

Deciding Equivalence of Top-Down XML Transformations in Polynomial Time

Sebastian Maneth

National ICT Australia Ltd. & UNSW, Sydney

Joint work w. Helmut Seidl (TU Munich)

January 20th, 2007 PLAN-X, Nice

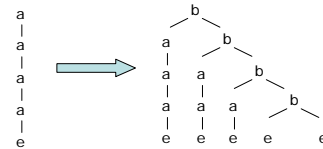
Prologue Tree Transducers

= (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]

$q_0(a(x)) \rightarrow b(q(x), q_0(x))$
 $q_0(e) \rightarrow e$
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$



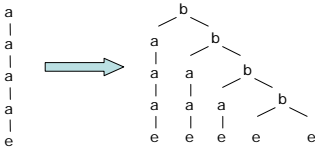
Prologue Tree Transducers

= (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]

$q_0(a(x)) \rightarrow b(q(x), q_0(x))$ $p_0(a(x)) \rightarrow p(x)$
 $q_0(e) \rightarrow e$ $p_0(e) \rightarrow e$
 $q(a(x)) \rightarrow a(q(x))$ $p(a(x)) \rightarrow b(a(q(x)), p(x))$
 $q(e) \rightarrow e$ $p(e) \rightarrow b(e, e)$



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]

$q_0(a(x)) \rightarrow b(q(x), q_0(x))$ $p_0(a(x)) \rightarrow p(x)$
 $q_0(e) \rightarrow e$ $p_0(e) \rightarrow e$
 $q(a(x)) \rightarrow a(q(x))$ $p(a(x)) \rightarrow b(a(q(x)), p(x))$
 $q(e) \rightarrow e$ $p(e) \rightarrow b(e, e)$

M1 is equivalent to M2

Transducers T1, T2 are **equivalent** iff \forall input s: $T_1(s) = T_2(s)$.

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.

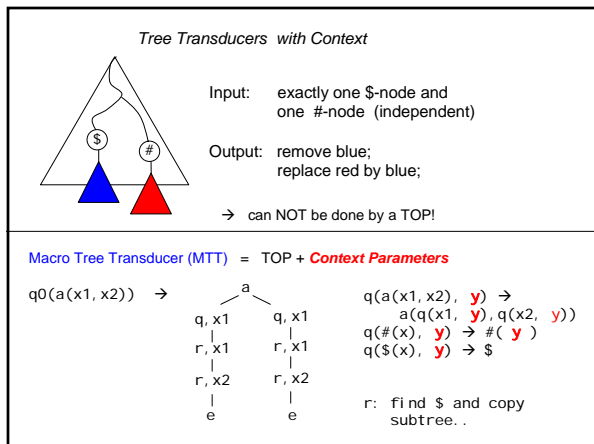
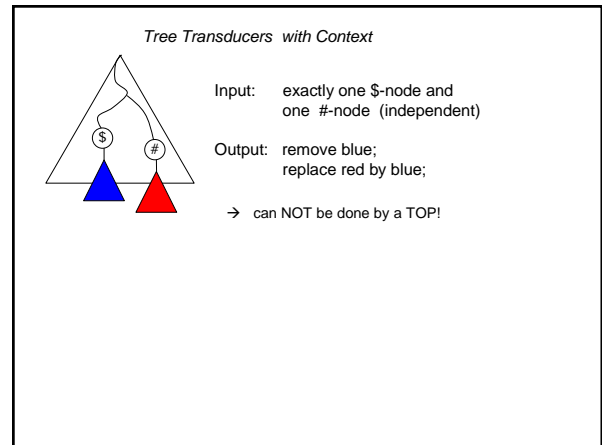
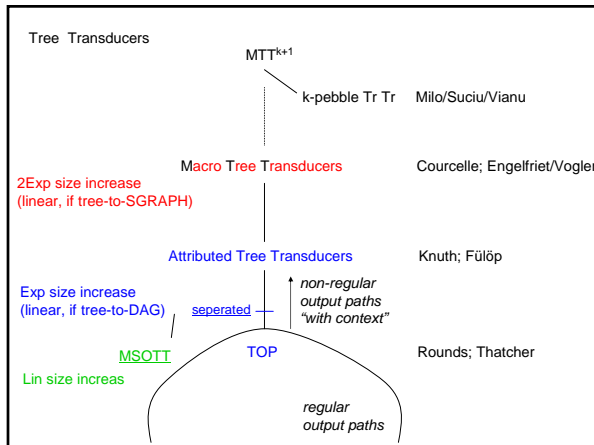
If M is total, then **Uniform&Earliest**(M) obtained in PTIME.

Outline

- Equivalence Problems & 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** [Griffiths88]
 - (→ 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-Zanetacci82]
 - 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)



Macro Tree Transducer (MTT) [Engelfriet/Vogler1985]

- FPs on trees, with **parameters**, pattern matching as ONLY operation

```

function q(s: itree, y: otree): otree
{
  case s=a(x) → return q(x, q(x, y))
  case s=e → return b(y, y)
}
  
```

- can simulate attribute grammars (seen as tree translations)
- always terminate (no circularities, strictly descent input tree)
- even compositions (!) can be computed in [Maneth03]
 - time $O(\text{size}(\text{input tree}) + \text{size}(\text{output tree}))$
 - "static garbage collection" ©
- Linear size incr. **decidable** = **MSO transducers** (→ **decidable equivalence**)

Macro Tree Transducer (MTT) [Engelfriet/Vogler1985]

→ FPs on trees, with **parameters**, pattern matching as ONLY operation

```

function q(s: i tree, y: otree): otree
{
  case s=a(x) → return q(x, q(x, y))
  case s=e → return b(y, y)
}

```

→ can simulate attribute grammars (seen as tree translations)

→ always terminate (no circularities, strictly descent input tree)

→ even compositions (!) can be computed in [Maneth02]
time $O(\text{size}(\text{input tree}) + \text{size}(\text{output tree}))$
"static garbage collection" ©

→ Linear size incr. *decidable* = **MSO transducers** (→ *decidable equivalence*)

But, unfortunately

OPEN whether **deterministic MTTs** have **decidable equivalence**.

Macro Tree Transducer (MTT) [Engelfriet/Vogler1985]

OPEN

Even for *monadic output* MTTs (= top-down **tree-to-string** transducers)

$$\begin{aligned} q(a(x), y) &\rightarrow a(q(x), q(x, a(y))) \\ q(e, y) &\rightarrow y \end{aligned}$$

$$\begin{aligned} p(a(x)) &\rightarrow p(x) a a p(x) \\ p(e) &\rightarrow \lambda \end{aligned}$$

$$\begin{aligned} q(a(x)) &\rightarrow a q(x) q(x) a \\ q(e) &\rightarrow \lambda \end{aligned}$$

Equivalent?!

OPEN for 30 years... ©

But, unfortunately

OPEN whether **deterministic MTTs** have **decidable equivalence**.

Uniform and Earliest TOPs

→ patterns (trees w. holes) $P_{\Sigma} = T_{\Sigma, \cup\{T\}}$

→ substitution $t[t1, t2, t3] = b(a(t1), b(t2, t3))$

→ pattern order $\sqcup \sqsubseteq p$ for all $p \in P_{\Sigma}$
 $p \sqsubseteq q$ if $\exists p1, \dots, pk: p = q[p1, \dots, pk]$

For example, $t \sqsubseteq b(T, T)$
 $\sqsubseteq b(a(T), T)$
 $\sqsubseteq b(T, b(T, T))$

$\sqcup \{p1, \dots, pk\}$ unique least upper bound (=nodes appearing in all pi)

Uniform and Earliest TOPs

Deterministic TOP $T = (Q, \Sigma, \Delta, \delta, A)$

Uniform = "if a state *blocks* at a certain input node, then ALL states translating that node *block*."

$B \subseteq Q$ is *consistent*, if $\bigcap_{q \in B} \text{dom}(q) \neq \emptyset$

Change $q \in Q$ to states $\langle q, B \rangle$ such that $\langle q, B \rangle, a$ -rule is defined iff for all $q' \in B$, the $\langle q', B \rangle, a$ -rule is defined.

All states that translate the current input node

Lemma
Uniform T' equivalent to T can be constructed in exponential time.

Uniform and Earliest TOPs

Deterministic TOP $T = (Q, \Sigma, \Delta, \delta, A)$

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

Common Prefix Pattern

$\text{pref}(q) := \sqcup \{ [[q]](s) \mid s \in T_{\Sigma} \text{ and } [[q]](s) \text{ defined} \}$

$$\begin{aligned} p0(a(x)) &\rightarrow p(x) \\ p0(e) &\rightarrow e \\ p(a(x)) &\rightarrow b(a(q(x)), p(x)) \\ p(e) &\rightarrow b(e, e) \\ q(a(x)) &\rightarrow a(q(x)) \\ q(e) &\rightarrow e \end{aligned}$$

$$\begin{aligned} \text{pref}(p0) &= T \\ \text{pref}(p) &= b(T, T) \end{aligned}$$

Uniform and Earliest TOPs

Deterministic TOP $T = (Q, \Sigma, \Delta, \delta, A)$

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

Common Prefix Pattern

$\text{pref}(q) := \sqcup \{ [[q]](s) \mid s \in T_{\Sigma} \text{ and } [[q]](s) \text{ defined} \}$

$$\begin{aligned} p0(a(x)) &\rightarrow p(x) \\ p0(e) &\rightarrow e \\ p(a(x)) &\rightarrow b(a(q(x)), p(x)) \\ p(e) &\rightarrow b(e, e) \\ q(a(x)) &\rightarrow a(q(x)) \\ q(e) &\rightarrow e \end{aligned}$$

$$\begin{aligned} \text{pref}(p0) &= T \\ \text{pref}(p) &= b(T, T) \end{aligned}$$

Definition

A uniform det. TOP is **earliest** if $\text{pref}(q) = T$ for all states q .

Uniform and Earliest TOPs

Deterministic TOP $T = (Q, \Sigma, \Delta, \delta, A)$

Theorem

$\text{pref}(q)$ can be computed in time $O(|T| \cdot \eta(T)^2)$

$\eta(T)$ = maximal size of a minimal output tree produced by any state.

Proof fixpoint iteration.

At most $\eta(T)$ iterations needed.

In each iteration, at most $|T|$ variables are updated, and each update takes at most time $O(\eta(T))$.

Theorem

T_1, T_2 earliest TOPs.

T_1 is **equivalent** to T_2 iff their rules are **equal up to state renaming**.

Uniform and Earliest TOPs

$p_0(a(x)) \rightarrow p(x)$
 $p_0(e) \rightarrow e$
 $p(a(x)) \rightarrow b(a(q(x)), p(x))$
 $p(e) \rightarrow b(e, e)$
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

M is uniform (because it is total).

→ Make it earliest

$\text{pref}(p) = b(T, T)$

In all right-hand sides, change $p(x)$ into $b(<p, 1>(x), <p, 2>(x))$

Uniform and Earliest TOPs

$p_0(a(x)) \rightarrow p(x) \quad b(<p, 1>(x), <p, 2>(x))$
 $p_0(e) \rightarrow e$
 ~~$p(a(x)) \rightarrow b(a(q(x)), p(x))$~~
 ~~$p(e) \rightarrow b(e, e)$~~
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

M is uniform (because it is total).

→ Make it earliest

$\text{pref}(p) = b(T, T)$

In all right-hand sides, change $p(x)$ into $b(<p, 1>(x), <p, 2>(x))$

Uniform and Earliest TOPs

$p_0(a(x)) \rightarrow p(x) \quad b(<p, 1>(x), <p, 2>(x))$
 $p_0(e) \rightarrow e$
 ~~$p(a(x)) \rightarrow b(a(q(x)), p(x))$~~
 ~~$p(e) \rightarrow b(e, e)$~~
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

M is uniform (because it is total).

→ Make it earliest

$\text{pref}(p) = b(T, T)$

In all right-hand sides, change $p(x)$ into $b(<p, 1>(x), <p, 2>(x))$

$<p, 1>(a(x)) \rightarrow a(q(x))$
 $<p, 1>(e) \rightarrow e$
 $<p, 2>(a(x)) \rightarrow b(<p, 1>(x), <p, 2>(x))$
 $<p, 2>(e) \rightarrow e$

Uniform and Earliest TOPs

$p_0(a(x)) \rightarrow p(x) \quad b(<p, 1>(x), <p, 2>(x))$
 $p_0(e) \rightarrow e$
 ~~$p(a(x)) \rightarrow b(a(q(x)), p(x))$~~
 ~~$p(e) \rightarrow b(e, e)$~~
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

$<p, 1>(a(x)) \rightarrow a(q(x))$
 $<p, 1>(e) \rightarrow e$
 $<p, 2>(a(x)) \rightarrow b(<p, 1>(x), <p, 2>(x))$
 $<p, 2>(e) \rightarrow e$

Uniform and Earliest TOPs

$p_0(a(x)) \rightarrow p(x) \quad b(<p, 1>(x), <p, 2>(x))$
 $p_0(e) \rightarrow e$
 ~~$p(a(x)) \rightarrow b(a(q(x)), p(x))$~~
 ~~$p(e) \rightarrow b(e, e)$~~
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

$<p, 1>(a(x)) \rightarrow a(q(x))$
 $<p, 1>(e) \rightarrow e$
 $<p, 2>(a(x)) \rightarrow b(<p, 1>(x), <p, 2>(x))$
 $<p, 2>(e) \rightarrow e$

$<p, 1>$ state-equivalent to q
 $<p, 2>$ state-equivalent to p_0

Uniform and Earliest TOPs

$p_0(a(x)) \rightarrow p(x)$ $b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$
 $p_0(e) \rightarrow e$
 ~~$p(a(x)) \rightarrow b(a(q(x)), p(x))$~~
 ~~$p(e) \rightarrow b(e, e)$~~
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

~~$\langle p, 1 \rangle(a(x)) \rightarrow a(q(x))$~~
 ~~$\langle p, 1 \rangle(e) \rightarrow e$~~
 ~~$\langle p, 2 \rangle(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$~~
 ~~$\langle p, 2 \rangle(e) \rightarrow e$~~

$\langle p, 1 \rangle$ state-equivalent to q
 $\langle p, 2 \rangle$ state-equivalent to p_0

Uniform and Earliest TOPs

$p_0(a(x)) \rightarrow p(x)$ $b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$
 $p_0(e) \rightarrow e$
 ~~$p(a(x)) \rightarrow b(a(q(x)), p(x))$~~
 ~~$p(e) \rightarrow b(e, e)$~~
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

~~$\langle p, 1 \rangle(a(x)) \rightarrow a(q(x))$~~
 ~~$\langle p, 1 \rangle(e) \rightarrow e$~~
 ~~$\langle p, 2 \rangle(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$~~
 ~~$\langle p, 2 \rangle(e) \rightarrow e$~~

$\langle p, 1 \rangle$ state-equivalent to q
 $\langle p, 2 \rangle$ state-equivalent to p_0

Uniform&Earliest(T1)
w. p_0 renamed to q_0

equals

T2

$q_0(a(x)) \rightarrow b(q(x), q_0(x))$
 $q_0(e) \rightarrow e$
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

Equivalence of TOPs with regular look-ahead

$M = (Q, \Sigma, \Delta, \delta, A, B)$

B det. bottom-up tree automaton
With $L(p_1) \cap L(p_2) = \emptyset$ for all states p_1, p_2

$q(a(x_1, x_2)) \rightarrow t \langle p_1, p_2 \rangle$

Given TOPs w la M_1, M_2 :

Change input symbol a into $\langle a, (p_1, p_2), (u_1, u_2) \rangle$
Then M_1, M_2 become ordinary TOPs (without lookahead)

Now, change M_1, M_2 so that they check if input tree is a correct relabeling wrt the automata B_1, B_2 .

Finally, make them uniform and earliest and check isomorphism as before.

Applications

\rightarrow XML query optimization

Assume result to query Q_1 is already materialized.

Given a new query Q_2 , check if Q_2 equivalent to Q_1 .
If so, return materialized result, instead of recomputing.

Possible extension

Q_1 "subsumes" Q_2 , if for all inputs s , $Q_2(s)$ can be obtained by deleting subtrees from $Q_1(s)$.

Conjecture
Given uniform and earliest Q_1, Q_2 , it is decidable whether or not Q_1 subsumes Q_2 .

Given a new query Q_2 , check if Q_1 subsumes Q_2 .
If so, return materialized result, with appropriate subtrees removed.

Applications

\rightarrow XML query optimization

Assume result to query Q_1 is already materialized.

Given a new query Q_2 , check if Q_2 equivalent to Q_1 .
If so, return materialized result, instead of recomputing.

Nicer....

Q_1 "subsumes" Q_2 , if there exists a TOP Q_3 such that for all inputs s , $Q_2(s) = Q_3(Q_1(s))$

Decidable?
Seems difficult...

Applications

\rightarrow XML query optimization

Assume result to query Q_1 is already materialized.

Given a new query Q_2 , check if Q_2 equivalent to Q_1 .
If so, return materialized result, instead of recomputing.

Nicer....

Q_1 "subsumes" Q_2 , if there exists a TOP Q_3 such that for all inputs s , $Q_2(s) = Q_3(Q_1(s))$

Decidable?
Seems difficult...

THE END