

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)

February 22nd, 2007 University of Tokyo

Main Theme *Transfer Properties from FLT*

[Chomsky 1956]
CF Parsing
[Cooke, Younger, Kasami 1962, 66, 67]

Attribute Grammars
[Knuth 1968]

Finite Automata
[Rabin, Scott 1959]

Transducers
[Aho/Ullman 1971] & earlier

→

→

→

→

PL's

Compilers

Verification

(XML) Databases

FLT = Formal Language Theory

Main Theme *Transfer Properties from FLT*

[Chomsky 1956]
CF Parsing
[Cooke, Younger, Kasami 1962, 66, 67]

Attribute Grammars
[Knuth 1968]

Finite Automata
[Rabin, Scott 1959]

Transducers
[Aho/Ullman 1971] & earlier

→

→

→

→

→

→

→

PL's

Compilers

Verification

(XML) Databases

Today's Topic
FLT: the Parikh Property

FLT = Formal Language Theory

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]

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

$q_0(e) \rightarrow e$

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

$q(e) \rightarrow e$

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

$p_0(e) \rightarrow e$

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

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

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]

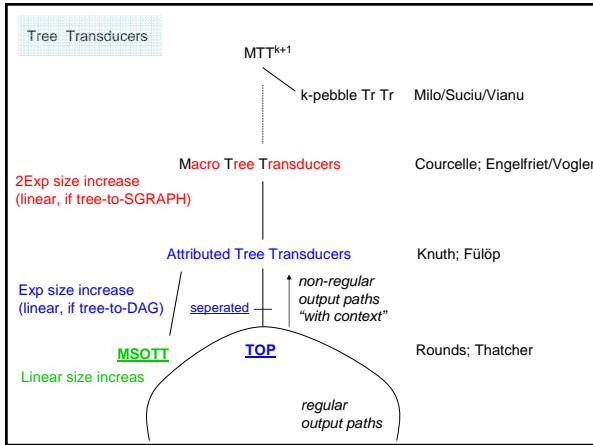
Canonical normal form for TOPs: "**uniform and earliest**"

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_{in} (2) only synthesized, over Δ_{syn}
- **MSO definable tree transducers** **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

$q0(a(x1, x2)) \rightarrow$

```

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

$q(a(x1, x2), y) \rightarrow a(q(x1, y), q(x2, y))$
 $q(\#(x), y) \rightarrow \#(y)$
 $q(\$(x), y) \rightarrow \$$

r: find \$ and copy subtree..

Macro Tree Transducer (MTT) [Engelfriet/Vogler1985]

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

```

function q(s: 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)

```

q(a(x), y) → a(q(x, q(x, a(y))))
q(e, y)    → y

```

```

p(a(x)) → p(x) a a p(x)
p(e)    → λ

```

```

q(a(x)) → a q(x) q(x) a
q(e)    → λ

```

Equivalent?!
OPEN for 30 years... ©

But, unfortunately

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

Outline

- MSO Definable Translations
- Parikh's Theorem and Semilinear Sets
- Deciding Equivalence of MSOTT
- 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 $\sigma \in \Sigma_2$
 $E_i(x, y)$: 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")

graph $g \in GR(\Sigma_1, \Gamma_1)$ graph $g \in GR(\Sigma_2, \Gamma_2)$

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 $\sigma \in \Sigma_2$
 $E_i(x, y)$: 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 $g = (V, E)$ $\rightarrow \text{size}(f(g)) = |U|$
output graph $h = f(g) = (U, F)$ $\text{is } \leq \text{size}(g) = |V|$

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

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

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

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 $h = f(g) = (U, F)$ $\text{is } \leq \text{size}(g) = |V|$

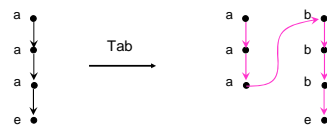
$U = \{ u \in V \mid \text{there is a unique } \sigma \in \Sigma_2 \text{ such that } (g, u) \models N_\sigma(x) \}$
 $F = \{ (u, \gamma, v) \mid \exists! \gamma \in \Gamma_2: (g, u, v) \models E_\gamma(x, y) \}$
 $\forall u \in U: \text{lab-}\sigma(u) \text{ iff } (g, u) \models N_\sigma(x)$

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

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

1. MSO Definable Translations

MSO string-to-string transducer Tab



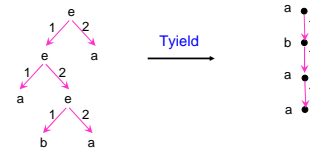
$C = \{ A, B \}$ $\text{dom} = \text{string} \wedge (\forall x)(\text{lab-e}(x) \leftrightarrow \text{last}(x))$

$N_{b,A}(x) = \text{lab-a}(x)$ $N_{b,B}(x) = \text{lab-a}(x)$
 $N_{a,A}(x) = \text{false}$ $N_{a,B}(x) = \text{lab-e}(x)$ $\leftarrow \text{all others are set to false} >$

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

1. MSO Definable Translations

MSO tree-to-string transducer Tyield



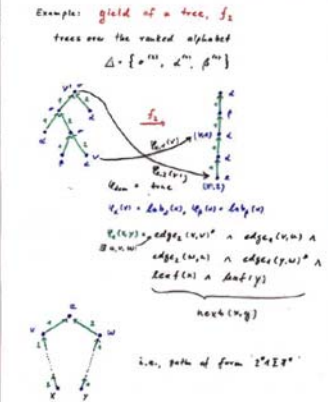
dom = true $\leftarrow \text{all others are set to false..!} >$

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

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

- Closed under composition
- inverses preserve REGT

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

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

→ are **Parikh**

FACTS

DMSOTT

- Closed under composition
- inverses preserve REGT

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

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

→ are **Parikh**

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.

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**.

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...

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

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.

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** $S \subseteq \mathbb{N}^2$,
 whether there exists $n \in \mathbb{N}$ such that $(n,n) \in S$.

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

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** $S \subseteq \mathbb{N}^2$,
 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. $Par(S) = R$.

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 (B) Construct 'Parikh-' $R_{a,b}$ for $\text{Out}(N12_{a,b})$
 (A) construct $N12^{a,b}$ (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 $N1 = M1 ; T\text{-pre}$

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

$M1 \text{ equiv. to } M2$ iff $N1 \text{ equiv. to } N2$

$\underbrace{\hspace{10em}}$
 Decidable
 (because N1, N2 are DMSOTT'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!

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

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

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

$\text{pref}(p0) = T$
 $\text{pref}(p) = b(T, T)$

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

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

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

Definition

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

$\text{pref}(p0) = T$
 $\text{pref}(p) = b(T, T)$

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 are 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

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

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

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

$p0(a(x)) \rightarrow b(\langle p, 1 \rangle(x), \langle p, 2 \rangle(x))$

$p0(e) \rightarrow e$

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

6. Deciding Equivalence of TOPs

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

M is uniform (because it is total).

→ Make it earliest

pref(p) = b(T, T)

In all right-hand sides, change p(x) into b(⟨p, 1⟩(x), ⟨p, 2⟩(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

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

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

6. Deciding Equivalence of TOPs

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

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

<p, 1> state-equivalent to q
 <p, 2> state-equivalent to p0

6. Deciding Equivalence of TOPs

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

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

<p, 1> state-equivalent to q
 <p, 2> state-equivalent to p0

6. Deciding Equivalence of TOPs

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

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

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

equals

T2

<p, 1> state-equivalent to q
 <p, 2> state-equivalent to p0

$$\begin{array}{l}
 q_0(a(x)) \rightarrow b(q(x), q_0(x)) \\
 q_0(e) \rightarrow e \\
 q(a(x)) \rightarrow a(q(x)) \\
 q(e) \rightarrow e
 \end{array}$$

Equivalence of TOPs with regular look-ahead

M = (Q, Σ, Δ, δ, A, B)

B det. bottom-up tree automaton
With $L(p_1) \cap L(p_2) = \emptyset$ for all states p1, p2

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

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.

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!!