

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

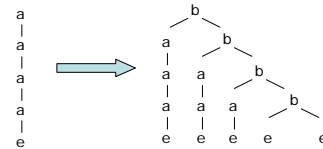
Prologue Tree Transducers

= (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]

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



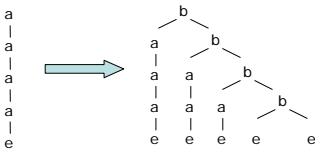
Prologue Tree Transducers

= (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: $M_1(s) = M_2(s)$.

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 [Plan-X 2007, with Helmut Seidl]

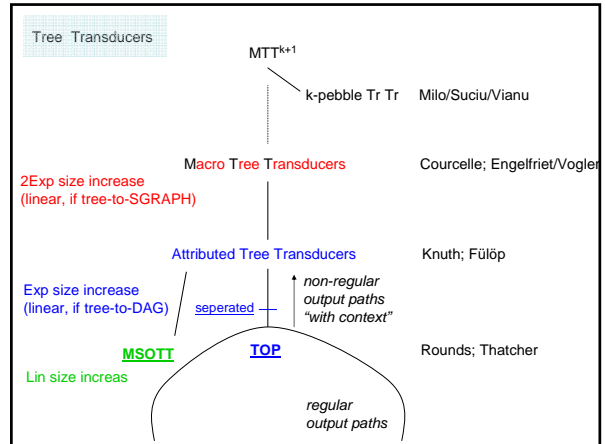
Canonical normal form for TOPs: "uniform and earliest"

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	
<ul style="list-style-type: none"> • <i>nondeterministic (one-way) finite state transducers</i> undecidable [Griffiths68] (→ reduction from PCP, use complement and union) 	
<ul style="list-style-type: none"> • <i>deterministic (one-way) finite state transducers</i> decidable [Gurari82] (→ use Parikh property) 	
<ul style="list-style-type: none"> • <i>deterministic top-down tree transducers</i> decidable [Esik80] 	
<ul style="list-style-type: none"> • <i>nonnested, seperated attributed/marco tree transducers</i> decidable [Courcelle/Franchi-Zanetacci82] → seperated = can be evaluated in two phases, (1) only inherited, over Δ_{in} (2) only synthesized, over Δ_{syn} 	
<ul style="list-style-type: none"> • <i>MSO definable tree transducers</i> decidable [Engelfriet/Maneth05] (→ use Parikh property) 	



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

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

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

$$q(\#(x), y) \rightarrow \#(y)$$

$$q(\$(x), y) \rightarrow \$$$

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)

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

$$q(e, y) \rightarrow y$$

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

$$p(e) \rightarrow \lambda$$

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

$$q(e) \rightarrow \lambda$$

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

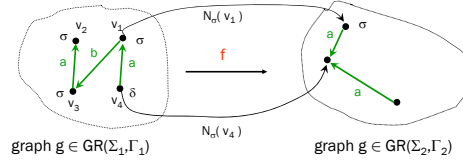
If time permits: 5. Uniform & Earliest TOPs
6. Deciding Equivalence of TOPs

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 $\gamma \in \Gamma_2$

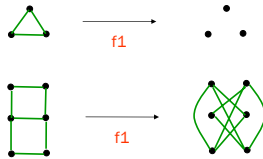
→ Define output graph, in terms of input graph.
Interpretation of one logical structure in another. (FMT: "reductions")



1. MSO Definable Translations (det.)

Example: edge complement f_1

dom: true
 $N_\sigma(x)$: $\text{lab}_\sigma(x)$ for all σ
 $E(x, y)$: $\neg \text{edg}(x, y)$



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 $\gamma \in \Gamma_2$

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

$U = \{ u \in V \mid \text{there is a unique } \sigma \in \Sigma_2 \text{ such that } (g, u) \models N_\sigma(x) \}$
 $F = \{ (u, v) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_\gamma(x, y) \}$
 $\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$

1. MSO Definable Translations (det.)

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

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 $\gamma \in \Gamma_2$

Input graph $g = (V, E)$ → $\text{size}(f(g)) = |U|$
 output graph $h = f(g) = (U, F)$ is $\leq \text{size}(g) = |V|$

$U = \{ u \in V \mid \text{there is a unique } \sigma \in \Sigma_2 \text{ such that } (g, u) \models N_\sigma(x) \}$
 $F = \{ (u, v) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_\gamma(x, y) \}$
 $\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$

1. MSO Definable Translations

MSO 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 $\gamma \in \Gamma_2$

Input graph $g = (V, E)$ → $\text{size}(f(g)) = |U|$
 output graph $h = f(g) = (U, F)$ is $\leq \text{size}(g) = |V|$

$U = \{ u \in V \mid \text{there is a unique } \sigma \in \Sigma_2 \text{ such that } (g, u) \models N_\sigma(x) \}$
 $F = \{ (u, v) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_\gamma(x, y) \}$
 $\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$

C: finite "copy" set $(\text{dom}, C, \{N_{\sigma,c}\}_{\sigma \in \Sigma_2, c \in C}, \{E_{\gamma,c,d}\}_{\gamma \in \Gamma_2, c,d \in C})$

$U = \{ (u, c) \in V \times C \mid \exists! \sigma \in \Sigma_2: (g, u) \models N_{\sigma,c}(x) \}$

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

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

1. MSO Definable Translations

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 $\gamma \in \Gamma_2$

Input graph $g = (V, E)$ $\rightarrow \text{size}(f(g)) = |U|$
 output graph $h = f(g) = (U, F)$ $\text{is } \leq \text{size}(g) = |V|$

$U = \{ u \in V \mid \text{there is a unique } \sigma \in \Sigma_2 \text{ such that } (g, u) \models N_\sigma(x) \}$
 $F = \{ (u, \gamma, v) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_\gamma(x, y) \}$
 $\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$

C: finite "copy" set $(\text{dom}, C, \{N_{\sigma,c} \mid \sigma \in \Sigma_2, c \in C\}, \{E_{\gamma,c,d} \mid \gamma \in \Gamma_2, c, d \in C\})$

$U = \{ (u, c) \in V \times C \mid \exists! \sigma \in \Sigma_2: (g, u) \models N_{\sigma,c}(x) \}$
 $E = \{ ((u, c), \gamma, (v, d)) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_{\gamma,c,d}(x, y) \}$
 $\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$ $\rightarrow \text{size}(f(g)) \leq |C| \cdot \text{size}(g)$

1. MSO Definable Translations

MSO string-to-string transducer Tab

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

$N_{a,A}(x) \wedge \text{lab-a}(x)$ $N_{b,B}(x) \wedge \text{lab-a}(x)$ $N_{e,A}(x) \wedge \text{lab-a}(x)$
 $N_{e,A}(x) \wedge \text{false}$ $N_{e,B}(x) \wedge \text{lab-e}(x)$ $N_{e,B}(x) \wedge \text{lab-e}(x)$ $\langle \text{all others are formulae} = \text{false} \rangle$

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

1. MSO Definable Translations

MSO tree-to-string transducer Tyield

dom = true $\langle \text{all others are set to false...!} \rangle$

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

MSO Definable Translations

Example Tyield

Translate a tree into its yield/frontier

Corollary

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

\rightarrow Strict superset
 Generate
 $\{ a^n b^n c^n d^n \mid n \geq 0 \}$

FACTS

DMSOTT

- \rightarrow Closed under composition
- \rightarrow inverses preserve REGT

DMSOTT(REGT) \leftarrow "most beautiful class of tree languages! ☺"
 = MSOTT(REGT)

- \rightarrow Closed under MSOTT
- \rightarrow path languages(MSOTT(REGT))
- = string languages(MSOTT(REGT))
- = MSOTS(REGT)

\rightarrow are **Parikh**

FACTS

DMSOTT

- \rightarrow Closed under composition
- \rightarrow inverses preserve REGT

DMSOTT(REGT) \leftarrow "most beautiful class of tree languages! ☺"
 = MSOTT(REGT)

- \rightarrow Closed under MSOTT
- \rightarrow path languages(MSOTT(REGT))
- = string languages(MSOTT(REGT))
- = MSOTS(REGT)

\rightarrow 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**

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

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

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

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

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...

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

Languages in MSOTT(REGT)
→ are **Parikh**

2. Parikh's Theorem and Semilinear Sets

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

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

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

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

A set $P \subseteq \mathbb{N}^k$ 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 MSOTT(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

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

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

A set $P \subseteq \mathbb{N}^k$ 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**.

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

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

$\text{Par}(L)$ is semilinear (i.e., $= \text{Par}(R)$ for some reg.l. R)

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

Proof. Let $P = \{ (n,n) \mid n \in \mathbb{N} \} = \text{Par}((ab)^+)$

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

Alternative Proof??

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

$\text{Par}(L)$ is semilinear (i.e., $= \text{Par}(R)$ for some reg.l. R)

Lemma It is decidable for a **semilinear set** $S \subseteq \mathbb{N}^2$, whether there exists $n \in \mathbb{N}$ 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. $\text{Par}(S) = R$.

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. **THEOREM** Let's first do **string output**
The Equivalence problem for **DMSOTS** is **decidable**.

3. **THEOREM** Let's first do **string output**
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. **THEOREM** Let's first do **string output**
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. **THEOREM** Let's first do **string output**
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 $N2^b$)

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. **THEOREM** Let's first do **string output**
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. **THEOREM** Let's first do **string output**
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-Emptiness w. $L3$ \square

3. **THEOREM** Now, let's do **tree output**
The Equivalence problem for DMSOTT is decidable.

T-pre is in **DMSOTS**
(similar to Tyield)

3. **THEOREM** Now, let's do **tree output**
The Equivalence problem for DMSOTT is decidable.

DMSOTTs M1, M2,

Construct $N1 = M1 ; T\text{-pre}$
 $N2 = M2 ; T\text{-pre}$

$M1$ equiv. to $M2$ iff $N1$ equiv. to $N2$

Decidable
(because N1, N2 are DMSOTS's) \square

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 dedicable 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,

$$Q2(s) = Q3(Q1(s))$$

"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,

$$Q2(s) = Q3(Q1(s))$$

"view"

Decidable?
Seems difficult...

THE END?

Back to top-down tree transducers.

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

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

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

$q0(a(x)) \rightarrow b(q(x), q0(x))$ $q0(e) \rightarrow e$ $q(a(x)) \rightarrow a(q(x))$ $q(e) \rightarrow e$	$p0(a(x)) \rightarrow p(x)$ $p0(e) \rightarrow e$ $p(a(x)) \rightarrow b(a(q(x)), p(x))$ $p(e) \rightarrow b(e, e)$
---	--

a
|
 a
|
 a
|
 a
|
 e

→

b
/ \
 a b
| / \
 a a b
| / \
 a a a b
| / \
 e e e e e

5. Uniform and Earliest TOPs

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

$$\begin{array}{c} b \\ / \ \\ a \quad b \\ | \quad / \ \\ T \quad T \quad T \end{array} = t$$

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

→ pattern order $\perp \sqsubseteq p$ 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)

5. Uniform and Earliest TOPs

Deterministic TOP $T = (Q, \Sigma, \Delta, \delta, A)$ "axiom" (start) tree

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

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

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

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

$p0(a(x)) \rightarrow p(x)$ $p0(e) \rightarrow e$ $p(a(x)) \rightarrow b(a(q(x)), p(x))$ $p(e) \rightarrow b(e, e)$ $q(a(x)) \rightarrow a(q(x))$ $q(e) \rightarrow e$	$\text{pref}(p0) = T$ $\text{pref}(p) = b(T, T)$
---	---

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

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

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

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

Definition

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

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

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

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

$$\begin{aligned} p0(a(x)) &\rightarrow p(x) \quad b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ p0(e) &\rightarrow e \\ p(a(x)) &\rightarrow b(a(q(x)), p(x)) \\ p(e) &\rightarrow b(e, e) \\ q(a(x)) &\rightarrow a(q(x)) \\ q(e) &\rightarrow e \end{aligned}$$

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

$$\begin{aligned} p0(a(x)) &\rightarrow p(x) \quad b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ p0(e) &\rightarrow e \\ p(a(x)) &\rightarrow b(a(q(x)), p(x)) \\ p(e) &\rightarrow b(e, e) \\ q(a(x)) &\rightarrow a(q(x)) \\ q(e) &\rightarrow e \end{aligned}$$

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

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

6. Deciding Equivalence of TOPs

$$\begin{aligned} p0(a(x)) &\rightarrow p(x) \quad b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x)) \\ p0(e) &\rightarrow e \\ p(a(x)) &\rightarrow b(a(q(x)), p(x)) \\ p(e) &\rightarrow b(e, e) \\ q(a(x)) &\rightarrow a(q(x)) \\ q(e) &\rightarrow e \end{aligned}$$

$$\langle p, 1 \rangle(a(x)) \rightarrow a(q(x))$$

$$\langle p, 1 \rangle(e) \rightarrow e$$

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

$$\langle p, 2 \rangle(e) \rightarrow e$$

6. Deciding Equivalence of 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

6. Deciding Equivalence of 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

6. Deciding Equivalence of TOPs

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

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

Uniform&Earliest(T1)
 w. p_0 renamed to q_0
 equals
 T2

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

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

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

→ 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 an MSOTT/TOP Q_3 such that
 for all inputs s ,
 $Q_2(s) = Q_3(Q_1(s))$

Decidable?
 Seems difficult...

THE END!!