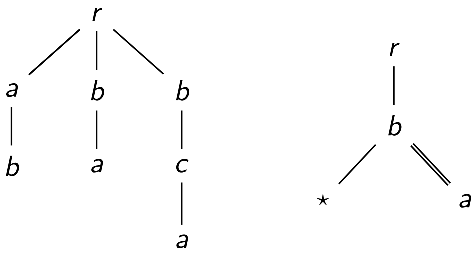
Tree pattern queries: learning and teaching

Sławek Staworko, Piotr Wieczorek

University of Wrocław

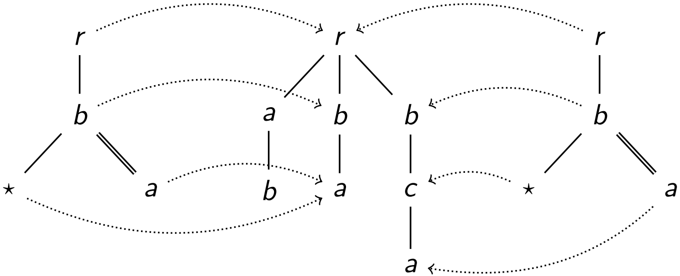
December 16, 2021

Trees and tree patterns (twigs)



(a) Tree t_0. (b) Twig query q_0.

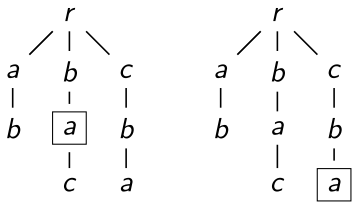
Trees and tree patterns (twigs): embeddings



(c) Embeddings of q_0 in t_0.

Unary tree patterns and decorated trees

(d) Unary path query p_0.



(e) Decorated trees t_1 and t_2.

What is this talk about?

characterizing queries for a given q provide a (polynomial) set of examples CS_q such that for all p it holds $CS_q \subseteq \mathcal{L}(p)$ iff $q \subseteq p$.

What is this talk about?

characterizing queries: for a given q provide a (polynomial) set of examples CS_q such that for all p it holds $CS_q \subseteq \mathcal{L}(p)$ iff $q \subseteq p$.

learning: provide a polynomial-time algorithm $\mathcal{A}$ such that

What is this talk about?

characterizing queries: for a given q provide a (polynomial) set of examples CS_q such that for all p it holds $CS_q \subseteq \mathcal{L}(p)$ iff $q \subseteq p$.

learning: provide a polynomial-time algorithm $\mathcal{A}$ such that

- for any sample S, $\mathcal{A}$ return a query consistent with S (soundness),

What is this talk about?

characterizing queries
: for a given q provide a (polynomial) set of examples CS_q such that for all p it holds $CS_q \subseteq \mathcal{L}(p)$ iff $q \subseteq p$.

learning
: provide a polynomial-time algorithm $\mathcal{A}$ such that
  - for any sample S, $\mathcal{A}$ return a query consistent with S (soundness),
  - for any query q there exists a (polynomial) characteristic sample $CS_q^{\mathcal{A}}$ such that for every sample S that satifies $CS_q^{\mathcal{A}} \subseteq S \subseteq \mathcal{L}(q)$, the algorithm $\mathcal{A}$ returns a query equivalent to q (completeness).

What is this talk about?

characterizing queries for a given q provide a (polynomial) set of examples CS_q such that for all p it holds $CS_q \subseteq \mathcal{L}(p)$ iff $q \subseteq p$.

learning provide a polynomial-time algorithm $\mathcal{A}$ such that

- for any sample S, $\mathcal{A}$ return a query consistent with S (soundness),
- for any query q there exists a (polynomial) characteristic sample $CS_q^{\mathcal{A}}$ such that for every sample S that satifies $CS_q^{\mathcal{A}} \subseteq S \subseteq \mathcal{L}(q)$, the algorithm $\mathcal{A}$ returns a query equivalent to q (completeness).

Here, we focus on samples with positive examples only.

What is this talk about?

characterizing queries for a given q provide a (polynomial) set of examples CS_q such that for all p it holds $CS_q \subseteq \mathcal{L}(p)$ iff $q \subseteq p$.

learning provide a polynomial-time algorithm $\mathcal{A}$ such that

- for any sample S, $\mathcal{A}$ return a query consistent with S (soundness),
- for any query q there exists a (polynomial) characteristic sample $CS_q^{\mathcal{A}}$ such that for every sample S that satifies $CS_q^{\mathcal{A}} \subseteq S \subseteq \mathcal{L}(q)$, the algorithm $\mathcal{A}$ returns a query equivalent to q (completeness).

Minimality vs. completeness?

Arbitrary twigs: problems

- sometimes queries are contained but no embedding exists

Arbitrary twigs: problems

- sometimes queries are contained but no embedding exists, e.g., $r/a//b$ and $r/*/*$

Arbitrary twigs: problems

- sometimes queries are contained but no embedding exists, e.g., $r/a//b$ and $r/*/*$
- containment is coNP-complete, subsumption (i.e., the existence of embeddings) is in PTIME

Arbitrary twigs: problems

- sometimes queries are contained but no embedding exists, e.g., $r/a//b$ and $r/*/*$
- containment is coNP-complete, subsumption (i.e., the existence of embeddings) is in PTIME
- moreover: exponential samples required

Arbitrary twigs require exponential samples

Theorem

For any natural number n there exists a twig query q such that any set characterizing q contains at least 2^n examples.

Arbitrary twigs require exponential samples

Theorem

For any natural number n there exists a twig query q such that any set characterizing q contains at least 2^n examples.

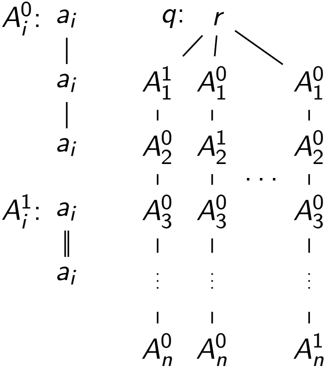
Proof.

Construct a query q and a set of twig queries U such that:

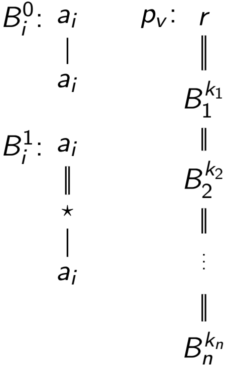
1. for each $p \in U$ we have $q \not\subseteq p$;
2. no single positive example t can witness the fact that any two distinct $p_1, p_2 \in U$ are not equivalent to q;
3. U contains 2^n queries.

$\square$

Arbitrary twigs require exponential samples



(f) Query q



(g) Query p_v for $v = (k_1, \ldots, k_n)$.

Anchored tree patterns: a good class

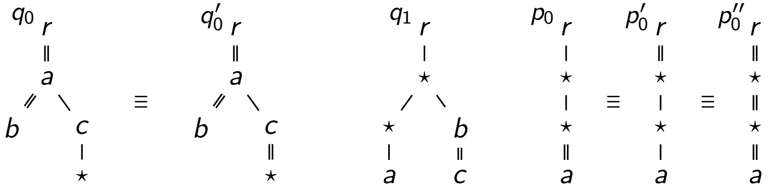
A twig query is *anchored* if:

1. A //-edge can be incident to a $\star$-node only if the node is a leaf.
2. A $\star$-node may be a leaf only if it is either incident to a //-edge or it is the selecting node.

Anchored tree patterns: a good class

A twig query is *anchored* if:

1. A //-edge can be incident to a $\star$-node only if the node is a leaf.
2. A $\star$-node may be a leaf only if it is either incident to a //-edge or it is the selecting node.



not anchored   anchored   anchored   not anchored

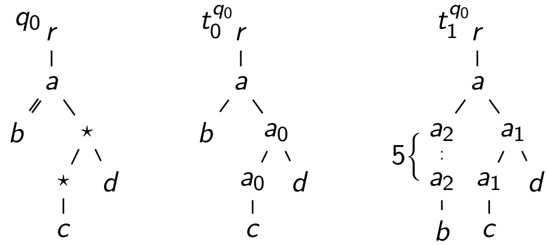
Anchored tree patterns: a good class

Theorem

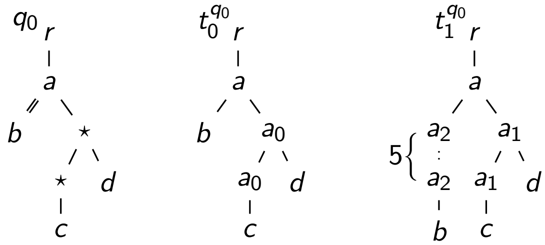
For anchored twigs:

- P1. *subsumption ≡ containment;*
- P2. *a small characteristic sample always exists (two trees are enough!).*

P2: Containment characterizing trees



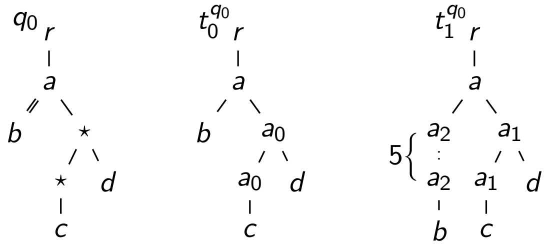
P2: Containment characterizing trees



Lemma

Take any anchored query q and construct t_1^q as above. For any p of height bounded by the height of q and not using labels a_1 and a_2, $t_1^q \preccurlyeq p$ implies $q \preccurlyeq p$.

P2: Containment characterizing trees



Lemma

Take any anchored query q and construct t_1^q as above. For any p of height bounded by the height of q and not using labels a_1 and a_2, $t_1^q \preccurlyeq p$ implies $q \preccurlyeq p$.

Corollary

If both t_0^q and t_1^q satisfy p, then $q \subseteq p$.

Learning for unary anchored path queries

Input: a sample S of decorated trees
Output: a minimal unary anchored path query p such that $S \subseteq \mathcal{L}(p)$

Learning for unary anchored path queries

$w := \min_{\leq_{can}}(SelPath(S))$
let w be of the form $r/a_1/\cdots/a_n$
$p := r//\star$
foreach subpath u of $a_1/a_2/\cdots/a_{n-1}$
in the order of decreasing lengths **do**
replace in p any $//$-edge by $//u//$ as long as $S \subseteq \mathcal{L}(p)$

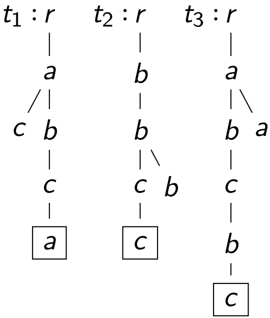
Learning for unary anchored path queries

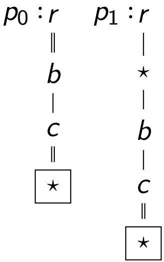
let p be of the form $r//p_0//b_1$
if $S \subseteq \mathcal{L}(p\{b_1 \leftarrow a_n\})$ **then**
$p := p\{b_1 \leftarrow a_n\}$

Learning for unary anchored path queries

foreach descendant edge α in p **do**
find maximal ℓ s.t. $S \subseteq \mathcal{L}(p\{\alpha \leftarrow /\!/(\star/)^{\ell}\})$
if $S \subseteq \mathcal{L}(p\{\alpha \leftarrow /(\star/)^{\ell}\})$ **then**
$p := p\{\alpha \leftarrow /(\star/)^{\ell}\}$
return p

A sample and the constructed queries.





Learning unary anchored path queries

Theorem

Our algorithm is sound and complete (i.e., it is sound and returns q if the input is S such that $S \supseteq CS_q = \{t_0^q, t_1^q\}$).

Proof.

- Assume some p is returned for a given sample S.

□

Learning unary anchored path queries

Theorem

Our algorithm is sound and complete (i.e., it is sound and returns q if the input is S such that $S \supseteq CS_q = \{t_0^q, t_1^q\}$).

Proof.

- Assume some p is returned for a given sample S.
- Obviously, $S \subseteq \mathcal{L}(p)$.

$\square$

Learning unary anchored path queries

Theorem

Our algorithm is sound and complete (i.e., it is sound and returns q if the input is S such that $S \supseteq CS_q = \{t_0^q, t_1^q\}$).

Proof.

- Assume some p is returned for a given sample S.
- Obviously, $S \subseteq \mathcal{L}(p)$.
- By case analysis: there is no $p' \neq p$ such that $p' \subseteq p$ and $S \subseteq \mathcal{L}(p')$.

□

Learning unary anchored path queries

Theorem

Our algorithm is sound and complete (i.e., it is sound and returns q if the input is S such that $S \supseteq CS_q = \{t_0^q, t_1^q\}$).

Proof.

- Assume some p is returned for a given sample S.
- Obviously, $S \subseteq \mathcal{L}(p)$.
- By case analysis: there is no $p' \neq p$ such that $p' \subseteq p$ and $S \subseteq \mathcal{L}(p')$.
- Since $CS_q \subseteq S \subseteq \mathcal{L}(p)$ then $q \subseteq p$ (P2). Hence, by the claim above, p and q are equivalent.

$\square$

Learning anchored queries

Our algorithm can be extended for

- boolean anchored path queries,
- conjunctions of anchored path queries,
- (path-subsumption-free) boolean twigs,
- (path-subsumption-free) unary twigs.