Deciding Equivalence of Top-Down Tree Transducers

Sebastian Maneth
National ICT Australia Ltd. & UNSW, Sydney

Joint work w. Helmut Seidl (TU Munich)

May 9th, 2007 - Mostrare Seminar, Lille

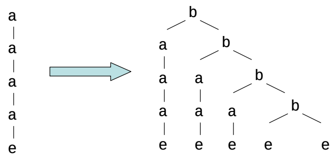
Prologue *Tree Transducers*

= (finitely described) *models for relations* on (ordered) *trees*

E.g. → finite-state (generalize FTA to input + output)

Example top-down tree transducer (TOP) [Rounds/Thatcher, 70's]

```
q0(a(x)) → b(q(x),q0(x))
q0(e)    → e
 q(a(x)) → a(q(x))
 q(e)    → e
```



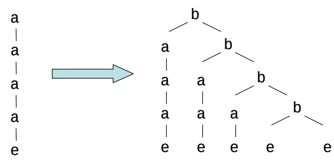
Prologue *Tree Transducers*

= (finitely described) *models for relations* on (ordered) *trees*

E.g. → finite-state (generalize FTA to input + output)

Example top-down tree transducer (TOP) [Rounds/Thatcher, 70's]

```
q0(a(x)) → b(q(x),q0(x))        p0(a(x)) → p(x)
q0(e)    → e                    p0(e)    → e
 q(a(x)) → a(q(x))               p(a(x)) → b(a(q(x)),p(x))
 q(e)    → e                     p(e)    → b(e,e)
```



Prologue *Tree Transducers*

= (finitely described) *models for relations* on (ordered) *trees*

E.g. → finite-state (generalize FTA to input + output)

Example top-down tree transducer (TOP) [Rounds/Thatcher, 70's]

```
q0(a(x)) → b(q(x),q0(x))        p0(a(x)) → p(x)
q0(e)    → e                    p0(e)    → e
 q(a(x)) → a(q(x))               p(a(x)) → b(a(q(x)),p(x))
 q(e)    → e                     p(e)    → b(e,e)
```

M1 *is equivalent to* M2

Transducers T1, T2 are **equivalent** iff ∀ input s: T1(s) = T2(s).

Theorem [Esik80]

For two deterministic TOPs it is **decidable** if they are **equivalent**.

Proof idea

Build tree automaton that keeps track of "difference trees".
CAVE Those trees can be very large! Complexity?!

Theorem [Esik80]

For two deterministic TOPs it is **decidable** if they are **equivalent**.

Proof idea

Build tree automaton that keeps track of "difference trees".
CAVE Those trees can be very large! Complexity?!

Our Contribution

Canonical normal form for TOPs: "*uniform and earliest*"

Theorem
Uniform&Earliest(T1) is **isomorphic** to Uniform&Earliest(T2)
if and only if M1 is **equivalent** to M2.

If M is total, then Uniform&Earliest(M) obtained in PTIME.

Outline

- Equivalence Problems of Transducers
- Top-Down Tree Transducers
- Uniform & Earliest
- Regular Look-Ahead
- Applications
  → Inclusion of XML Queries

Equivalence Problems of String / Tree Transducers

• *nondeterministic (one-way) finite state transducers* **undecidable** [Griffiths68]
(→ reduction from PCP, use complement and union)

• *deterministic (one-way) finite state transducers* **decidable** [Gurari82]
(→ use Parikh property)

• *deterministic top-down tree transducers* **decidable** [Esik80]

• *nonnested, seperated attributed/marco tree transducers* **decidable** [Courcelle/Franchi-Zannetacci82]
→ seperated = can be evaluated in two phases,
(1) only inherited, over Δ_{inh} (2) only synthesized, over Δ_{syn}

• *MSO definable tree transducers* **decidable** [Engelfriet/Maneth05]
(→ use Parikh property)

Deterministic Top-Down Tree Transducers (TOPs)

T = (Q, Σ, Δ, δ, A)

$t[t_1,t_2,\ldots,t_r]$ = replace k-th ⊤ in t by t_k

Q finite set of states
Σ, Δ ranked alphabets of input/output symbols
$\delta(q, a) = p[\, q_1(x_{i_1}), \ldots, q_r(x_{i_r})\,]$

where p is a pattern over Δ (tree over $\Delta \cup \{\top^{(0)}\}$)
$i_1, \ldots, r_r \in \{1, 2, \ldots, \text{rank}(a)\}$

A = axiom = $p[\, q_1(x_0), \ldots, q_r(x_0)\,]$

```
q0(a(x)) → b(q(x),q0(x))
q0(e)    → e
 q(a(x)) → a(q(x))
 q(e)    → e
```

```
δ(q0,a)=b(⊤,⊤)[q(x1),q0(x1)]
δ(q0,e)=e
...
A = ⊤[q0(x0)]
```

[[q]] partial function from T_Σ to T_Δ

DOM(q) = domain of [[q]]

$t[t_1,t_2,\ldots,t_r]$ = replace k-th ⊤ in t by t_k

e.g.: `b(a(⊤),b(⊤,⊤))[t1,t2,t3] = b(a(t1),b(t2,t3))`

pattern order ⊥ ⊑ p for all p∈ P_Σ
p ⊑ u if ∃ p1,…,pk: p = u[p1,…,pk] |p| ≥ |u|

⊑ "less exact" "replace subtrees by ⊤"

e.g. `b(a(⊤),b(⊤,⊤)) ⊑ b(⊤,b(⊤,⊤)) ⊑ b(⊤,⊤)` ⊑ ⊤
and
`b(a(⊤),b(⊤,⊤)) ⊑ b(a(⊤),⊤) ⊑ b(⊤,⊤)`

$t[t_1,t_2,\ldots,t_r]$ = replace k-th ⊤ in t by t_k

e.g.: `b(a(⊤),b(⊤,⊤))[t1,t2,t3] = b(a(t1),b(t2,t3))`

pattern order ⊥ ⊑ p for all p∈ P_Σ
p ⊑ u if ∃ p1,…,pk: p = u[p1,…,pk] |p| ≥ |u|

⊑ "less exact" "replace subtrees by ⊤"

e.g. `b(a(⊤),b(⊤,⊤)) ⊑ b(⊤,b(⊤,⊤)) ⊑ b(⊤,⊤)` ⊑ ⊤
and
`b(a(⊤),b(⊤,⊤)) ⊑ b(a(⊤),⊤) ⊑ b(⊤,⊤)`

least upper bound (unique!)

$\sqcup\{p_1, p_2, \ldots, p_n\}$ = nodes (paths) common to all p_i

e.g. `⊔{ b(e,e), b(a(⊤,⊤),⊤) } = b(⊤,⊤)`

Uniform & Earliest TOPs

TOP T = (Q, Σ, Δ, δ, A) with Σ = { $a^{(2)}, e^{(0)}$ }

$q_1(a(x_1,x_2)) \to q(x_1)$
$q_2(a(x_1,x_2)) \to q(x_2)$
$q(e) \to e$

DOM(q) = { e }

$DOM(q_1)$ = { a(e,t) | t ∈ T_Σ }

$DOM(q_2)$ = { a(t,e) | t ∈ T_Σ }

A = $g(q_1(x_0),q_2(x_0))$

DOM(T) = $DOM(q_1) \cap DOM(q_2)$ = { a(e,e) }

→ would like to have $DOM(q_1) = DOM(q_2)$
if q_1 and q_2 translate the same input

Uniform & Earliest TOPs

TOP $T = (Q, \Sigma, \Delta, \delta, A)$ with $\Sigma = \{ a^{(2)}, e^{(0)} \}$

```
q1(a(x1,x2)) → q(x1)
q2(a(x1,x2)) → q(x2)
 q(e)        → e
```

DOM(q) = { e }

DOM(q_1) = { a(e,t) | $t \in T_\Sigma$ }

DOM(q_2) = { a(t,e) | $t \in T_\Sigma$ }

A = $g(q_1(x_0), q_2(x_0))$

DOM(T) = DOM(q_1) ∩ DOM(q_2) = { a(e,e) }

NOT possible! (wo chaning the output..)

→ would like to have DOM(q_1) = DOM(q_2) if q_1 and q_2 translate the same input

Uniform & Earliest TOPs

TOP $T = (Q, \Sigma, \Delta, \delta, A)$ with $\Sigma = \{ a^{(2)}, e^{(0)} \}$

```
q1(a(x1,x2)) → q(x1)
q2(a(x1,x2)) → q(x2)
 q(e)        → e
```

DOM(q) = { e }   A = $g(q_1(x_0), q_2(x_0))$
DOM(q_1) = { a(e,t) | $t \in T_\Sigma$ }
DOM(q_2) = { a(t,e) | $t \in T_\Sigma$ }

Top-down deterministic tree automaton D for DOM(T):

Initial state = { q ∈ Q | q occurrs in A }

for state S ⊆ Q, and a∈Σ of rank k:

if δ(q,a) undefined for a q∈ S, then d(S, a) undefined
otherwise d(S, a) = (S_1,…, S_k) where S_i consists of all q' s.t. q'(x_i) occurs in δ(q,a), q∈ S

Uniform & Earliest TOPs

TOP $T = (Q, \Sigma, \Delta, \delta, A)$ with $\Sigma = \{ a^{(2)}, e^{(0)} \}$

```
q1(a(x1,x2)) → q(x1)
q2(a(x1,x2)) → q(x2)
 q(e)        → e
```

DOM(q) = { e }   A = $g(q_1(x_0), q_2(x_0))$
DOM(q_1) = { a(e,t) | $t \in T_\Sigma$ }
DOM(q_2) = { a(t,e) | $t \in T_\Sigma$ }

Top-down deterministic tree automaton D for DOM(T):

Initial state = $\{q_1, q_2\}$

```
d({q1,q2},a) = ({q},{q})
d({q},e)     = ()
```

Uniform & Earliest TOPs

TOP $T = (Q, \Sigma, \Delta, \delta, A)$ with $\Sigma = \{ a^{(2)}, e^{(0)} \}$

```
q1(a(x1,x2)) → q(x1)
q2(a(x1,x2)) → q(x2)
 q(e)        → e
```

DOM(q) = { e }   A = $g(q_1(x_0), q_2(x_0))$
DOM(q_1) = { a(e,t) | $t \in T_\Sigma$ }
DOM(q_2) = { a(t,e) | $t \in T_\Sigma$ }

Top-down deterministic tree automaton D for DOM(T):

Initial state = $\{q_1, q_2\}$

```
d({q1,q2},a) = ({q},{q})
d({q},e)     = ()
```

q_1 q_2 —ρ→ DOM({q_1,q_2}) = {a(e,e)}

q —ρ→ DOM({q}) = {e}

relevant domain

ρ ***relevance map*** of T

Uniform & Earliest TOPs

Def. TOP T = (Q,Σ,Δ,δ,A)
minimal dtta D = (Q_T, Σ, d, B_T) for DOM(T)   (B_T: Initial state)

Then T is **uniform**, if

(1) every state is reachable (= occurs in a successful computation)

(2) there is a unique mapping ρ: Q → Q_T such that

(a) ∅ ⊊ DOM(ρ(q)) ⊆ DOM(q) for all q

(b) ρ(q) = B_T for all q occurring in A

(c) if $\delta(q,a) = p[q_1(x_{i_1}),\dots,q_r(x_{i_r})]$, then D has $d(\rho(q),a) = (B_1,\dots,B_k)$ such that for all j, $\rho(q_j) = B_{i_j}$

Uniform & Earliest TOPs

Def. TOP T = (Q,Σ,Δ,δ,A)
minimal dtta D = (Q_T, Σ, d, B_T) for DOM(T)   (B_T: Initial state)

Then T is **uniform**, if

(1) every state is reachable (= occurs in a successful computation)

(2) there is a unique mapping ρ: Q → Q_T such that

(a) ∅ ⊊ DOM(ρ(q)) ⊆ DOM(q) for all q

(b) ρ(q) = B_T for all q occurring in A

(c) if $\delta(q,a) = p[q_1(x_{i_1}),\dots,q_r(x_{i_r})]$, then A has $d(\rho(q),a) = (B_1,\dots,B_k)$ such that for all j, $\rho(q_j) = B_{i_j}$

Note
Every total transducer is uniform!

Lemma For every TOP T, an equivalent uniform transducer T' can be constructed in **time $O(2^{|Q|})$**.

Proof. ..easy.. →change q into corresponding <q,B> and let ρ(<q,B>)=B

Uniform & **Earliest** TOPs

Uniform TOP (T, D, ρ)

[[q]](s) := outputs of T, starting in state q

(largest) Common Prefix Pattern — q's relevant domain

pref(q) := ⊔{ [[q]](s) | s ∈ DOM(ρ(q)) }

```
p0(a(x)) → p(x)
p0(e)    → e
p(a(x))  → b(a(q(x)),p(x))
p(e)     → b(e,e)
q(a(x))  → a(q(x))
q(e)     → e
```

"output of q **always** starts with **pref**(q), no matter what input is.."

pref(p0) = ⊤
pref(p) = **b(** ⊤, ⊤ **)**

Uniform & **Earliest** TOPs

Uniform TOP (T, D, ρ)

[[q]](s) := outputs of T, starting in state q

(largest) Common Prefix Pattern — q's relevant domain

pref(q) := ⊔{ [[q]](s) | s ∈ DOM(ρ(q)) }

```
p0(a(x)) → p(x)
p0(e)    → e
p(a(x))  → b(a(q(x)),p(x))
p(e)     → b(e,e)
q(a(x))  → a(q(x))
q(e)     → e
```

"output of q **always** starts with **pref**(q), no matter what input is.."

pref(p0) = ⊤
pref(p) = **b(** ⊤, ⊤ **)**

Definition

A uniform TOP is **earliest** if **pref**(q)=⊤ for all states q.

Uniform and Earliest TOPs

Uniform TOP (T, D, ρ)

Theorem
pref(q) can be computed in time $O(|T| \cdot (\log(n+1) + \eta(T)^2))$
η(T)=maximal size of a minimal output tree produced by any state.
n = number of states of D

Proof. Fixpoint iteration.

System of in-equations

$Y_q \sqsupseteq p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})]$ if $\delta(q,a) = p[q_1(x_{i_1}),\ldots,q_r(x_{i_r})]$

"Y_q is a *prefix pattern* of p[…]" ($|Y_q| \leq |p[\ldots]|$)

We will want to compute the *least solution*, w.r.t. ⊑.

(that is: the *largest common pref pattern* of q)

Uniform and Earliest TOPs

Uniform TOP (T, D, ρ)

Theorem
pref(q) can be computed in time $O(|T| \cdot (\log(n+1) + \eta(T)^2))$
η(T)=maximal size of a minimal output tree produced by any state.
n = number of states of D

Proof. Fixpoint iteration.

(1) *Initialization* take any small tree $s_q \in DOM(\rho(q))$
Initial t_q := $[[q]](s_q)$

A (DAG) representation of trees s_q can be computed
In time $O(|T| \cdot \log(n+1))$

Why?
→ Least solution of system of equations with MIN, ADD, MULT
Can be computed in time $O(|S| \cdot \log |S|)$ [Seidl1994]

If D has $d(B,a) = (B_1,B_2,\ldots, B_n)$ then $|S_B| = 1 + \min(S_{B_1}, S_{B_2}, \ldots, S_{B_n})$

Uniform and Earliest TOPs

Uniform TOP (T, D, ρ)

Theorem
pref(q) can be computed in time $O(|T| \cdot (\log(n+1) + \eta(T)^2))$
η(T)=maximal size of a minimal output tree produced by any state.
n = number of states of D

Proof. Fixpoint iteration.

(1) Initialization: take any small tree $s_q \in DOM(\rho(q))$
Initial t_q := $[[q]](s_q)$

(2) At most η(T) iterations needed.
In each iteration, at most |T| variables are updated,
and each update takes at most time O(η(T)).

Uniform and **Earliest TOPs**

Theorem

Every TOP T is equivalent to an **earliest TOP** T'.

If T is total, then T' can be constructed in time $O(|T|^3)$.

If T is uniform, then in time $O(|T| \cdot (\log(n+1) + \eta(T)^2))$

```
p0(a(x)) → p(x)
p0(e)    → e
p(a(x))  → b(a(q(x)),p(x))
p(e)     → b(e,e)
q(a(x))  → a(q(x))
q(e)     → e
```

Idea: replace p(x) by **pref**(p)

Uniform and **Earliest TOPs**

Theorem

Every TOP T is equivalent to an **earliest TOP** T'.

If T is total, then T' can be constructed in time $O(|T|^3)$.

If T is uniform, then in time $O(|T| \cdot (\log(n+1) + \eta(T)^2))$

```
p0(a(x))  → b(<p,1>(x),<p,2>(x))
p0(e)     → e

p(a(x))   → b(a(q(x)),p(x))
p(e)      → b(e,e)

q(a(x))   → a(q(x))
q(e)      → e
```

Idea: replace `p(x)` by **pref**(p)

Uniform and **Earliest TOPs**

Theorem

Every TOP T is equivalent to an **earliest TOP** T'.

If T is total, then T' can be constructed in time $O(|T|^3)$.

If T is uniform, then in time $O(|T| \cdot (\log(n+1) + \eta(T)^2))$

```
p0(a(x))  → b(<p,1>(x),<p,2>(x))
p0(e)     → e
                               <p,1>(a(x))  → a(q(x))
p(a(x))   → b(a(q(x)),p(x))    <p,1>(e)     → e
p(e)      → b(e,e)             <p,2>(a(x))  → b(<p,1>(x),<p,2>(x
                               <p,2>(e)     → e
q(a(x))   → a(q(x))
q(e)      → e
```

Idea: replace `p(x)` by **pref**(p) & also in new rules

Uniform and **Earliest TOPs**

Theorem

Every TOP T is equivalent to an **earliest TOP** T'.

If T is total, then T' can be constructed in time $O(|T|^3)$.

If T is uniform, then in time $O(|T| \cdot (\log(n+1) + \eta(T)^2))$

```
p0(a(x))     → b(<p,1>(x),<p,2>(x))
p0(e)        → e
<p,1>(a(x))  → a(q(x))
<p,1>(e)     → e
<p,2>(a(x))  → b(<p,1>(x),<p,2>(x))
<p,2>(e)     → e
q(a(x))      → a(q(x))
q(e)         → e
```

Idea: replace `p(x)` by **pref**(p)

Minimization of Earliest TOPs

```
p0(a(x))     → b(<p,1>(x),<p,2>(x))
p0(e)        → e
<p,1>(a(x))  → a(q(x))
<p,1>(e)     → e
<p,2>(a(x))  → b(<p,1>(x),<p,2>(x))
<p,2>(e)     → e
q(a(x))      → a(q(x))
q(e)         → e
```

$\sim$ is largest relation $\sim'$ on states st $q \sim' q'$ implies

→ $\rho(q) = \rho(q')$
→ $\forall a$: $\delta(q,a)$ and $\delta(q',a)$ have equal pattern p and $q_i \sim' q_i'$

Minimization of Earliest TOPs

```
p0(a(x))     → b(<p,1>(x),<p,2>(x))
p0(e)        → e
<p,1>(a(x))  → a(q(x))
<p,1>(e)     → e
<p,2>(a(x))  → b(<p,1>(x),<p,2>(x))
<p,2>(e)     → e
q(a(x))      → a(q(x))
q(e)         → e
```

$\sim$ is largest relation $\sim'$ on states st $q \sim' q'$ implies

→ $\rho(q) = \rho(q')$
→ $\forall a$: $\delta(q,a)$ and $\delta(q',a)$ have equal pattern p and $q_i \sim' q_i'$

Clearly: `p0` $\sim$ `<p,2>`

Minimization of Earliest TOPs

```
p0(a(x))     → b(<p,1>(x),p0(x))
p0(e)        → e
<p,1>(a(x))  → a(q(x))
<p,1>(e)     → e
<p,2>(a(x))  → b(<p,1>(x),<p,2>(x))
<p,2>(e)     → e
q(a(x))      → a(q(x))
q(e)         → e
```

$\sim$ is largest relation $\sim'$ on states st $q \sim' q'$ implies

→ $\rho(q) = \rho(q')$
→ $\forall a$: $\delta(q,a)$ and $\delta(q',a)$ have equal pattern p and $q_i \sim' q_i'$

Clearly: `p0` $\sim$ `<p,2>`

Minimization of Earliest TOPs

```
p0(a(x))  → b(<p,1>(x),p0(x))
p0(e)     → e
<p,1>(a(x)) → a(q(x))
<p,1>(e)    → e

q(a(x))   → a(q(x))
q(e)      → e
```

~ is largest relation ~' on states st q ~' q' implies

→ ρ(q) = ρ(q')
→ ∀a: δ(q,a) and δ(q',a) have equal pattern p
and q_i ~' q_i'

Clearly: p0 ~ <p,2>

Minimization of Earliest TOPs

```
p0(a(x))  → b(<p,1>(x),p0(x))
p0(e)     → e
<p,1>(a(x)) → a(q(x))
<p,1>(e)    → e

q(a(x))   → a(q(x))
q(e)      → e
```

~ is largest relation ~' on states st q ~' q' implies

→ ρ(q) = ρ(q')
→ ∀a: δ(q,a) and δ(q',a) have equal pattern p
and q_i ~' q_i'

Clearly: p0 ~ <p,2>

ρ(<p,1>) = ρ(q) Because dtta is minimal!!

Minimization of Earliest TOPs

```
p0(a(x))  → b(q(x),p0(x))
p0(e)     → e
<p,1>(a(x)) → a(q(x))
<p,1>(e)    → e

q(a(x))   → a(q(x))
q(e)      → e
```

~ is largest relation ~' on states st q ~' q' implies

→ ρ(q) = ρ(q')
→ ∀a: δ(q,a) and δ(q',a) have equal pattern p
and q_i ~' q_i'

Clearly: p0 ~ <p,2>

ρ(<p,1>) = ρ(q) Because dtta is minimal!!

Thus: q ~ <p,1>

Minimization of Earliest TOPs

```
p0(a(x))  → b(q(x),p0(x))
p0(e)     → e

q(a(x))   → a(q(x))
q(e)      → e
```

~ is largest relation ~' on states st q ~' q' implies

→ ρ(q) = ρ(q')
→ ∀a: δ(q,a) and δ(q',a) have equal pattern p
and q_i ~' q_i'

Clearly: p0 ~ <p,2>

ρ(<p,1>) = ρ(q) Because dtta is minimal!!

Thus: q ~ <p,1>

Minimization of Earliest TOPs

```
p0(a(x))  → b(q(x),p0(x))
p0(e)     → e
q(a(x))   → a(q(x))
q(e)      → e
```

Minimal, Earliest transducer

Theorem

(1) ~ can be computed in Polynomial time wrt |T|

(2) Replace each state q by its equivalence class wrt ~
(gives a "canonical transducer" M)

Canonical transducers M1, M2 (i.e., each equiv class of ~ is singleton):

M1 equivalent to M2 **iff** M1= state-renaming(M2)

Equivalence of TOPs with regular look-ahead

M = (Q, Σ, Δ, δ, A, B)

B det. bottom-up tree automaton
With L(p1)∩L(p2)=∅ for all states p1,p2

q(a(x1,x2)) → t <p1,p2>

Given TOPs w la M1, M2:

Change input symbol a into <a, (p1, p2), (u1, u2)>
Then M1, M2 become ordinary TOPs (without lookahead)

Now, change M1, M2 so that they
check if input tree is a correct relabeling wrt the automata B1, B2.

new axiom A1' = #(f(x_0), A1)

Where f = (init(B1), init(B2)) -- does partial identity wrt B1 × B2

M1' equiv M2' **iff** M1 equiv M2

Applications

→ *XML query optimization*

Assume result to query Q1 is already materialized.

Given a new query Q2, check if Q2 equivalent to Q1.
If so, return materialized result, instead of recomputing.

Applications

→ *XML query optimization*

Assume result to query Q1 is already materialized.

Given a new query Q2, check if Q2 equivalent to Q1.
If so, return materialized result, instead of recomputing.

Possible extension

Q1 "subsumes" Q2, if there exists a TOP Q3 such that
for all inputs s,
Q2(s) = Q3(Q1(s))

Decidable?
Seems difficult…

Applications

→ *XML query optimization*

Assume result to query Q1 is already materialized.

Given a new query Q2, check if Q2 equivalent to Q1.
If so, return materialized result, instead of recomputing.

Possible extension

Q1 "subsumes" Q2, if there exists a TOP Q3 such that
for all inputs s,
Q2(s) = Q3(Q1(s))

Decidable?
Seems difficult…

→ Probably **UNDECIDABLE… ☹**

Applications

→ *XML query optimization*

Assume result to query Q1 is already materialized.

Given a new query Q2, check if Q2 equivalent to Q1.
If so, return materialized result, instead of recomputing.

Possible extension (mild) (replacing by ⊤)

Q1 "subsumes" Q2, if for all inputs s,
Q2(s) can be obtained by deleting subtrees from Q1(s). Q1(s) ⊑ Q2(s)

Conjecture
Given uniform and earliest Q1,Q2, it is decidable whether
or not Q1 subsumes Q2.

Given a new query Q2, check if Q1 subsumes Q2.
If so, return materialized result, with appropriate subtrees removed.

Applications

→ *XML query optimization*

Assume result to query Q1 is already materialized.

Given a new query Q2, check if Q2 equivalent to Q1.
If so, return materialized result, instead of recomputing.

→ What about extension to

Unranked top-down tree transducers? To **basic Macro** tree transducers?

→ Can we decide other interesting subsumption orderings?

e.g. permutation of subtrees
deletion of paths
… ?

Applications

→ *XML query optimization*

Assume result to query Q1 is already materialized.

Given a new query Q2, check if Q2 equivalent to Q1.
If so, return materialized result, instead of recomputing.

→ What about extension to

Unranked top-down tree transducers? To **basic Macro** tree transducers?

→ Can we decide other interesting subsumption orderings?

e.g. permutation of subtrees
deletion of paths
… ?

(not) … THE **END**