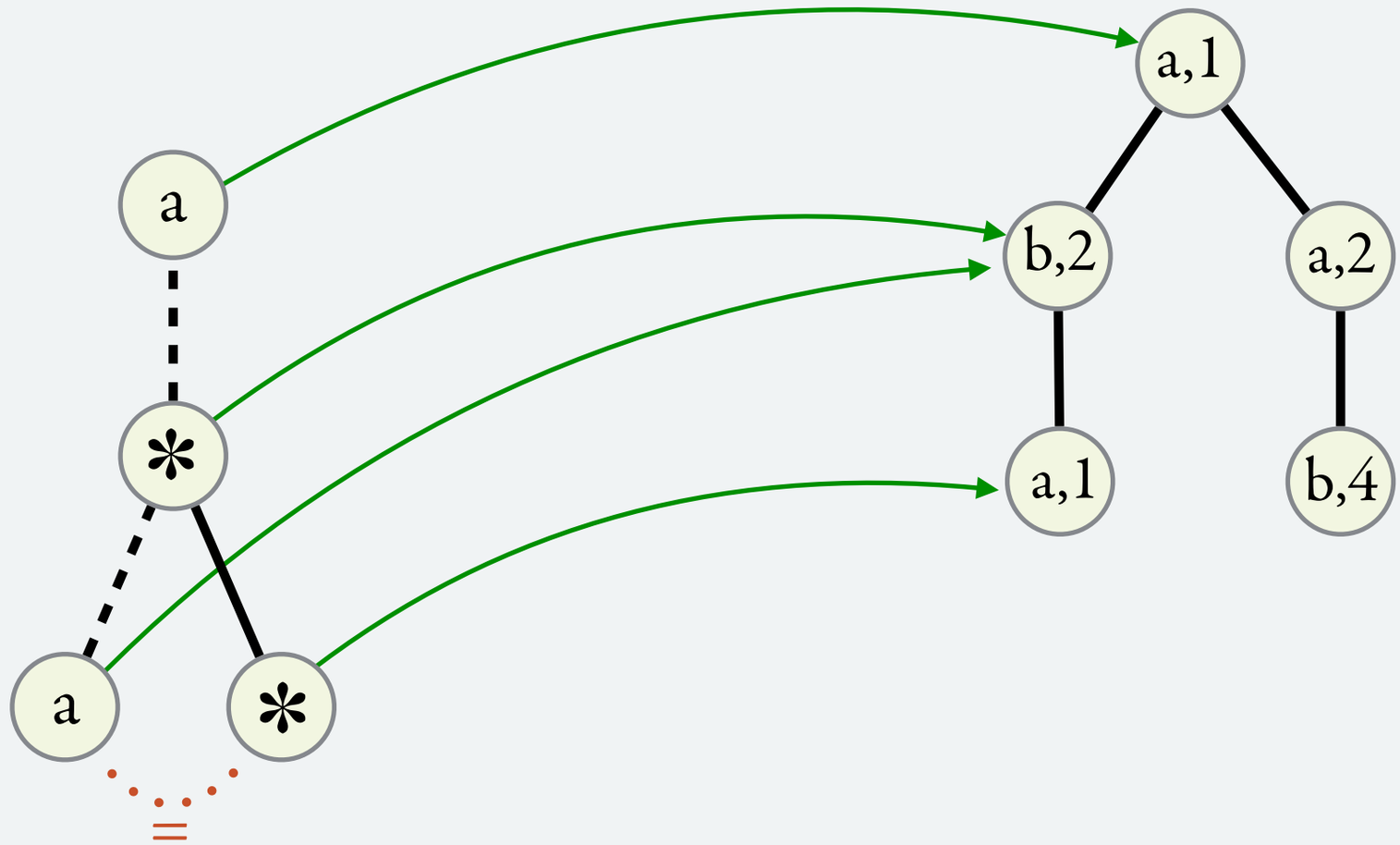
data pattern



data tree

Data Tree Patterns

Example 3:

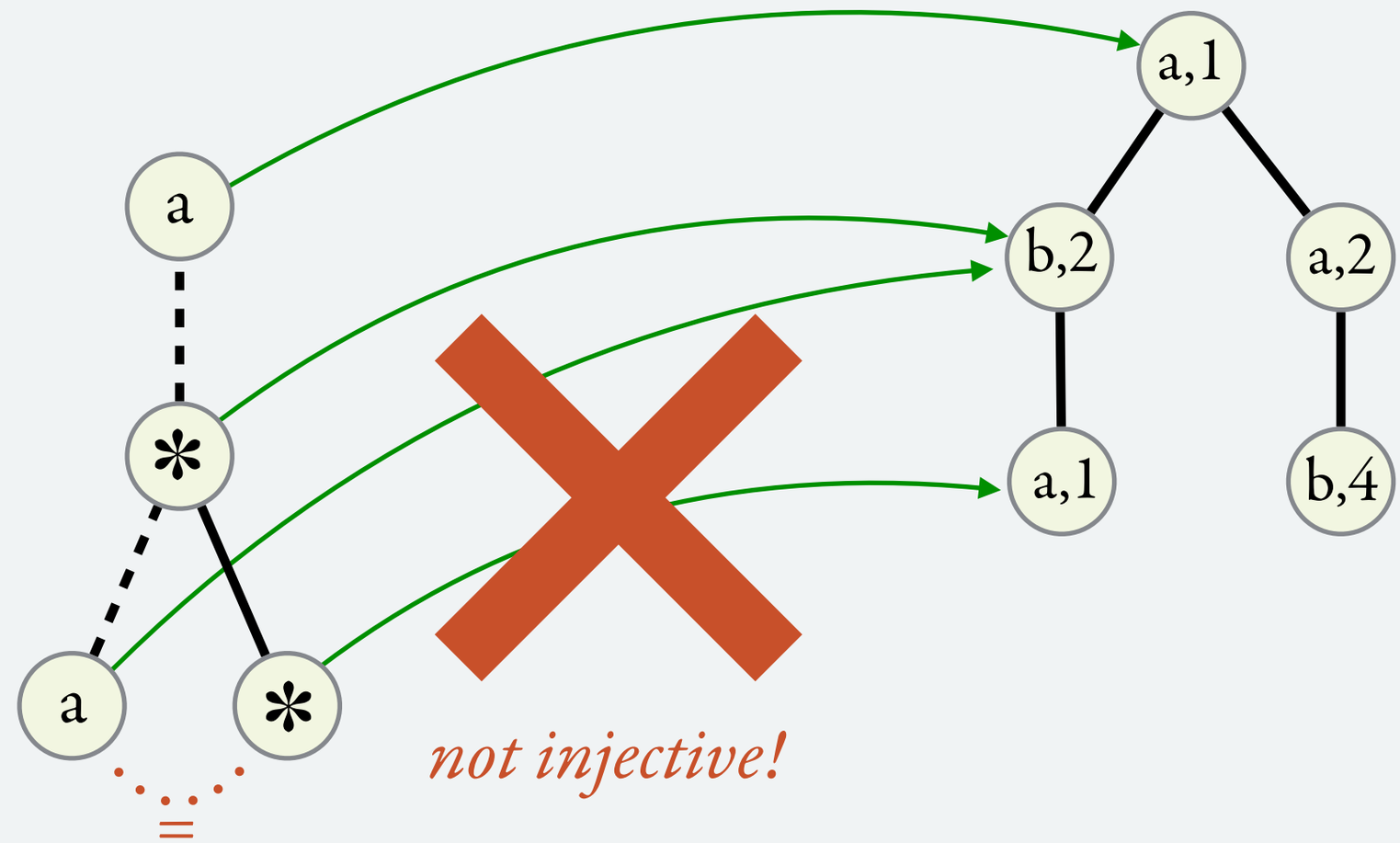


data pattern

data tree

Data Tree Patterns

Example 3:



data pattern

data tree

Satisfiability of Data Patterns

Satisfiability problem: Given a regular tree property and a boolean combination of patterns, is there a tree satisfying both?

Satisfiability of Data Patterns

The satisfiability problem is undecidable [David MFCS'08]

	satisfiability
BC(child, desc)	undecidable
BC(child)	2Exptime-c
BC(desc)	NExptime-c
BC^{-}(child,desc)	undecidable
BC^{+}(child, desc)	NP-c

The satisfiability problem is undecidable

- Reduction from emptiness of a given Minsky Machine M
  - Remember
    - automaton (on words) with 2 counters,
      - instructions: inc(1), inc(2), dec(1), dec(2), tz(1), tz(2)
- The regular language $\mathbf{L}_M$ describes the *shape* of the (data-blind) property of the tree:
  - *"The tree contains a branch labeled with transitions of M."*
- The BC of patterns assures that the labels of the branch form a correct accepting run of M.

The satisfiability problem is undecidable

- The regular language L_M describes the shape of the (data-blind)tree:

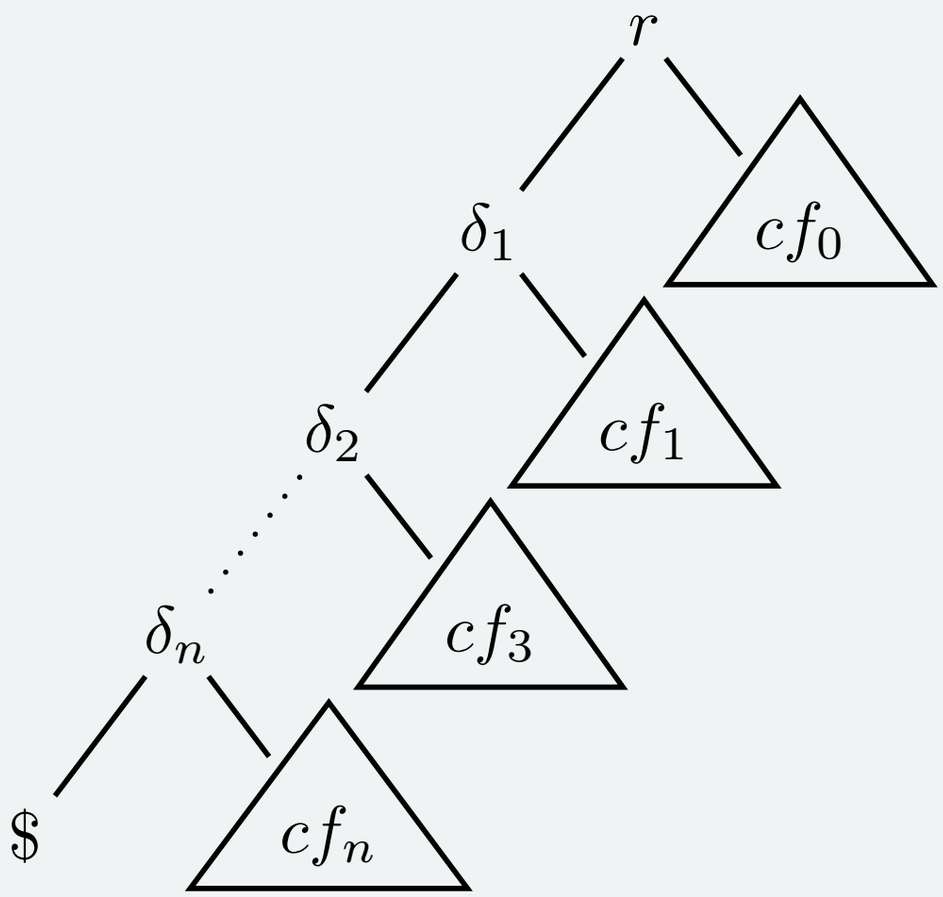
  ★ There is a long branch, labeled with transitions $\delta_1,\delta_2,\ldots$ of M which are consistent:

  ✳ initial/final states at endpoints,

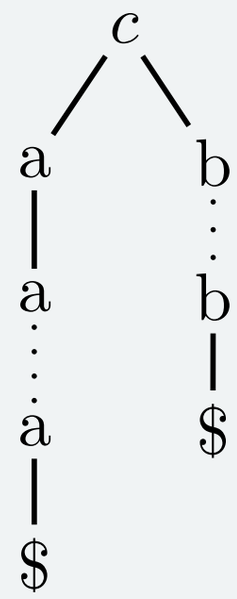
  ✳ source state of δ_{i+1} = target state of δ_i.

★ From each node of this branch, there is a subtree coding the configuration of the counters

The satisfiability problem is undecidable



coding of an execution



coding of a configuration cf_i

The satisfiability problem is undecidable

- The BC of patterns assures that the labels of the branch form a correct accepting run of M

  (I) Every pair of positions in the a-branch of the configuration have different data value.

The satisfiability problem is undecidable

- The BC of patterns assures that the labels of the branch form a correct accepting run of M

  (II) If two a-nodes are at the same index in two successive configurations ⇒ they have the same data value.

The satisfiability problem is undecidable

- The BC of patterns assures that the labels of the branch form a correct accepting run of M

  (III) The length of a-branches in successive configurations is consistent. Eg: if there is an inc(1) then the a-branch increases its length by one.

"it can't be that it decreases ... nor that it increases in more than one"

The satisfiability problem is undecidable

Thus, $L_M \wedge \neg P_1 \wedge \cdots \wedge \neg P_n$ is satisfiable $\Leftrightarrow$ M has an accepting run.

Thus, the satisfiability pb for BC^-(child,desc) is also undecidable.

Other form of patterns: CQ

We consider **conjunctive queries** over the binary relations

- child
- desc
- next-sib
- foll-sib
- data-eq
- data-neq
- a, for $a \in A$

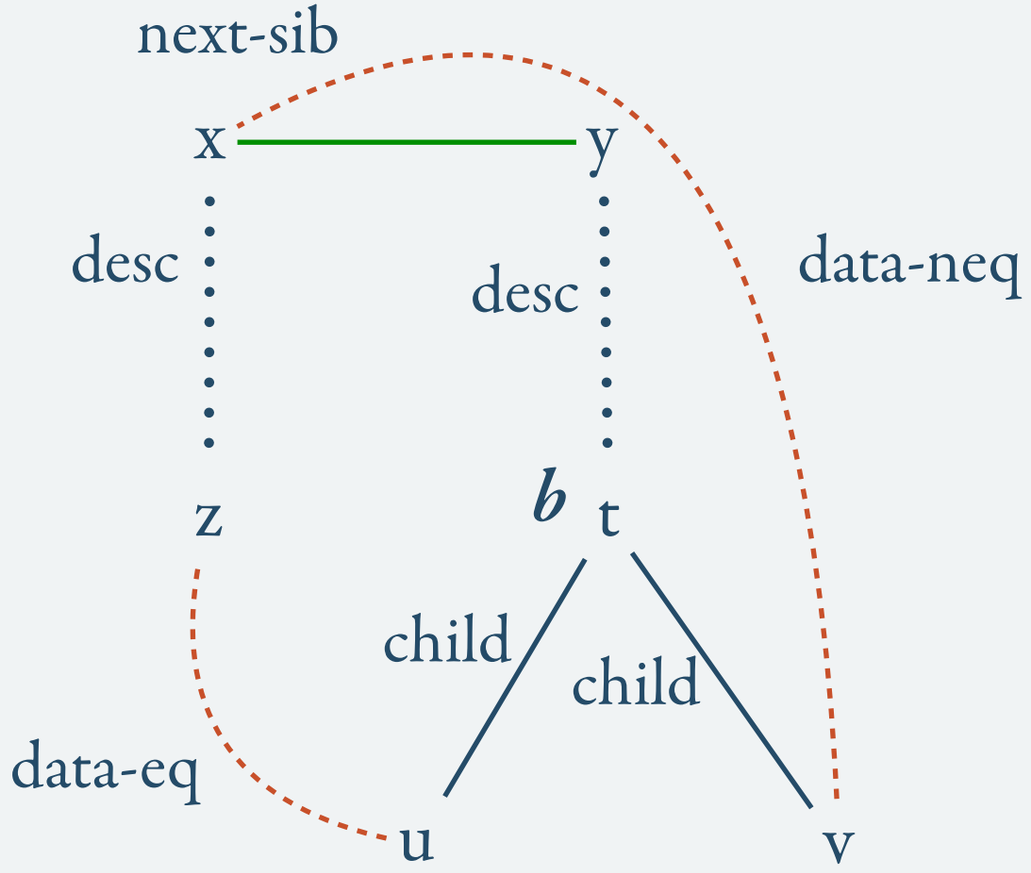
A conjunctive query: a conjunction of atoms over these relations (we can use any number of variables).

Semantics: existence of homomorphism (ie, whether the FO sentence holds in the structure).

Other form of patterns: CQ

Example:

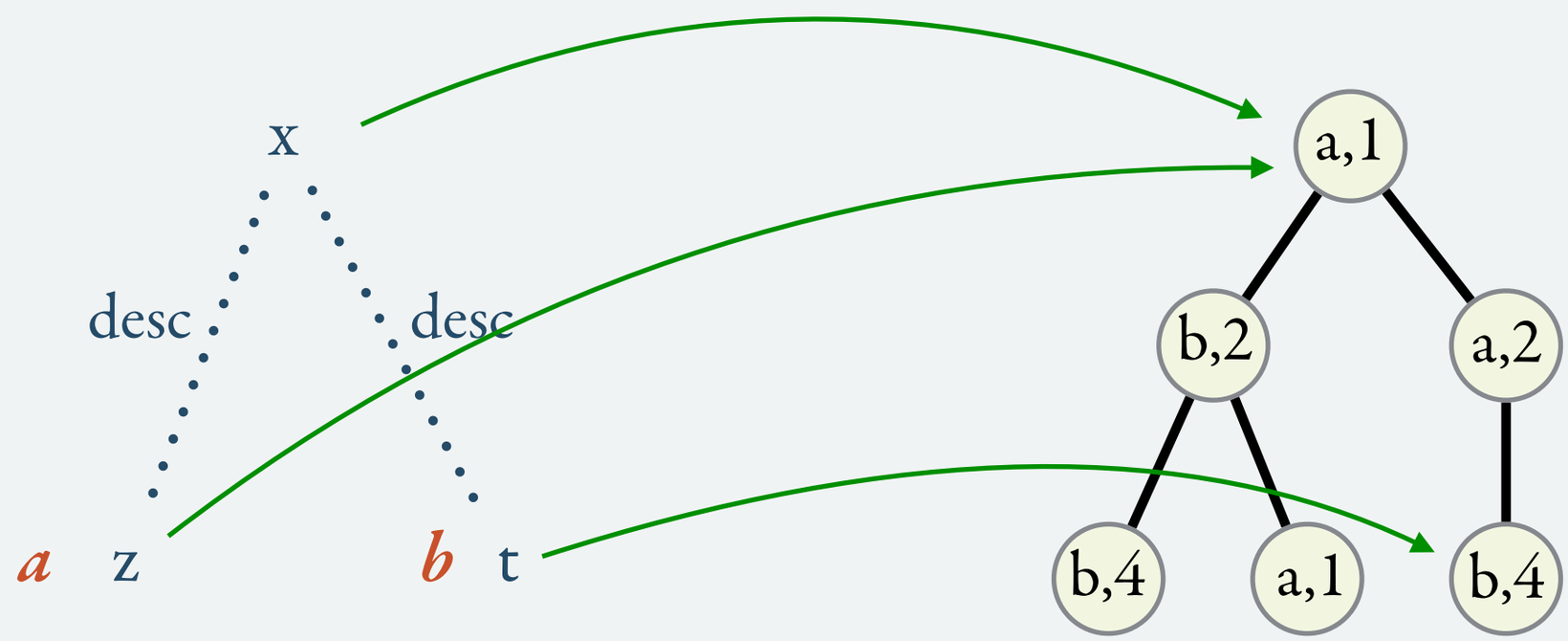
ϕ = desc(x,z) $\wedge$ next-sib(x,y) $\wedge$ desc(y,t) $\wedge$ child(u,t) $\wedge$ child(v,t) $\wedge$ a(t) $\wedge$ data-neq(x,v) $\wedge$ data-eq(z,u) $\wedge$ ***b***(t)



Other form of patterns: CQ

Semantics: existence of homomorphism (ie, whether the FO sentence holds in the structure).

- Akin to XPath with path intersection.
- Non-injective semantics
- Boolean combinations of data tree patterns are **more expressive** but **less succinct** than CQs



Other form of patterns: CQ

Satisfiability of CQ on data trees under regular constraints is NP-complete.

Containment of CQ on data trees under regular constraints is undecidable.

[Bjorklund, Martens, Schwentick]

First-Order logic, an old friend

- We can also use FO^2 on data trees where we have:

  ~ a data equality relation
  child a child relation
  desc a descendant relation
  next-sib a next-sibling relation
  foll-sibl a following-sibling relation

(child(x,y) : "x is a child of y"
  desc(x,y) : "x is a descendant of y"
  next-sib(x,y) : "x is the next sibling of y"
  foll-sib(x,y) : "x is one of the following siblings of y")

First-Order logic

Example 1:

$$\forall x \ (\quad (\neg \exists y \ \text{child}(y,x)) \quad \Leftrightarrow \quad \exists y \ \text{desc}(x,y) \wedge x \sim y \quad)$$

"the only data value that repeats along a branch is that of the leaf"

First-Order logic

Example 2:

$$\forall x \ (\quad \mathbf{a}(x) \quad \Rightarrow \quad \exists y \ \mathbf{b}(y) \neg \text{desc}(x,y) \wedge \neg \text{desc}(y,x) \wedge x \sim y \quad)$$

"for every node there is another one with the same data value in an incomparable position"

First-Order logic

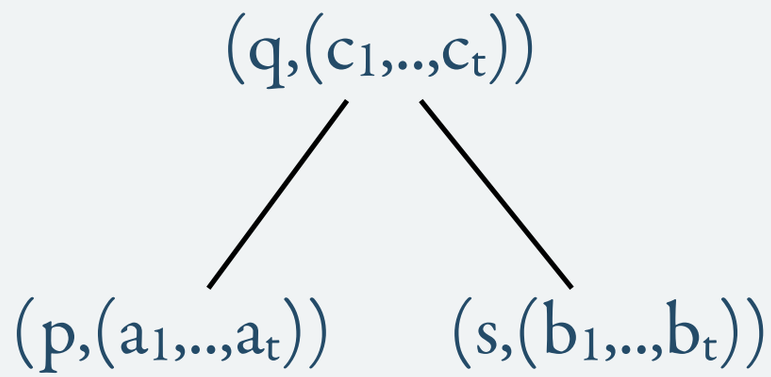
SAT-FO²(~,child,desc,next-sib,foll-sib) is at least as hard as reach-BVASS.

BVASS = "Branching Vector Addition System with States"

It is a branching version of VASS

(brief digression into BVASS)

- Set of states Q
- Set of rules of the form "$q \xrightarrow{inc(j)} p,s$" or "$q \xrightarrow{dec(j)} p,s$"
- Configurations of the form "$(q,(n_1,..,n_t))$"
- Derivation tree: a binary tree labeled with configurations so that for every parent and children



we have:
- ∃ rule $q \xrightarrow{inc(j)} p,s$ or $q \xrightarrow{dec(j)} p,s$
- $a_i + b_i = c_i$ for all $i \neq j$, and
- $a_j + b_j + 1 = c_j$ or $a_j + b_j - 1 = c_j$

- **Reachability problem**: whether there exists a derivation tree where
  - all leaves are labeled $(q_{leaf},(0,\ldots,0))$ and
  - the root is labeled $(q_{root},(0,\ldots,0))$.

(brief digression into BVASS)

Its reachability problem is **unknown** to be decidable,
and it is considered a hard open problem.

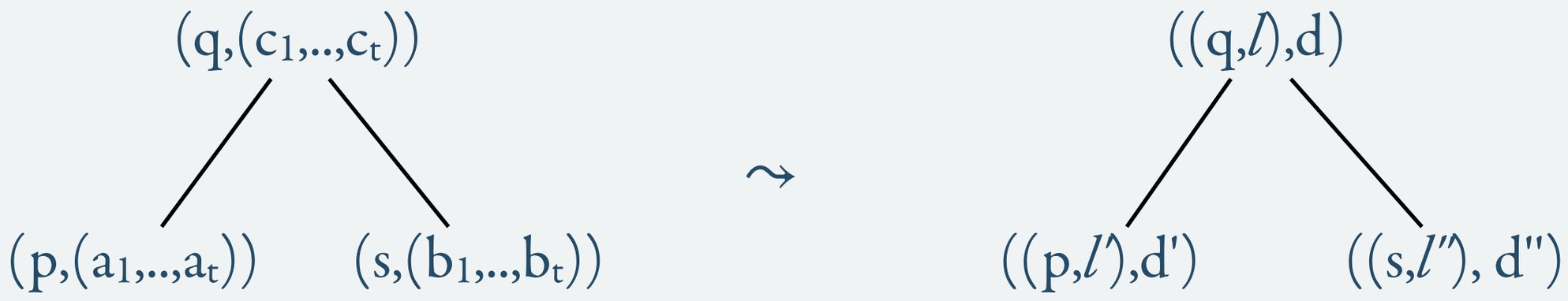
First-Order logic

SAT-FO^2(~,child,desc,next-sib,foll-sib) is at least as hard as reach-BVASS.

Reduction reach-BVASS $\rightsquigarrow$ SAT-FO^2(~,...):

- A decision procedure for SAT-FO^2(~,...) would yield decidability of reach-BVASS.
- However, the converse is not necessarily true.

reach-BVASS $\rightsquigarrow$ Sat-FO^2



$l, l', l'' \in \{\text{inc},\text{dec}\} \times \{1,\ldots t\}$

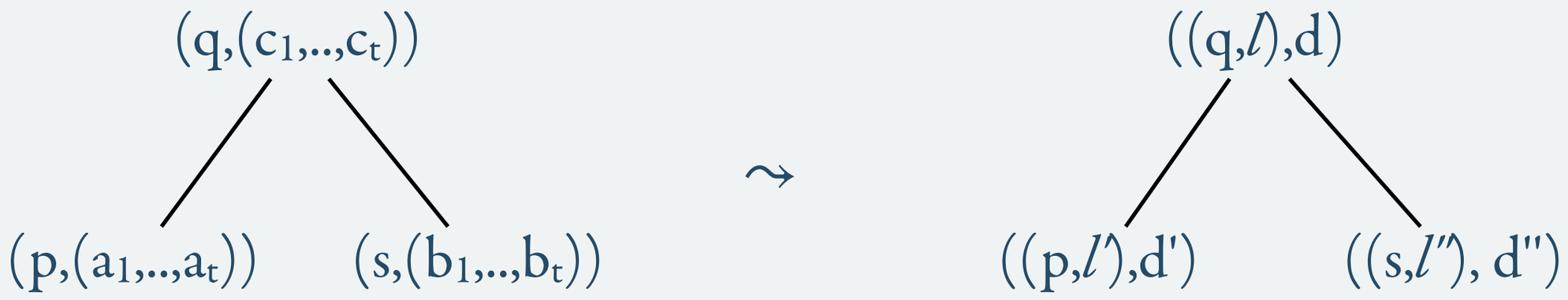
If witnessed by inc(j) then $l = \text{inc(j)}$

"Structure is ok":

- starts with initial states,
- all leaves have final states
- every node has 0 or two children
- for every ((q,l),d) parent of ((p,l'),d'), ((s,l''),d'') there is a rule $q \longrightarrow^{l} p,s$

Note: this needs next-sibling and child relations

reach-BVASS $\rightsquigarrow$ Sat-FO^2



$l, l', l'' \in \{\text{inc},\text{dec}\} \times \{1,\ldots t\}$

If witnessed by inc(j) then $l = \text{inc}(j)$

- Every inc(i) has a different data value along a branch (idem with dec(i))

  $\forall x,y\ (x \neq y) \wedge \phi_{\text{inc}}(x) \wedge \phi_{\text{inc}}(y) \Rightarrow \neg\ (x \sim y)$

  where $\phi_{\text{inc}}(x) = \vee_q\ (q,\text{inc}(i))(x)$
- Every inc(i) has a descending dec(i) with the same data value

  $\forall x\ \ \phi_{\text{inc}}(x) \Rightarrow (\exists y\ \text{desc}(y,x) \wedge \phi_{\text{dec}}(y) \wedge x \sim y)$
- Every dec(i) has an ancestor inc(i) with the same data value

First-Order logic

SAT-FO^2(~,child,next-sib) problem is decidable.

[Bojańczyk&al]

Complexity between NExpTime and 3NExpTime.
The proof is non-trivial.