

Open Problems in Tree Language Theory

Sebastian Maneth

Tree Automata Workshop
Bonn, June 7th, 2006

Outline

1. Tree Languages and Tree Transducers
2. Logic
3. Rankedness versus Unrankedness

1. Tree Languages and Transducers

- (1) The class **REGT** of **regular tree languages**
(accepted, e.g., by nondet BU/TD or det BU tree automata.)

Problem 0

What is the **complexity** of the *word problem for det. BU tree automata*, i.e., of the problem whether A accepts t, for a given (binary) tree t and a deterministic bottom-up tree automaton A.

1. Tree Languages and Transducers

- (1) The class **REGT** of **regular tree languages**
(accepted, e.g., by nondet BU/TD or det BU tree automata.)

Problem 0

What is the **complexity** of the *word problem for det. BU tree automata*, i.e., of the problem whether A accepts t, for a given (binary) tree t and a deterministic bottom-up tree automaton A.

[Lohrey01]: complexity of this problem for

- **nondet. BU/TD tree automata**: **LOGCFL-complete**
- **det. BU tree automata**: **LOGDCFL**
- **deterministic. TD tree automata**: **LOGSPACE-complete**.

1. Tree Languages and Transducers

(1) The class $\text{HOM}(\text{REGT})$

REGT = the class of **regular tree languages**

HOM = class of **tree homomorphisms**

= 1-state det. top-down tree transducers

= 1-state det. bottom-up tree transducers = ...

Example:

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

$a^n(e)$ into *full binary tree* of height n

→ The tree language $h(a^n(e))$ is **NOT regular !!**

1. Tree Languages and Transducers

(1) The class $\text{HOM}(\text{REGT})$

REGT = the class of regular tree languages

HOM = class of tree homomorphisms

= 1-state det. top-down tree transducers

= 1-state det. bottom-up tree transducers = ...

Example:

$h: q(a(x)) \rightarrow b(q(x), q(x))$

$q(e) \rightarrow e$

$q(b(x_1, x_2)) \rightarrow b(q(x_1), q(x_2))$

→ The tree language $h(a^n(e))$ is **NOT regular !!**

1. Tree Languages and Transducers

(1) The class $\text{HOM}(\text{REGT})$

REGT = the class of regular tree languages

HOM = class of tree homomorphisms

= 1-state det. top-down tree transducers

= 1-state det. bottom-up tree transducers = ...

Example:

$h: q(a(x)) \rightarrow b(q(x), q(x))$

$q(e) \rightarrow e$

$q(b(x_1, x_2)) \rightarrow b(q(x_1), q(x_2))$

→ The tree language $h(a^n(e))$ is **NOT regular** !!

→ But $h(a^n(e) \cup T_\Sigma)$ **IS** regular for $\Sigma = \{b^{(2)}, e^{(0)}\}$

1. Tree Languages and Transducers

(1) The class $\text{HOM}(\text{REGT})$

REGT = the class of regular tree languages

HOM = class of tree homomorphisms

= 1-state det. top-down tree transducers

= 1-state det. bottom-up tree transducers

Problem 1

Given h, R . Is it **decidable** whether $h(R)$ is regular?

→ mentioned, e.g., in [Fulop94, TATA-book,
Bogaert/Seynhaeve/Tison00]

1. Tree Languages and Transducers

(1) The class $\text{HOM}(\text{REGT})$

REGT = the class of regular tree languages

HOM = class of tree homomorphisms

= 1-state det. top-down tree transducers

= 1-state det. bottom-up tree transducers

Problem 1

Given h, R . Is it **decidable** whether $h(R)$ is regular?

→ mentioned, e.g., in [Fulop94, TATA-book,
Bogaert/Seynhaeve/Tison00]

h : copying only between siblings of a node

THEN **decidable!** (simulate $h(R)$ by an *automaton with equality constraints between brothers*)



1. Tree Languages and Transducers

(1) The class $\text{HOM}(\text{REGT})$

REGT = the class of regular tree languages

HOM = class of tree homomorphisms

= 1-state det. top-down tree transducers

= 1-state det. bottom-up tree transducers

Problem 2

Is $\text{HOM}(\text{REGT})$ closed under **intersection**?

1. Tree Languages and Transducers

(1) The class $\text{HOM}(\text{REGT})$

REGT = the class of regular tree languages

HOM = class of tree homomorphisms

= 1-state det. top-down tree transducers

= 1-state det. bottom-up tree transducers

Problem 2

Is $\text{HOM}(\text{REGT})$ closed under **intersection**?

→ Probably not.

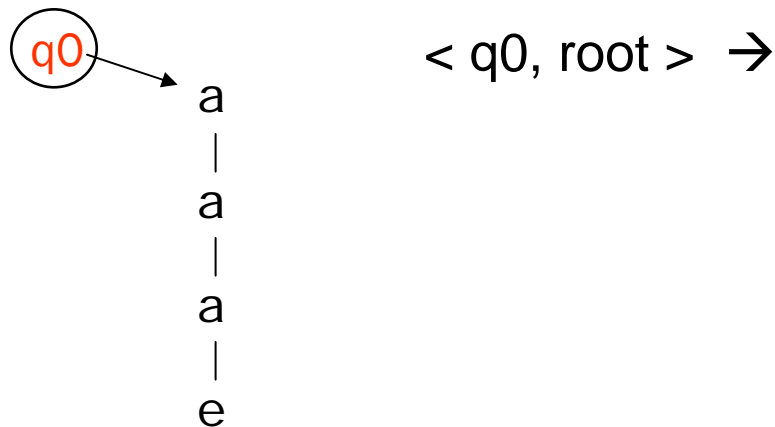
$\text{HOM}(\text{REGT})$ is NOT *effectively* closed under intersection.

→ code PCP into intersection of three $\text{HOM}(\text{REGT})$ languages..

1. Tree Languages and Transducers

(2) **Tree-walking Transducers** $(Q, \Sigma, \Delta, q_0, R)$

| | | | |
|---------------|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | \rightarrow | $a(\langle q_0, \downarrow_1 \rangle)$ | |
| $q_0, e, 1$ | \rightarrow | $\langle q, \uparrow \rangle$ | child number of current input node |
| $q_0, e, 0$ | \rightarrow | e | |
| $q, a, 1$ | \rightarrow | $b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | \rightarrow | $b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | \rightarrow | $c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | \rightarrow | e | |

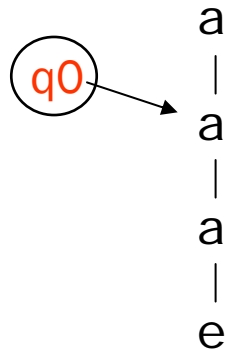


1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | | |
|---------------|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | \rightarrow | $a(\langle q_0, \downarrow_1 \rangle)$ | |
| $q_0, e, 1$ | \rightarrow | $\langle q, \uparrow \rangle$ | child number of current input node |
| $q_0, e, 0$ | \rightarrow | e | |
| $q, a, 1$ | \rightarrow | $b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | \rightarrow | $b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | \rightarrow | $c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | \rightarrow | e | |

$$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$$

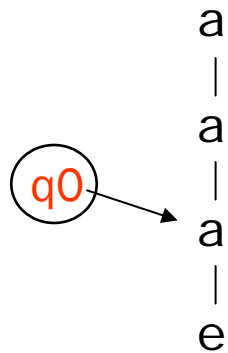


1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | |
|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | $\rightarrow a(\langle q_0, \downarrow_1 \rangle)$ | child number of current input node |
| $q_0, e, 1$ | $\rightarrow \langle q, \uparrow \rangle$ | |
| $q_0, e, 0$ | $\rightarrow e$ | |
| $q, a, 1$ | $\rightarrow b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | $\rightarrow b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | $\rightarrow c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | $\rightarrow e$ | |

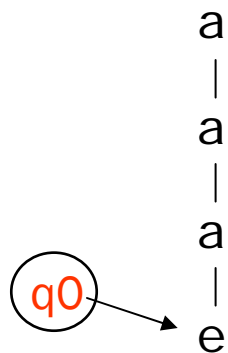
$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$
 $\rightarrow a(a(\langle q_0, 11 \rangle))$



1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | |
|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | $\rightarrow a(\langle q_0, \downarrow_1 \rangle)$ | child number of current input node |
| $q_0, e, 1$ | $\rightarrow \langle q, \uparrow \rangle$ | |
| $q_0, e, 0$ | $\rightarrow e$ | |
| $q, a, 1$ | $\rightarrow b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | $\rightarrow b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | $\rightarrow c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | $\rightarrow e$ | |

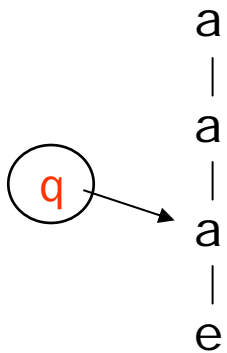


$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$
 $\rightarrow a(a(\langle q_0, 11 \rangle))$
 $\rightarrow a(a(a(\langle q_0, 111 \rangle)))$

1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | |
|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | $\rightarrow a(\langle q_0, \downarrow_1 \rangle)$ | child number of current input node |
| $q_0, e, 1$ | $\rightarrow \langle q, \uparrow \rangle$ | |
| $q_0, e, 0$ | $\rightarrow e$ | |
| $q, a, 1$ | $\rightarrow b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | $\rightarrow b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | $\rightarrow c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | $\rightarrow e$ | |

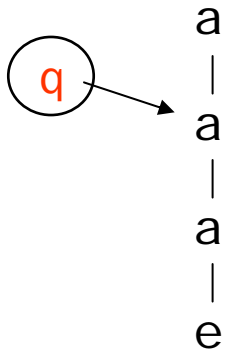


$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$
 $\rightarrow a(a(\langle q_0, 11 \rangle))$
 $\rightarrow a(a(a(\langle q_0, 111 \rangle)))$
 $\rightarrow a(a(a(\langle q, 11 \rangle)))$

1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | |
|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | $\rightarrow a(\langle q_0, \downarrow_1 \rangle)$ | child number of current input node |
| $q_0, e, 1$ | $\rightarrow \langle q, \uparrow \rangle$ | |
| $q_0, e, 0$ | $\rightarrow e$ | |
| $q, a, 1$ | $\rightarrow b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | $\rightarrow b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | $\rightarrow c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | $\rightarrow e$ | |

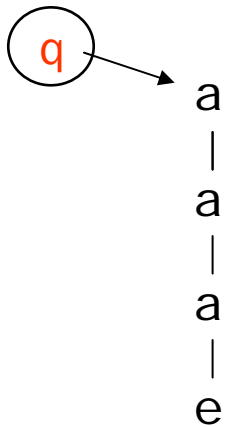


$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$
 $\rightarrow a(a(\langle q_0, 11 \rangle))$
 $\rightarrow a(a(a(\langle q_0, 111 \rangle)))$
 $\rightarrow a(a(a(\langle q, 11 \rangle)))$
 $\rightarrow a(a(a(b(\langle q, 1 \rangle))))$

1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | |
|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | $\rightarrow a(\langle q_0, \downarrow_1 \rangle)$ | child number of current input node |
| $q_0, e, 1$ | $\rightarrow \langle q, \uparrow \rangle$ | |
| $q_0, e, 0$ | $\rightarrow e$ | |
| $q, a, 1$ | $\rightarrow b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | $\rightarrow b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | $\rightarrow c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | $\rightarrow e$ | |

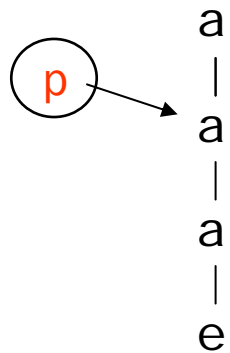


$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$
 $\rightarrow a(a(\langle q_0, 11 \rangle))$
 $\rightarrow a(a(a(\langle q_0, 111 \rangle)))$
 $\rightarrow a(a(a(\langle q, 11 \rangle)))$
 $\rightarrow a(a(a(b(\langle q, 1 \rangle))))$
 $\rightarrow a(a(a(b(b(\langle q, \text{root} \rangle)))))$

1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | |
|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | $\rightarrow a(\langle q_0, \downarrow_1 \rangle)$ | child number of current input node |
| $q_0, e, 1$ | $\rightarrow \langle q, \uparrow \rangle$ | |
| $q_0, e, 0$ | $\rightarrow e$ | |
| $q, a, 1$ | $\rightarrow b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | $\rightarrow b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | $\rightarrow c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | $\rightarrow e$ | |

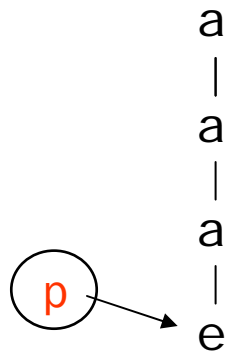


$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$
 $\rightarrow a(a(\langle q_0, 11 \rangle))$
 $\rightarrow a(a(a(\langle q_0, 111 \rangle)))$
 $\rightarrow a(a(a(\langle q, 11 \rangle)))$
 $\rightarrow a(a(a(b(\langle q, 1 \rangle))))$
 $\rightarrow a(a(a(b(b(\langle q, \text{root} \rangle)))))$
 $\rightarrow a^3(b^3(c(\langle p, 1 \rangle)))$

1. Tree Languages and Transducers

(2) Tree-walking Transducers $(Q, \Sigma, \Delta, q_0, R)$

| | | |
|---------------|---|---------------------------------------|
| $q_0, a, 0/1$ | $\rightarrow a(\langle q_0, \downarrow_1 \rangle)$ | child number of current input node |
| $q_0, e, 1$ | $\rightarrow \langle q, \uparrow \rangle$ | |
| $q_0, e, 0$ | $\rightarrow e$ | |
| $q, a, 1$ | $\rightarrow b(\langle q, \uparrow \rangle)$ | |
| $q, a, 0$ | $\rightarrow b(c(\langle p, \downarrow_1 \rangle))$ | |
| $p, a, 1$ | $\rightarrow c(\langle p, \downarrow_1 \rangle)$ | |
| $p, e, 1$ | $\rightarrow e$ | |



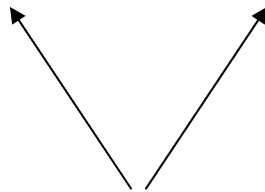
$\langle q_0, \text{root} \rangle \rightarrow a(\langle q_0, 1 \rangle)$
 $\rightarrow a(a(\langle q_0, 11 \rangle))$
 $\rightarrow a(a(a(\langle q_0, 111 \rangle)))$
 $\rightarrow a(a(a(\langle q, 11 \rangle)))$
 $\rightarrow a(a(a(b(\langle q, 1 \rangle))))$
 $\rightarrow a(a(a(b(b(\langle q, \text{root} \rangle)))))$
 $\rightarrow a^3(b^3(c(\langle p, 1 \rangle)))$
 $\rightarrow \rightarrow \mathbf{a^3(b^3(c^3(e)))}$

1. Tree Languages and Transducers

(2) **Tree-walking Transducers** $(Q, \Sigma, \Delta, q_0, R)$

$q_0, e, 0/1 \rightarrow e$

$q_0, a, 0/1 \rightarrow b(\langle q_0, \Downarrow_1 \rangle, \langle q_0, \Downarrow_1 \rangle)$



Non-monadic output:

\rightarrow split in independent (parallel) copies!

This is the homomorphism **fbt** of before.

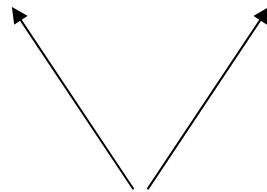
$a^n(e)$ into *full binary tree* of height n

1. Tree Languages and Transducers

(2) **Tree-walking Transducers** $(Q, \Sigma, \Delta, q_0, R)$

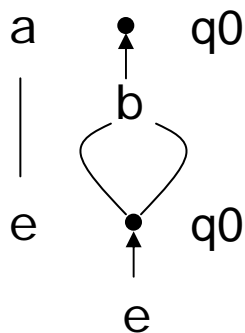
$q_0, e, 0/1 \rightarrow e$

$q_0, a, 0/1 \rightarrow b(\langle q_0, \Downarrow_1 \rangle, \langle q_0, \Downarrow_1 \rangle)$



Non-monadic output:

\rightarrow split in independent (parallel) copies!



Naturally describes the output DAG of an **attribute grammar**.

\rightarrow #of different output subtrees is linearly bounded by the size of the input tree!

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 3

Is equivalence decidable for deterministic TWTTs?

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 3

Is equivalence decidable for deterministic TWTTs?

If of *linear size increase (LSI)*, THEN **decidable!**

because

MSO-definable tree translations have decidable equivalence

[Engelfriet/M.05]

LSI property is decidable [Engelfriet/M.03b]

(nondet. case: undecidable [Griffiths68])

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 3

Is equivalence decidable for deterministic TWTTs?

More general:

→ Is equivalence decidable for **compositions** of TWTTs??

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 4

Is it decidable for two TWTTs whether their composition can be realized by one TWTT?

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 4

Is it decidable for two TWTTs whether their composition can be realized by one TWTT?

Conjecture: (1) Composition of TWTT's can be realized by **one TWTT**,
iff #(output subtrees) of T is linear bounded by
size of the input tree.

(2) This property is decidable.

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 4

Is it decidable for two TWTTs whether their composition can be realized by one TWTT?

Note: $\text{dTWTT}^* \cap \text{LSI} \subseteq \text{DMSOTT}$

↑
(effectively) [M.03]

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 4

Is it decidable for two TWTTs whether their composition can be realized by one TWTT?

Note: $\text{dTWTT}^* \cap \text{LSI} \subseteq \text{DMSOTT}$

Btw: **DMSOTT(REGT)**: very nice class of tree languages!

→ path languages = yield languages

→ closed under MSOTT

1. Tree Languages and Transducers

(2) Tree-walking Transducers (TWTTs)

= Tree-walking Automata + Tree Output

= 0-pebble Tree Transducers [Milo/Suciu/Vianu03]

= (partial, nondet.) Attributed Tree Transducers cf. [Engelfriet/M.03a]

Problem 4

Is it decidable for two TWTTs whether their composition can be realized by one TWTT?

Slightly weaker:

Is it decidable for a **1-pebble TWTT**, whether it can be realized by a TWTT?

1. Tree Languages and Transducers

Slightly weaker:

Is it decidable for a 1-pebble TWTT, whether it can be realized by a TWTT?

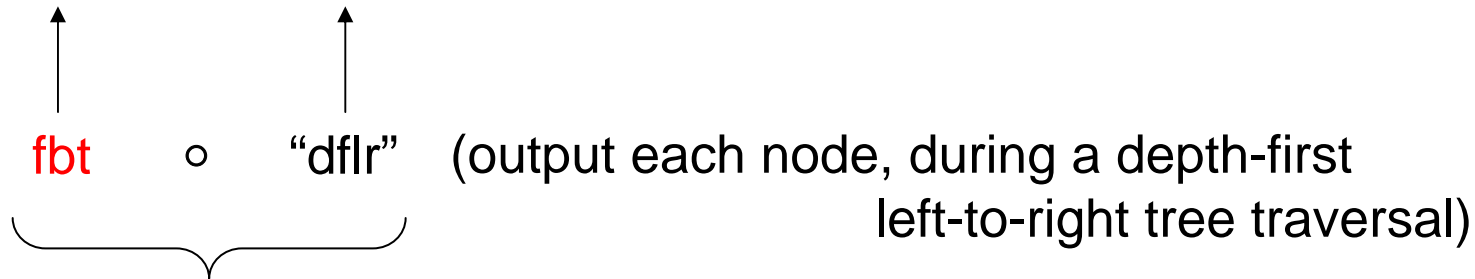
dTWTT \circ dTWTT $\not\subseteq$ n-pebble TWTT

1. Tree Languages and Transducers

Slightly weaker:

Is it decidable for a 1-pebble TWTT, whether it can be realized by a TWTT?

dTWTT \circ dTWTT $\not\subseteq$ n-pebble TWTT



Exponential
height increase!

Lemma: For every n-pebble TWTT M there is a $c > 0$ such that for every s ,

if $M(s)$ is finite then $\text{height}(t) \leq c \cdot \text{size}(s)^{n+1}$
for every output tree t in $M(s)$.

see [Engelfriet/M.03a]

1. Tree Languages and Transducers

Slightly **weaker**:

Is it decidable for a **1-pebble TWTT**, whether it can be realized by a TWTT?

Related open problem for TW-Automata:

Problem 5

Is is decidable for a 1-pebble TW automaton, whether it can be realized by a 0-pebble TW automaton?

Probably very hard...

1. Tree Languages and Transducers

Composition Hierarchies for TWTTs.

Problem 6

Is $\underbrace{\text{TWTT} \circ \text{TWTT} \circ \dots \circ \text{TWTT}}_{n \text{ times}}$ a proper hierarchy wrt n ?

1. Tree Languages and Transducers

Composition Hierarchies for TWTTs.

Problem 6

Is $\underbrace{\text{TWTT} \circ \text{TWTT} \circ \dots \circ \text{TWTT}}_{n \text{ times}}$ a proper hierarchy wrt n ?

Deterministic case: known, even for output **string** languages!

Using “bridge theorems” about the class of output languages of deterministic macro tree transducers [Engelfriet/M02] it can be shown that:

$$\text{yield}(\text{dTWTT}^n(\text{REGT})) \subsetneq \text{yield}(\text{dTWTT}^{n+1}(\text{REGT}))$$

Note that **pebble-hierarchies** can be proved easily,
In both the nondeterministic and the deterministic case, i.e.,

$$n\text{-TWTT} \subsetneq (n+1)\text{-TWTT}$$

and

$$n\text{-dTWT} \subsetneq (n+1)\text{-dTWT}$$

→ with n pebbles translate $s = a^n(e)$ into a monadic tree of height $\text{size}(s)^{n+2}$.

→ Drop pebble-1 and pebble-2 at root and copy input tree.

Repeat

Repeat

→ Move pebble-2 one node down and copy input tree

Until pebble-2 at leaf.

Move pebble-1 one node down and copy input tree

Until pebble-1 at leaf.

Note that **pebble-hierarchies** can be proved easily,
 In both the nondeterministic and the deterministic case, i.e.,

$$n\text{-TWTT} \subsetneq (n+1)\text{-TWTT}$$

and

$$n\text{-dTWT} \subsetneq (n+1)\text{-dTWT} \quad \text{yield}(n\text{-dTWT(REGT)}) \subsetneq \text{yield}((n+1)\text{-dTWT(REGT)})$$

→ with n pebbles translate $s = a^n(e)$ into a monadic tree of height $\text{size}(s)^{n+2}$.

→ Drop pebble-1 and pebble-2 at root and copy input tree.

Repeat

Repeat

→ Move pebble-2 one node down and copy input tree

Until pebble-2 at leaf.

Move pebble-1 one node down and copy input tree

Until pebble-1 at leaf.

Note that **pebble-hierarchies** can be proved easily,
In both the nondeterministic and the deterministic case, i.e.,

$$n\text{-TWTT} \subsetneq (n+1)\text{-TWTT}$$

and

Even:

$$n\text{-dTWT} \subsetneq (n+1)\text{-dTWT}$$

$$\begin{aligned} & \text{yield}(n\text{-dTWT(REGT)}) \\ & \subsetneq \text{yield}((n+1)\text{-dTWT(REGT)}) \end{aligned}$$

Even in the **Automata case**, pebble-hierarchies have been proved!
[Bojanczyk/Samuelides/Schwentick/Segoufin06]

$$n\text{-TWA} \subsetneq (n+1)\text{-TWA} \dots \subseteq \text{PTWA} \subsetneq \text{REGT}$$

and

$$n\text{-dTWA} \subsetneq (n+1)\text{-dTWA}$$

Note that **pebble-hierarchies** can be proved easily,
In both the nondeterministic and the deterministic case, i.e.,

$$n\text{-TWTT} \subsetneq (n+1)\text{-TWTT}$$

and

$$n\text{-dTWT} \subsetneq (n+1)\text{-dTWT}$$

Even:

$$\begin{aligned} & \text{yield}(n\text{-dTWT(REGT)}) \\ & \subsetneq \text{yield}((n+1)\text{-dTWT(REGT)}) \end{aligned}$$

Even in the **Automata case**, pebble-hierarchies have been proved!
[Bojanczyk/Samuelides/Schwentick/Segoufin06]

$$n\text{-TWA} \subsetneq (n+1)\text{-TWA} \dots \subseteq \text{PTWA} \subsetneq \text{REGT}$$

and

$$n\text{-dTWA} \subsetneq (n+1)\text{-dTWA}$$

Open:

Is $\text{dTWA} \subsetneq \text{TWA}$?

2. Logic

Problem 7

Does there exist a natural logical characterization of $TWTT^n$?

→ How to generalize MSO-transductions to non-linear size increase??

2. Logic

Problem 7

Does there exist a natural logical characterization of $TWTT^n$?

→ How to generalize MSO-transductions to non-linear size increase??

Even in the [Automata case](#), pebble-hierarchies have been proved!
[Bojanczyk/Samuelides/Schwentick/Segoufin06]

n -TWA \subsetneq $(n+1)$ -TWA $\dots \subseteq$ PTWA \subsetneq REGT

||

FO + posTC1

[Engelfriet/Hoogeboom06]

2. Logic

Problem 7

Does there exist a natural logical characterization of $TWTT^n$?

→ How to generalize MSO-transductions to non-linear size increase??

Even in the [Automata case](#), pebble-hierarchies have been proved!
[Bojanczyk/Samuelides/Schwentick/Segoufin06]

n -TWA \subsetneq $(n+1)$ -TWA $\dots \subseteq$ PTWA \subsetneq REGT

||

FO + posTC1

[Engelfriet/Hoogeboom06]

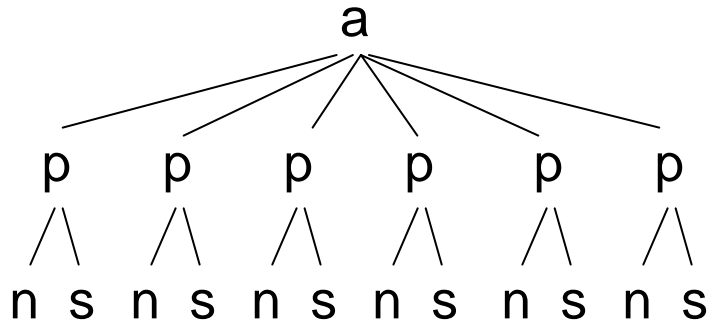
Problem 8

Is FO + TC1 \subsetneq REGT?

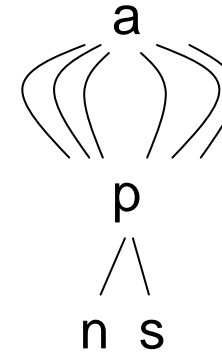
3. Unranked vs Ranked Trees

Problem of **efficient pointer-representations**:

18/19



→
Min. DAG



8/4

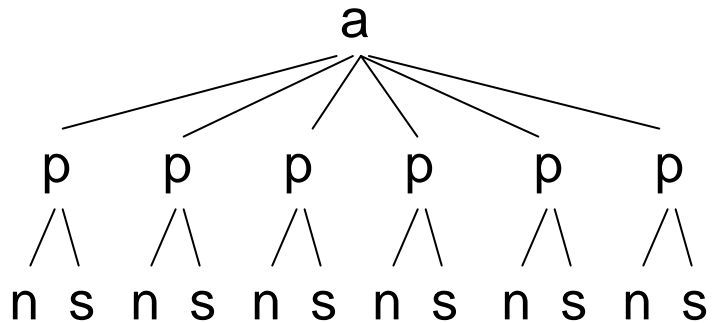
[Buneman/Grohe/Koch03]

For common XML document trees, the minimal unranked DAG has $\approx 10\%$ of the number of edges of the original tree!

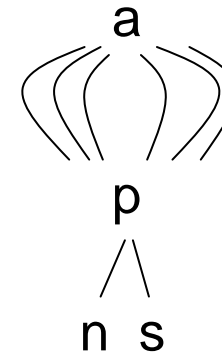
3. Unranked vs Ranked Trees

Problem of **efficient pointer-representations**:

18/19



→
Min. DAG



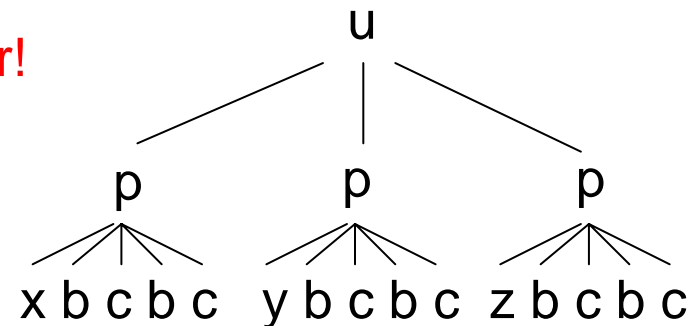
8/4

[Buneman/Grohe/Koch03]

For common XML document trees, the minimal unranked DAG has $\approx 10\%$ of the number of edges of the original tree!

BUT, **minimal binary DAG might be smaller!**

18 edges (unranked) vs
12 edges (binary)



3. Unranked vs Ranked Trees

Problem of **efficient pointer-representations** (DAGs)

- (1) Given an unranked tree: Which binary encoding gives the smallest minimal DAG?
- (2) Given a DTD, which binary encoding gives the smallest minimal DAGs?
- (3) For typical access models (top-down / DOM) what is the relative access speed for a given encoding?

How expensive are update operations?

3. Unranked vs Ranked Trees

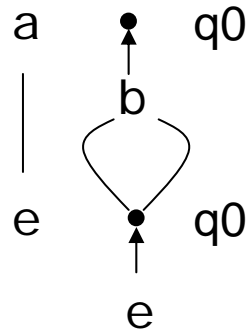
Problem of **efficient pointer-representations** (DAGs)

Given an (unranked or ranked) TWTT M , is it possible to change M so that it outputs **minimal DAGs**?

→ Probably not.

→ How close to the minimal DAGs can one get?

$q_0, e, 0/1 \rightarrow e$
 $q_0, a, 0/1 \rightarrow b(\langle q_0, \Downarrow_1 \rangle, \langle q_0, \Downarrow_1 \rangle)$

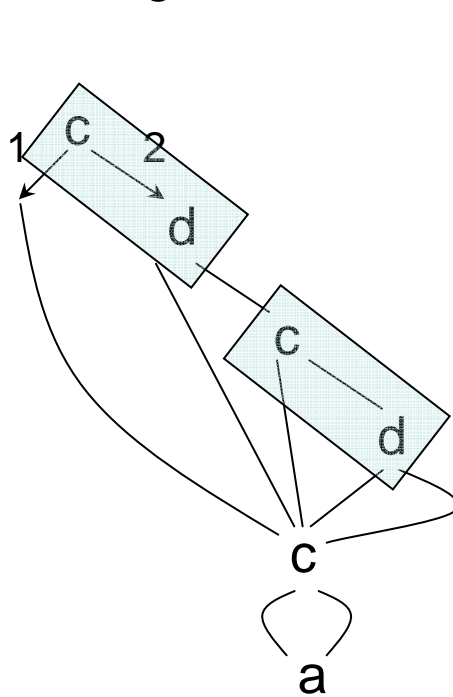


3. Unranked vs Ranked Trees

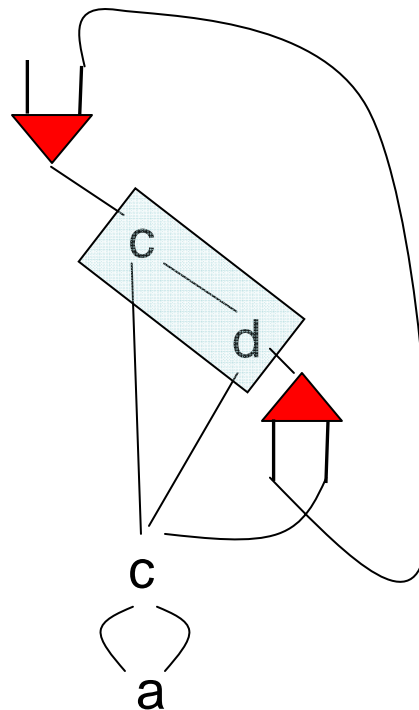
Problem of **efficient pointer-representations**

→ Generalize sharing of subtrees to sharing of connected subgraphs!

What is the **minimal DPDA** (sharing graph / context-free tree grammar) for a given tree?



minimal DAG



minimal sharing graph = minimal cf tree grammar

$$S \rightarrow D(D(A))$$

$$D(y) \rightarrow c(A, d(A, y))$$

$$A \rightarrow c(B, B)$$

$$B \rightarrow a$$

3. Unranked vs Ranked Trees

Problem of **efficient pointer-representations**

→ Generalize sharing of subtrees to sharing of connected subgraphs!

What is the **minimal DPDA** (sharing graph / context-free tree grammar) for a given tree?

NP-complete, already for a given string [Charikar/Lehman/Liu/Panigrahy/
Prabhakaran/Sahai/Shelat05]

Strings: \exists many approximation algorithms, and their approximation ratios are known.

Trees: What are good approximation algorithms?
How good do they approximate?

3. Unranked vs Ranked Trees

Problem of **efficient pointer-representations**

BPLEX (bottom-up, window-based pattern search algorithm)

[Busatto/Lohrey/M05]

- (1) Given an unranked tree: Which binary encoding gives the smallest **minimal context-free tree grammar**?

- (2) Given a DTD, which binary encoding gives the smallest minimal **cf. tree grammars**?

- (3) For typical access models (top-down / DOM) what is the relative access speed for given encoding **on a cf. tree grammar**?

How expensive are update operations?

End of Presentation.



(hopefully)

Time for more
Open Problems...

... a typical pebble game ...