

*MSO Logic and Tree Transducers with
Decidable Equivalence*

Sebastian Maneth

National ICT Australia Ltd. & UNSW, Sydney

Joint work w. [Joost Engelfriet](#) (Univ. Leiden)
[Helmut Seidl](#) (TU Munich)

January 24th, 2007

Univ. of Edinburgh

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]

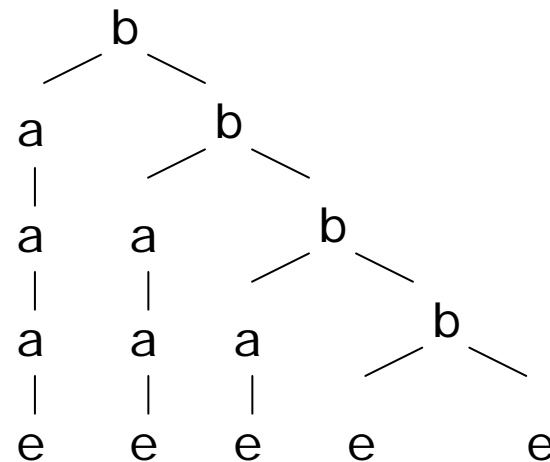
$q_0(a(x)) \rightarrow b(q(x), q_0(x))$

$q_0(e) \rightarrow e$

$q(a(x)) \rightarrow a(q(x))$

$q(e) \rightarrow e$

a
|
a
|
a
|
a
|
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]

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

$q_0(e) \rightarrow e$

$q(a(x)) \rightarrow a(q(x))$

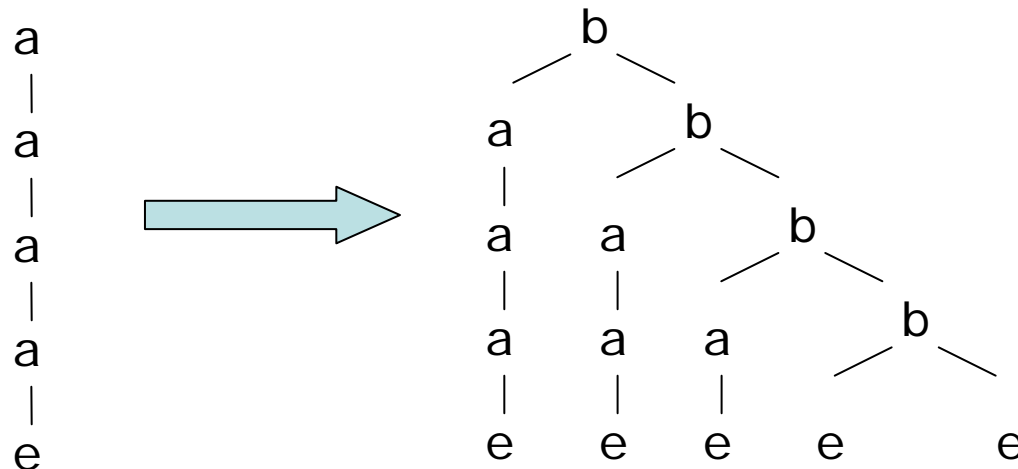
$q(e) \rightarrow e$

$p_0(a(x)) \rightarrow p(x)$

$p_0(e) \rightarrow e$

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

$p(e) \rightarrow b(e, e)$



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]

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

$$q_0(e) \rightarrow e$$

$$q(a(x)) \rightarrow a(q(x))$$

$$q(e) \rightarrow e$$

$$p_0(a(x)) \rightarrow p(x)$$

$$p_0(e) \rightarrow e$$

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

$$p(e) \rightarrow b(e, e)$$

M1

is equivalent to

M2

Transducers **M1**, **M2** are **equivalent** iff \forall input s : **M1**(s) = **M2**(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 [Plan-X 2007, with Helmut Seidl]

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.

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 synthsized, over Δ_{syn}
 - *MSO definable tree transducers* **decidable**
(→ use Parikh property) [Engelfriet/Maneth05]

Tree Transducers

MTT^{k+1}

k-pebble Tr Tr

Milo/Suciu/Vianu

Macro Tree Transducers

Courcelle; Engelfriet/Vogler

2Exp size increase
(linear, if tree-to-SGRAPH)

Attributed Tree Transducers

Knuth; Fülöp

Exp size increase
(linear, if tree-to-DAG)

seperated

↑ *non-regular
output paths
"with context"*

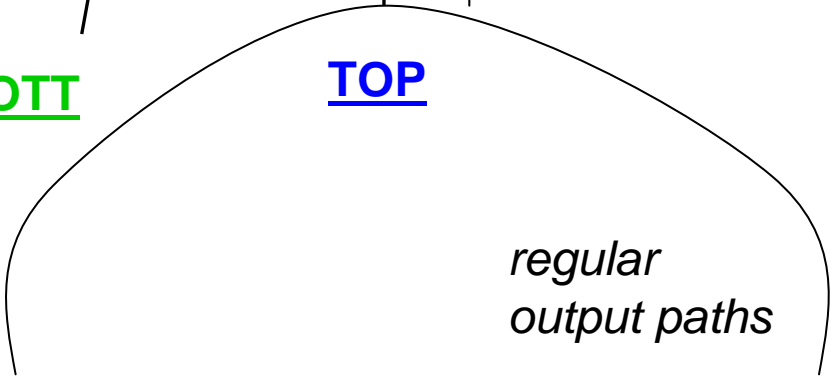
MSOTT

TOP

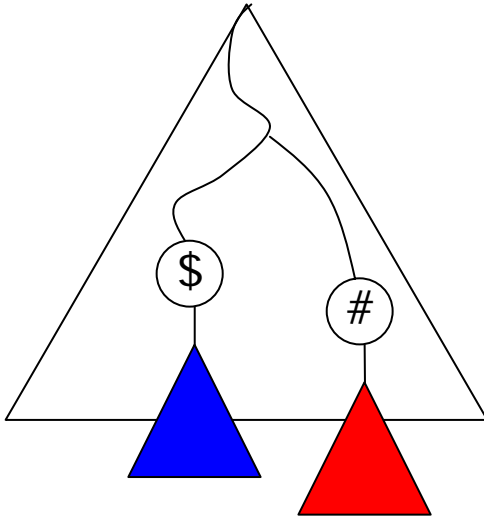
Rounds; Thatcher

Lin size increas

*regular
output paths*



Tree Transducers with Context

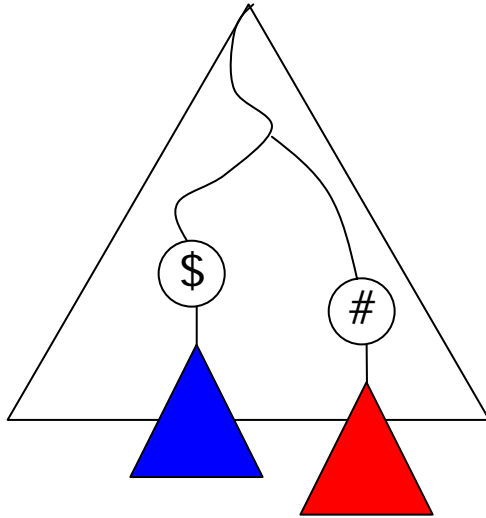


Input: exactly one \$-node and
one #-node (independent)

Output: remove blue;
replace red by blue;

→ can NOT be done by a TOP!

Tree Transducers with Context



Input: exactly one \$-node and one #-node (independent)

Output: remove blue;
replace red by blue;

→ can NOT be done by a TOP!

Macro Tree Transducer (MTT) = TOP + **Context Parameters**

$q_0(a(x_1, x_2)) \rightarrow$

```

      a
     / \
  q, x1 q, x1
   |   |   |
  r, x1 r, x1
   |   |   |
  r, x2 r, x2
   |   |   |
   e   e   e
  
```

$q(a(x_1, x_2), y) \rightarrow$
 $a(q(x_1, y), q(x_2, y))$
 $q(\#(x), y) \rightarrow \#(y)$
 $q(\$(x), y) \rightarrow \$$

r: find \$ and copy subtree..

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 [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)

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

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

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

Equivalent?!
OPEN for 30 years... ☹

But, unfortunately

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

Outline

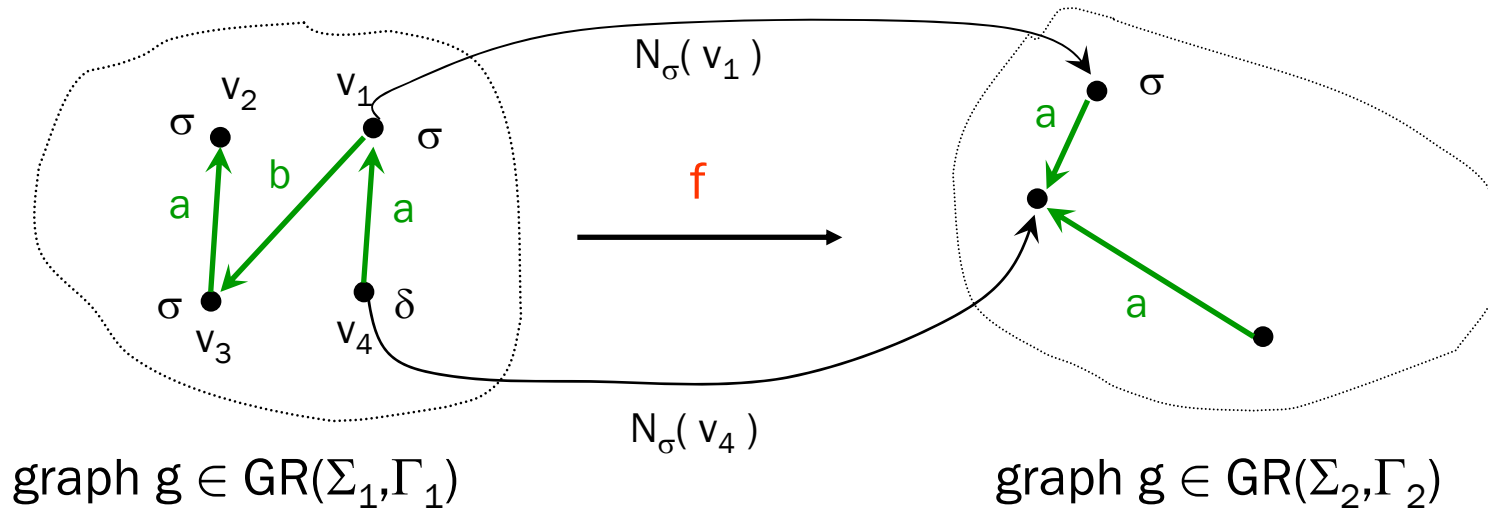
1. MSO Definable Translations
 2. Parikh's Theorem and Semilinear Sets
 3. Deciding Equivalence of MSOTT
 4. Applications
- If time permits:
5. Uniform & Earliest TOPs
 6. Deciding Equivalence of TOPs

1. MSO Definable Translations (det.)

MSO graph translation $f : GR(\Sigma_1, \Gamma_1) \rightarrow GR(\Sigma_2, \Gamma_2)$

dom: domain formula	$L(\text{dom}) = \text{dom}(f)$
$N_\sigma(x)$: node formulas	for every $\sigma \in \Sigma_2$
$E_\gamma(x, y)$: edge formulas	for every $\gamma \in \Gamma_2$

→ Define output graph, in terms of input graph.
Interpretation of one logical structure in another. (FMT: “reductions”)



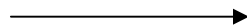
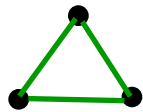
1. MSO Definable Translations (det.)

Example: edge complement f_1

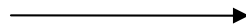
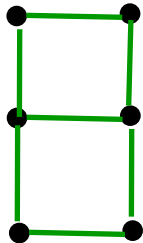
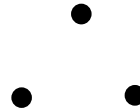
$\text{dom} \quad \ulcorner \text{true}$

$N_\sigma(x) \quad \ulcorner \text{lab}_\sigma(x) \text{ for all } \sigma$

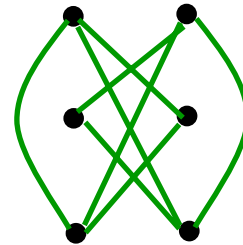
$E(x, y) \quad \ulcorner \neg \text{edg}(x, y)$



f_1



f_1



1. MSO Definable Translations (det.)

MSO graph translation $f : \text{GR}(\Sigma_1, \Gamma_1) \rightarrow \text{GR}(\Sigma_2, \Gamma_2)$

dom: domain formula $L(\text{dom}) = \text{dom}(f)$

$N_\sigma(x)$: node formulas for every $\sigma \in \Sigma_2$

$E_\gamma(x, y)$: edge formulas for every $\gamma \in \Gamma_2$

Input graph $g = (V, E)$.

output graph $h = f(g) = (U, F)$

$U = \{ u \in V \mid \text{there is a unique } \sigma \in \Sigma_2 \text{ such that } (g, u) \models N_\sigma(x) \}$

$F = \{ (u, \gamma, v) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_\gamma(x, y) \}$

$\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$

1. MSO Definable Translations (det.)

MSO graph translation $f : \text{GR}(\Sigma_1, \Gamma_1) \rightarrow \text{GR}(\Sigma_2, \Gamma_2)$

dom: domain formula $L(\text{dom}) = \text{dom}(f)$

$N_\sigma(x)$: node formulas for every $\sigma \in \Sigma_2$

$E_\gamma(x, y)$: edge formulas for every $\gamma \in \Gamma_2$

Input graph $g = (V, E)$.

output graph $h = f(g) = (U, F)$

$\rightarrow \text{size}(f(g)) = |U|$

is $\leq \text{size}(g) = |V|$

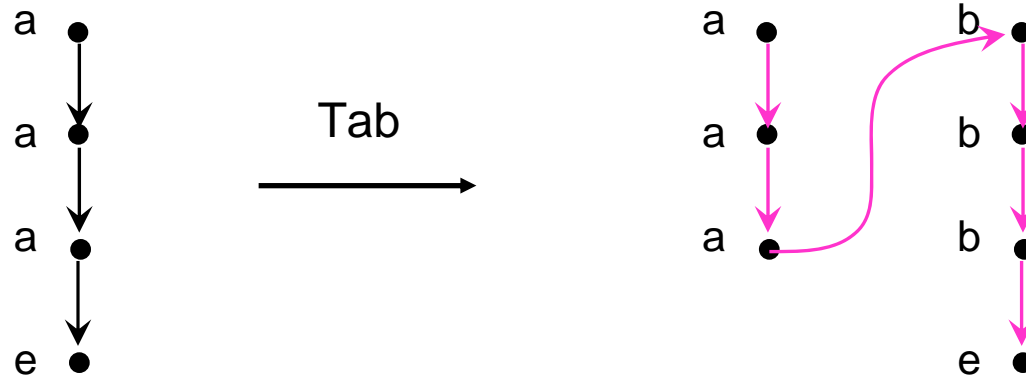
$U = \{ u \in V \mid \text{there is a unique } \sigma \in \Sigma_2 \text{ such that } (g, u) \models N_\sigma(x) \}$

$F = \{ (u, \gamma, v) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_\gamma(x, y) \}$

$\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$

1. MSO Definable Translations

MSO *string-to-string* transducer Tab



$$C = \{ A, B \} \quad \text{dom} \quad \text{string} \wedge (\forall x)(\text{lab-e}(x) \leftrightarrow \text{last}(x))$$

$$N_{a,A}(x) \quad \text{lab-a}(x) \qquad N_{b,B}(x) \quad \text{lab-a}(x)$$

$$N_{e,A}(x) \quad \text{false} \qquad N_{e,B}(x) \quad \text{lab-e}(x)$$

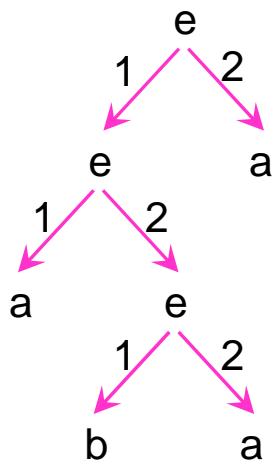
< all others are formulae=false >

$$E_{A,A}(x, y) \quad E_{B,B}(x, y) \quad \text{edge}(x, y)$$

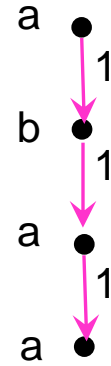
$$E_{A,B}(x, y) \quad (\exists z)(\text{edge}(x, z) \wedge \text{lab-e}(z)) \wedge \neg(\exists z) \text{edge}(z, y)$$

1. MSO Definable Translations

MSO *tree-to-string* transducer T_{yield}



T_{yield}



$\text{dom } \ulcorner \text{true}$

< all others are set to false..! >

$N_a(x) \ulcorner N_b(x) \ulcorner \text{lab-a}(x)$

$E_1(x, y) \ulcorner \text{leaf}(x) \wedge \text{leaf}(y) \wedge$
 $(\exists u)(\exists v)(\exists w) \text{ edge-2}^*(v, x) \wedge \text{edge-1}(u, v)$
 $\wedge \text{edge-2}(u, w) \wedge \text{edge-1}^*(w, y)$

MSO Definable Translations

Example **T_{yield}**

Translate a tree
into
its yield/frontier

Corollary

Class of **string languages**
generated by MSOT's
includes the CF languages

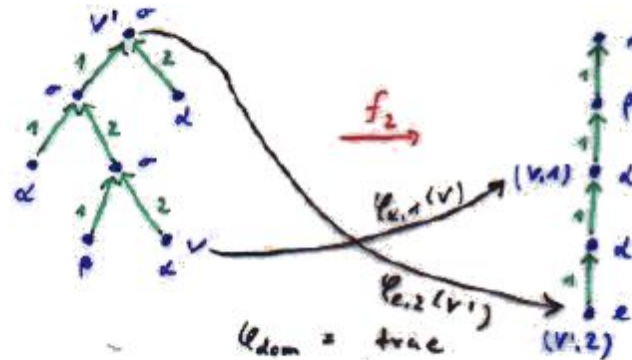
→ Strict superset
Generate

$$\{ a^n b^n c^n d^n \mid n \geq 0 \}$$

Example: **yield of a tree, f_2**

trees over the ranked alphabet

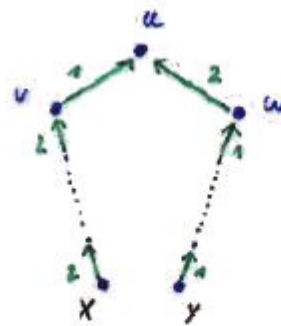
$$\Delta = \{ \sigma^{(2)}, \alpha^{(0)}, \beta^{(0)} \}$$



$$\phi_{\alpha}(x) = \text{lab}_{\alpha}(x), \phi_{\beta}(x) = \text{lab}_{\beta}(x)$$

$$\begin{aligned} \phi_{\sigma}(x,y) = & \text{edge}_2(x,v)^* \wedge \text{edge}_1(v,u) \wedge \\ \exists u,v,w: & \text{edge}_2(w,u) \wedge \text{edge}_1(y,w)^* \wedge \\ & \text{leaf}(x) \wedge \text{leaf}(y) \end{aligned}$$

$\text{next}(x,y)$



i.e., paths of form $2^* 1 2^*$

FACTS

DMSOTT

- Closed under composition
- inverses preserve REGT

$$\text{DMSOTT(REGT)} \\ = \text{MSOTT(REGT)}$$

“most beautiful class
of tree languages! 😊



- Closed under MSOTT
- path languages(MSOTT(REGT))
= string languages(MSOTT(REGT))
= MSOTS(REGT)
- are **Parikh**

FACTS

DMSOTT

- *Closed under composition*
- inverses preserve REGT

$$\text{DMSOTT(REGT)} \\ = \text{MSOTT(REGT)}$$



“most beautiful class
of tree languages! 😊

- *Closed under MSOTT*
- path languages(MSOTT(REGT))
= string languages(MSOTT(REGT))
= MSOTS(REGT)
- are *Parikh*

2. Parikh's Theorem and Semilinear Sets

Languages in MSOTT(REGT)

→ are **Parikh**

Let $\Sigma = \{ \sigma_1, \dots, \sigma_k \}$.

For a discrete graph / string / tree g over Σ , define

$$\text{Par}(g) = (n_1, \dots, n_k) \in \mathbf{N}^k$$

such that n_i is the number of σ_i 's in g .

A set $P \subseteq \mathbf{N}^k$ is **semilinear** if there exists

a **regular language** R such that $P = \text{Par}(R)$.

A set S (of graphs/stings) is **Parikh**, if $\text{Par}(S)$ is semilinear.

2. Parikh's Theorem and Semilinear Sets

Languages in MSOTT(REGT)

→ are **Parikh**

Let $\Sigma = \{ \sigma_1, \dots, \sigma_k \}$.

For a discrete graph / string / tree g over Σ , define

$$\text{Par}(g) = (n_1, \dots, n_k) \in \mathbf{N}^k$$

such that n_i is the number of σ_i 's in g .

A set $P \subseteq \mathbf{N}^k$ is **semilinear** if there exists

a **regular language** R such that $P = \text{Par}(R)$.

A set S (of graphs/stings) is **Parikh**, if $\text{Par}(S)$ is semilinear.

PARIKH'S THEOREM

Every context-free language is **Parikh**.

2. Parikh's Theorem and Semilinear Sets

Context-free languages are **Parikh**

$$L1 = \{ a^n b^n \mid n \geq 1 \}$$

2. Parikh's Theorem and Semilinear Sets

Context-free languages are **Parikh**

$$L1 = \{ a^n b^n \mid n \geq 1 \}$$

$$\text{Par}(L1) = \{ (n, n) \mid n \geq 1 \}$$

2. Parikh's Theorem and Semilinear Sets

Context-free languages are **Parikh**

$$L1 = \{ a^n b^n \mid n \geq 1 \}$$

$$\text{Par}(L1) = \{ (n, n) \mid n \geq 1 \} = \text{Par}((ab)^*)$$

2. Parikh's Theorem and Semilinear Sets

Context-free languages are **Parikh**

$$L1 = \{ a^n b^n \mid n \geq 1 \}$$

$$\text{Par}(L1) = \{ (n, n) \mid n \geq 1 \} = \text{Par}((ab)^*)$$

$$L2 = \{ w w \mid w \in \{ a, b \}^* \}$$

$$\text{Par}(L2) =$$

2. Parikh's Theorem and Semilinear Sets

Context-free languages are **Parikh**

$$L1 = \{ a^n b^n \mid n \geq 1 \}$$

$$\text{Par}(L1) = \{ (n, n) \mid n \geq 1 \} = \text{Par}((ab)^*)$$

$$L2 = \{ w w \mid w \in \{ a, b \}^* \}$$

$$\text{Par}(L2) = \text{Par}((aa)^*(bb)^*)$$

$$L3 = \{ w \in \{ a, b \}^* \mid \#a(w) = \#b(w) \}$$

→ Is $L3$ context-free??

2. Parikh's Theorem and Semilinear Sets

Context-free languages are **Parikh**

$$L1 = \{ a^n b^n \mid n \geq 1 \}$$

$$\text{Par}(L1) = \{ (n, n) \mid n \geq 1 \} = \text{Par}((ab)^*)$$

$$L2 = \{ w w \mid w \in \{ a, b \}^* \}$$

$$\text{Par}(L2) = \text{Par}((aa)^*(bb)^*)$$

$$L3 = \{ w \in \{ a, b \}^* \mid \#a(w) = \#b(w) \}$$

→ Is L3 context-free?? SURE! ...even deterministic...

And, again, $\text{Par}(L3) = \text{Par}(L1)$

2. Parikh's Theorem and Semilinear Sets

Languages in MSOTT(REGT)

→ are **Parikh**

Let $\Sigma = \{ \sigma_1, \dots, \sigma_k \}$.

For a discrete graph / string / tree g over Σ , define

$$\text{Par}(g) = (n_1, \dots, n_k) \in \mathbf{N}^k$$

such that n_i is the number of σ_i 's in g .

A set $P \subseteq \mathbf{N}^k$ is **semilinear** if there exists

a *regular language* R such that $P = \text{Par}(R)$.

A set S (of graphs/stings) is **Parikh**, if $\text{Par}(S)$ is semilinear.

PARIKH'S THEOREM

Every context-free language is **Parikh**.

Languages in $\text{MSOTT}(\text{REGT}) = \text{ATTsur}^R(\text{REGT})$

→ are **Parikh**

Let $\Sigma = \{ \sigma_1, \dots, \sigma_k \}$.

For a discrete graph / string / tree g over Σ , define

$$\text{Par}(g) = (n_1, \dots, n_k) \in \mathbf{N}^k$$

such that n_i is the number of σ_i 's in g .

A set $P \subseteq \mathbf{N}^k$ is **semilinear** if there exists
a *regular language* R such that $P = \text{Par}(R)$.

A set S (of graphs/stings) is **Parikh**, if $\text{Par}(S)$ is semilinear.

put all output symbols in the rules for a production into a production of a *new cf grammar*.

PARIKH'S THEOREM

Every context-free language is Parikh.

Languages L in $MSOTT(REGT) = ATTsur^R(REGT)$

→ are **Parikh**

Par(L) is semilinear (i.e., = **Par**(R) for some reg.l. R)

Lemma It is decidable for a **semilinear set** $S \subseteq \mathbf{N}^2$,
whether there exists $n \in \mathbf{N}$ such that $(n,n) \in S$.

Proof. Let $P = \{ (n,n) \mid n \in \mathbf{N} \} = \mathbf{Par}((ab)^*)$

Then $S \cap P$ is semilinear, and semilinear sets have
decidable emptiness. [[GinsburgSpanier64](#), difficult!]

Alternative Proof??

Languages L in $MSOTT(REGT) = ATTsur^R(REGT)$
→ are **Parikh**

Par(L) is semilinear (i.e., = **Par**(R) for some reg.l. R)

Lemma It is decidable for a **semilinear set** $S \subseteq \mathbf{N}^2$,
whether there exists $n \in \mathbf{N}$ such that $(n,n) \in S$.

Alternative Proof

$$L3 = \{ w \in \{ a, b \}^* \mid \#a(w) = \#b(w) \}$$

Let R be reg. language s.t. **Par**(S) = R .

Then $R \cap L3$ is context-free (“triple-construction”)
and cf. languages have decidable emptiness.

3.

THEOREM

The Equivalence problem for **DMSOTT** is **decidable**.

3.

Let's first do string output

THEOREM

The Equivalence problem for **DMSOTS** is **decidable**.

3.

Let's first do string output

THEOREM

The Equivalence problem for **DMSOTS** is **decidable**.

M1, M2 deterministic MSO tree-to-string transducers

(1) Add an end marker \$ (gives dmsots's N1/N2)

$$N1(s) = M1(s) \$$$

$$N2(s) = M2(s) \$$$

(2) M1 is equiv. to M2

iff $\neg (\exists s \exists n \ N1(s) / n \neq N2(s) / n)$

3.

Let's first do string output

THEOREM

The Equivalence problem for **DMSOTS** is **decidable**.

M1, M2 deterministic MSO tree-to-string transducers

(1) Add an end marker \$ (gives dmsots's N1/N2)

$$N1(s) = M1(s) \$$$

$$N2(s) = M2(s) \$$$

(2) M1 is equiv. to M2

iff $\neg (\exists s \exists n \ N1(s) / n \neq N2(s) / n)$

iff $\neg (\exists a, b: a \neq b \wedge \exists s \exists n \ N1(s) / n = a \wedge N2(s) / n = b)$

3.

Let's first do string output

THEOREM

The Equivalence problem for **DMSOTS** is **decidable**.

M1, M2 deterministic MSO tree-to-string transducers

(1) Add an end marker \$ (gives dmsots's N1/N2)

$$N1(s) = M1(s) \$$$

$$N2(s) = M2(s) \$$$

(2) M1 is equiv. to M2 (gives $N1^a(s) = \{ a^n \mid N1(s) / n = a \}$
and $N2^b(s)$)

iff $\neg (\exists s \exists n \ N1(s) / n \neq N2(s) / n)$

iff $\neg (\exists a, b: a \neq b \wedge \exists s \exists n \ N1(s) / n = a \wedge N2(s) / n = b)$

iff $\neg (\dots \wedge \exists s \exists n \ a^n \in N1^a(s) \wedge b^n \in N2^b(s))$

3.

Let's first do string output

THEOREM

The Equivalence problem for **DMSOTS** is **decidable**.

M1, M2 deterministic MSO tree-to-string transducers

(1) Add an end marker \$ (gives dmsots's N1/N2)

$$N1(s) = M1(s) \$$$

$$N2(s) = M2(s) \$$$

(2) M1 is equiv. to M2 (gives $N12^{a,b}(s) = \{ a^n b^m \mid N1(s) / n = a, N2(s) / m = b \}$)

iff $\neg (\exists s \exists n \ N1(s) / n \neq N2(s) / n)$

iff $\neg (\exists a, b: a \neq b \wedge \exists s \exists n \ N1(s) / n = a \wedge N2(s) / n = b)$

iff $\neg (\dots \wedge \exists s \exists n \ a^n \in N1^a(s) \wedge b^n \in N2^b(s))$

iff $\neg (\dots \wedge \text{Par}(\text{Out}(N12^{a,b})) \cap \{(n, n) \mid n \geq 1\} \neq \emptyset)$

3.

Let's first do string output

THEOREM

The Equivalence problem for **DMSOTS** is **decidable**.

M1, M2 deterministic MSO tree-to-string transducers

(1) Add an end marker \$ (gives dmsots's N1/N2)

$$N1(s) = M1(s) \$$$

$$N2(s) = M2(s) \$$$

(2) M1 is equiv. to M2 (gives $N12^{a,b}(s) = \{ a^n b^m \mid N1(s) / n = a, N2(s) / m = b \}$)

iff $\neg (\exists a, b: a \neq b \wedge \text{Par}(\text{Out}(N12^{a,b})) \cap \{ (n, n) \mid n \geq 1 \} \neq \emptyset)$
 $\in \Delta \cup \{ \$ \}$

for finitely many a,b

(A) construct $N12^{a,b}$

(B) Construct 'Parikh-I' $R_{a,b}$ for $\text{Out}(N12_{a,b})$

(C) Check Intersection-Emtpiness w. **L3**

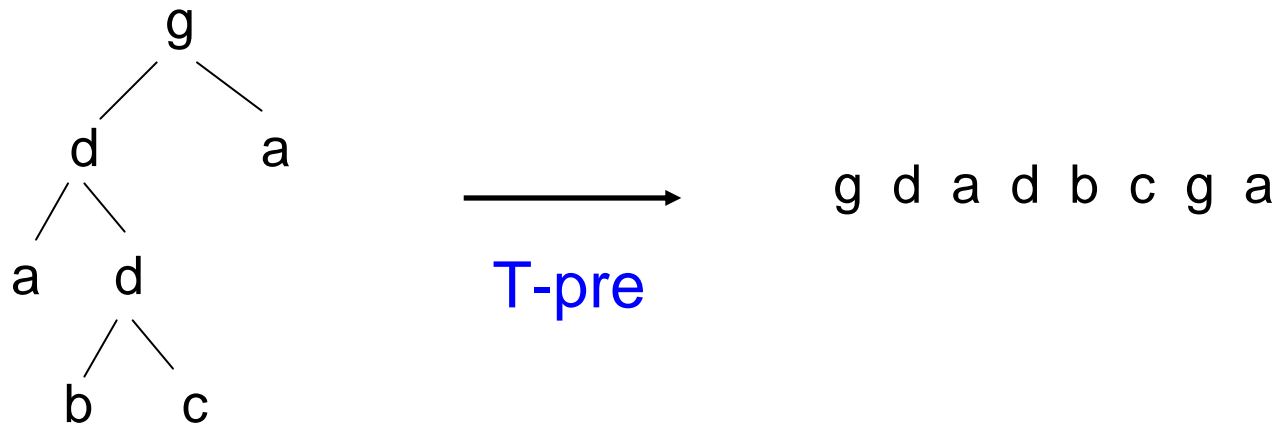
□

3.

Now, let's do **tree output**

THEOREM

The Equivalence problem for **DMSOTT** is **decidable**.



T-pre is in **DMSOTS**

(similar to Tyield)

3.

Now, let's do tree output

THEOREM

The Equivalence problem for **DMSOTT** is **decidable**.

DMSOTTs $M_1, M_2,$

Construct $N_1 = M_1 ; T\text{-pre}$

$N_2 = M_2 ; T\text{-pre}$

M_1 equiv. to M_2 iff N_1 equiv. to N_2

Decidable
(because N_1, N_2 are DMSOTS's)

□


Even works for MSO **graph**-to-string/tree transducers,
if the input restricted to a context-free graph language!

Because [\[Courcelle1994\]](#)

Images of cf. graph languages under
MSO **graph**-to-string/tree transductions are Parikh!

THEOREM

It is decidable for [det. MSO graph-to-string/tree](#) transducers whether
they are equivalent on a **cf set of graphs**.


= set of graphs of bounded tree-width!

4. Applications

DMSOTT appears naturally in the context of **XML query languages**:

(Structural) XQuery / XSLT \subseteq MTT* [Milo/Suciu/Vianu02]
+ [Engelfriet/M.03]

Restriction to queries of **linear size increase**:

(struct) XQuery / XSLT \cap LSI \subseteq DMSOTT [M. FSTTCS'03]

COROLLARY

**(struct) XML queries of linear size increase
have deducible equivalence.**

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

Conjecture

Given **DMSOTTs** 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.

Nicer....

Q1 “subsumes” Q2, if there exists a **DMSOTT Q3** such that
for all inputs s,

$$Q2(s) = Q3(\underbrace{Q1(s)}_{\text{“view”}})$$

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.

Nicer....

Q1 “subsumes” Q2, if there exists a **DMSOTT Q3** such that
for all inputs s,

$$Q2(s) = Q3(\underbrace{Q1(s)}_{\text{“view”}})$$

Decidable?

Seems difficult...

THE END?

Back to top-down tree transducers.

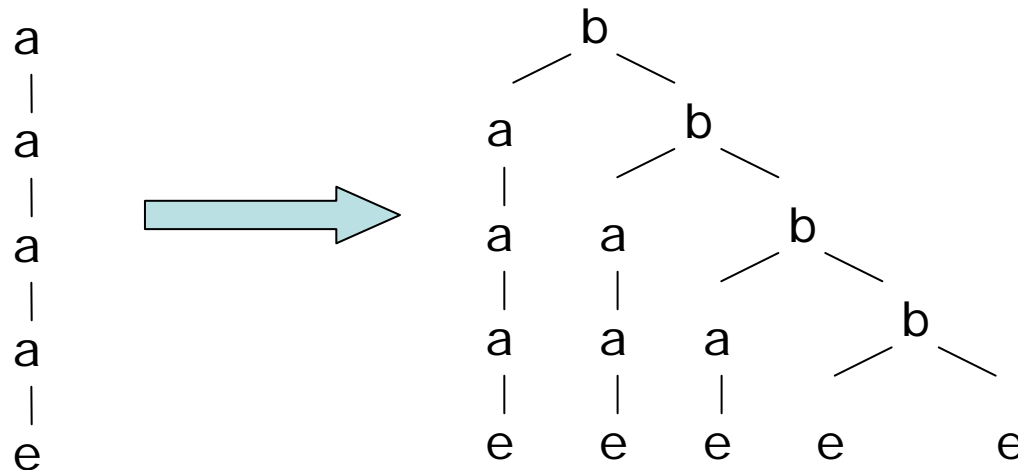
(decidable equivalence known from 80's / Complexity was open..)

E.g. \rightarrow **finite-state** (generalize FTA to input + output)

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

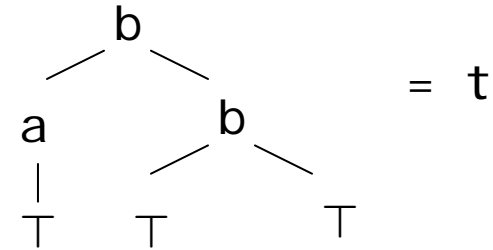
$q_0(a(x)) \rightarrow b(q(x), q_0(x))$
 $q_0(e) \rightarrow e$
 $q(a(x)) \rightarrow a(q(x))$
 $q(e) \rightarrow e$

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



5. Uniform and Earliest TOPs

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



→ substitution $t[t_1, t_2, t_3] = b(a(t_1), b(t_2, t_3))$

→ pattern order $\perp \sqsubseteq p$ for all $p \in P_\Sigma$
 $p \sqsubseteq q$ if $\exists p_1, \dots, p_k: p = q[p_1, \dots, p_k]$

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

$\sqcup \{ p_1, \dots, p_k \}$ unique least upper bound (=nodes appearing in all p_i)

5. Uniform and Earliest TOPs

“axiom” (start) tree

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.

5. Uniform and Earliest TOPs

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

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

Common Prefix Pattern

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

$$p0(a(x)) \rightarrow p(x)$$

$$p0(e) \rightarrow e$$

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

$$p(e) \rightarrow b(e, e)$$

$$q(a(x)) \rightarrow a(q(x))$$

$$q(e) \rightarrow e$$

$$\text{pref}(p0) = T$$

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

5. Uniform and Earliest TOPs

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

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

Common Prefix Pattern

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

$$p0(a(x)) \rightarrow p(x)$$

$$p0(e) \rightarrow e$$

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

$$p(e) \rightarrow b(e, e)$$

$$q(a(x)) \rightarrow a(q(x))$$

$$q(e) \rightarrow e$$

$$\text{pref}(p0) = T$$

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

Definition

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

6. Deciding Equivalence of 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**.

6. Deciding Equivalence of TOPs

$$\begin{aligned} p_0(a(x)) &\rightarrow p(x) \\ p_0(e) &\rightarrow e \\ p(a(x)) &\rightarrow b(a(q(x)), p(x)) \\ p(e) &\rightarrow b(e, e) \\ q(a(x)) &\rightarrow a(q(x)) \\ q(e) &\rightarrow e \end{aligned}$$

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 $b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$

6. Deciding Equivalence of TOPs

$$\begin{array}{l} p_0(a(x)) \rightarrow \cancel{p(x)} \quad b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ p_0(e) \rightarrow e \\ \cancel{p(a(x)) \rightarrow b(a(q(x)), p(x))} \\ \cancel{p(e) \rightarrow b(e, e)} \\ q(a(x)) \rightarrow a(q(x)) \\ q(e) \rightarrow e \end{array}$$

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 $b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$

6. Deciding Equivalence of TOPs

$$\begin{array}{l} p_0(a(x)) \rightarrow \cancel{p(x)} \quad b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ p_0(e) \rightarrow e \\ \cancel{p(a(x)) \rightarrow b(a(q(x)), p(x))} \\ \cancel{p(e) \rightarrow b(e, e)} \\ q(a(x)) \rightarrow a(q(x)) \\ q(e) \rightarrow e \end{array}$$

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 $b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$

$$\begin{array}{l} \langle p, 1 \rangle(a(x)) \rightarrow a(q(x)) \\ \langle p, 1 \rangle(e) \rightarrow e \\ \langle p, 2 \rangle(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ \langle p, 2 \rangle(e) \rightarrow e \end{array}$$

6. Deciding Equivalence of TOPs

$$\begin{array}{l} p_0(a(x)) \rightarrow \cancel{p(x)} \quad b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ p_0(e) \rightarrow e \\ \cancel{p(a(x)) \rightarrow b(a(q(x)), p(x))} \\ \cancel{p(e) \rightarrow b(e, e)} \\ q(a(x)) \rightarrow a(q(x)) \\ q(e) \rightarrow e \end{array}$$

$$\begin{array}{l} \langle p, 1 \rangle(a(x)) \rightarrow a(q(x)) \\ \langle p, 1 \rangle(e) \rightarrow e \\ \langle p, 2 \rangle(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ \langle p, 2 \rangle(e) \rightarrow e \end{array}$$

6. Deciding Equivalence of TOPs

$$\begin{array}{l} p_0(a(x)) \rightarrow \cancel{p(x)} \quad b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ p_0(e) \rightarrow e \\ \cancel{p(a(x)) \rightarrow b(a(q(x)), p(x))} \\ \cancel{p(e) \rightarrow b(e, e)} \\ q(a(x)) \rightarrow a(q(x)) \\ q(e) \rightarrow e \end{array}$$

$$\begin{array}{l} \langle p, 1 \rangle(a(x)) \rightarrow a(q(x)) \\ \langle p, 1 \rangle(e) \rightarrow e \\ \langle p, 2 \rangle(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ \langle p, 2 \rangle(e) \rightarrow e \end{array}$$

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

6. Deciding Equivalence of TOPs

$$\begin{array}{l}
 p_0(a(x)) \rightarrow \cancel{p(x)} \quad b(\overset{q}{\cancel{\langle p, 1 \rangle}}(x), \overset{p_0}{\cancel{\langle p, 2 \rangle}}(x)) \\
 p_0(e) \rightarrow e \\
 \cancel{p(a(x)) \rightarrow b(a(\overset{q}{q}(x)), \overset{p}{p}(x))} \\
 \cancel{p(e) \rightarrow b(e, e)} \\
 q(a(x)) \rightarrow a(q(x)) \\
 q(e) \rightarrow e
 \end{array}$$

$$\begin{array}{l}
 \cancel{\langle p, 1 \rangle(a(x)) \rightarrow a(q(x))} \\
 \cancel{\langle p, 1 \rangle(e) \rightarrow e} \\
 \cancel{\langle p, 2 \rangle(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))} \\
 \cancel{\langle p, 2 \rangle(e) \rightarrow e}
 \end{array}$$

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

6. Deciding Equivalence of TOPs

$$\begin{array}{l}
 p0(a(x)) \rightarrow \cancel{p(x)} \quad b(\overset{q}{\cancel{\langle p, 1 \rangle}}(x), \overset{p0}{\cancel{\langle p, 2 \rangle}}(x)) \\
 p0(e) \rightarrow e \\
 \cancel{p(a(x)) \rightarrow b(a(\overset{q}{q}(x)), \overset{p0}{p}(x))} \\
 \cancel{p(e) \rightarrow b(e, e)} \\
 q(a(x)) \rightarrow a(q(x)) \\
 q(e) \rightarrow e
 \end{array}$$

$$\begin{array}{l}
 \cancel{\langle p, 1 \rangle(a(x)) \rightarrow a(q(x))} \\
 \cancel{\langle p, 1 \rangle(e) \rightarrow e} \\
 \cancel{\langle p, 2 \rangle(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))} \\
 \cancel{\langle p, 2 \rangle(e) \rightarrow e}
 \end{array}$$

Uniform&Earliest(T1)
w. $p0$ renamed to $q0$

equals

T2

$\langle p, 1 \rangle$ state-equivalent to q
 $\langle p, 2 \rangle$ state-equivalent to $p0$

$$\begin{array}{l}
 q0(a(x)) \rightarrow b(q(x), q0(x)) \\
 q0(e) \rightarrow e \\
 q(a(x)) \rightarrow a(q(x)) \\
 q(e) \rightarrow e
 \end{array}$$

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

→ *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.

Nicer....

Q1 “subsumes” Q2, if there exists an **MSOTT/TOP Q3** such that
for all inputs s,
 $Q2(s) = Q3(Q1(s))$

Decidable?

Seems difficult...

THE END!!