

Type Checking XML Transformations In Polynomial Time

Sebastian Maneth, Thomas Perst,
Helmut Seidl

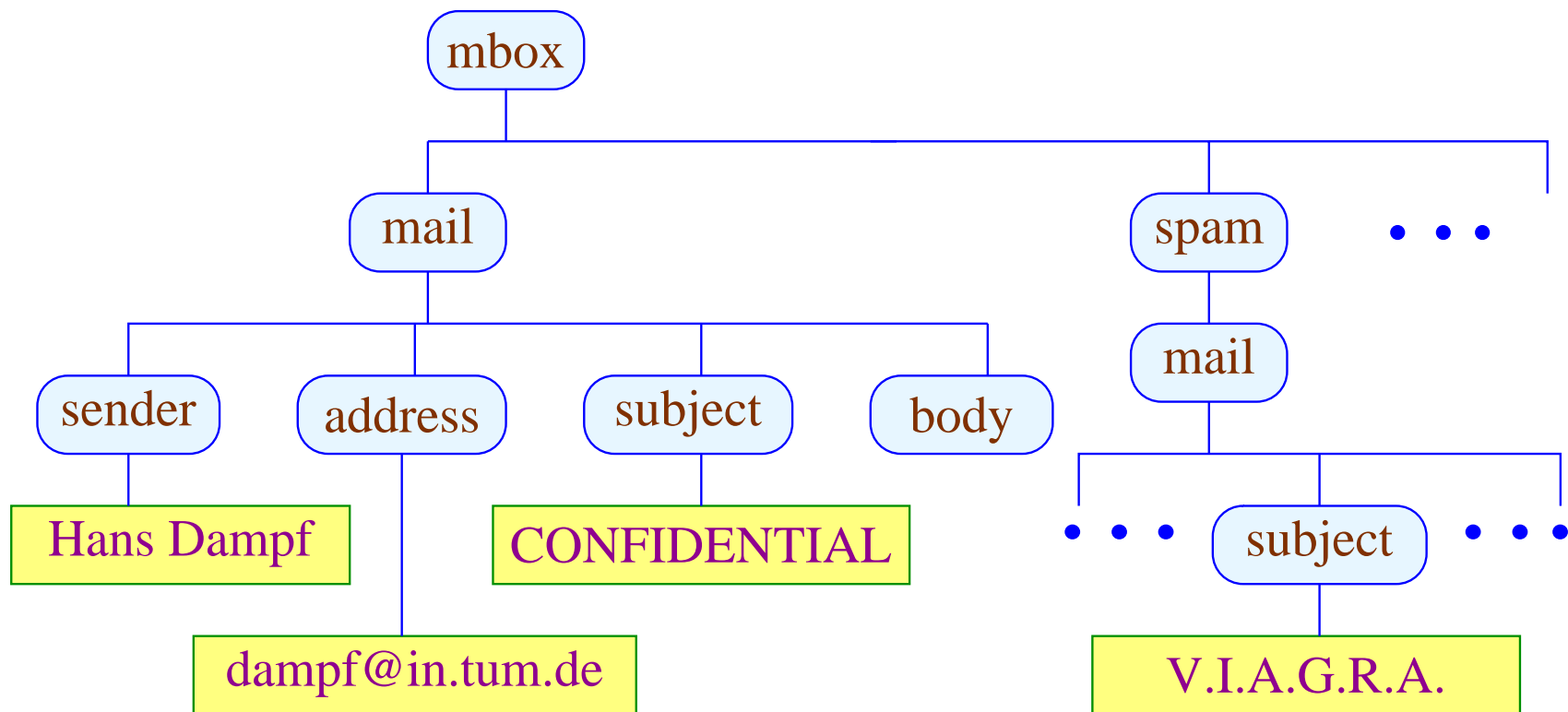
Sydney, München

2006

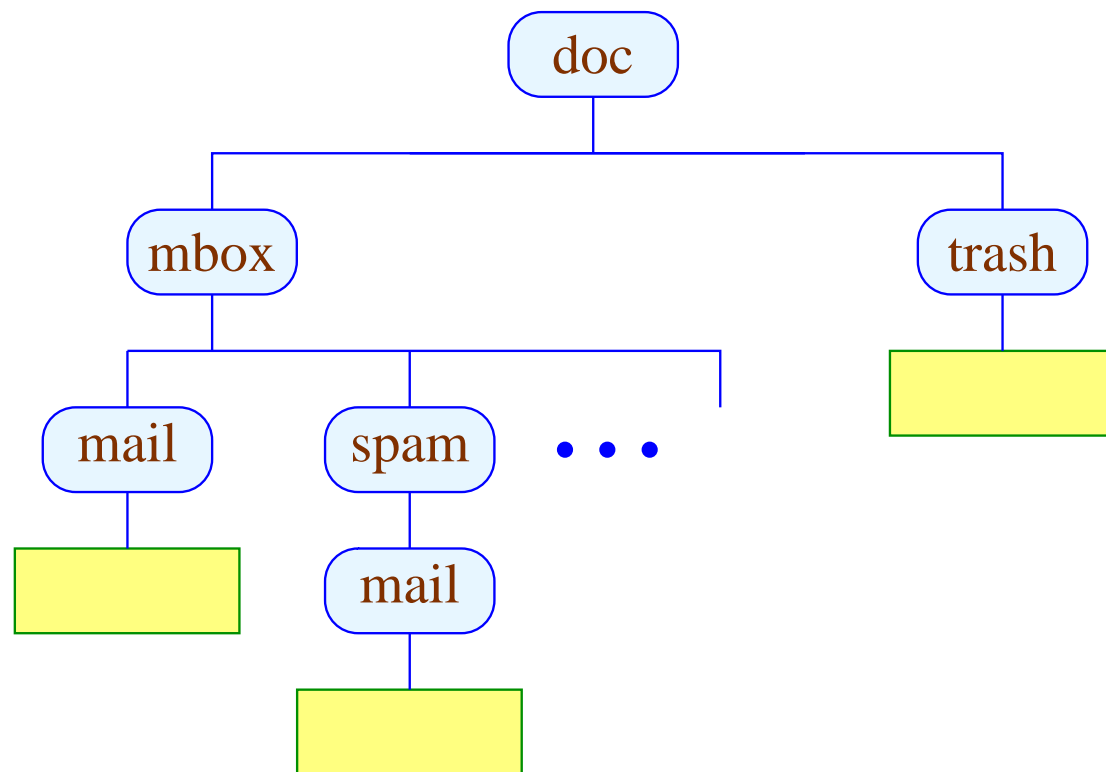
XML Document:

```
<mbox>
  <mail>
    <sender> Hans Dampf </sender>
    <address> dampf@in.tum.de </address>
    <subject> CONFIDENTIAL </subject>
    <body> ... </body> </mail>
  <spam><mail> ...
    <subject> V.I.A.G.R.A. </subject>
    ... </mail></spam>
  ...
</mbox>
```

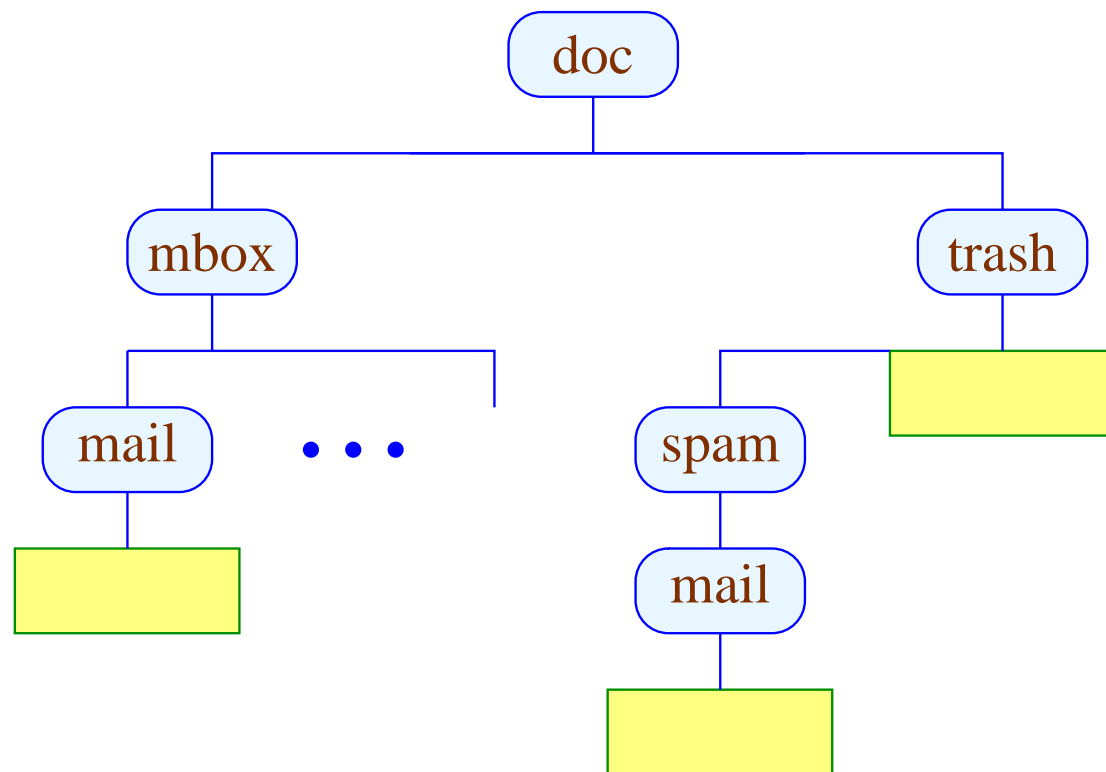
XML = unranked Trees:



XML Transformation:



XML Transformation:



Type Checking for Transformation \mathcal{T} :

Given: Recognizable sets I, O of admissible input and output documents;

Check: Does $\mathcal{T}(I) \subseteq O$ hold ?

Type Checking for Transformation \mathcal{T} :

Given: Recognizable sets I, O of admissible input and output documents;

Check: Does $\mathcal{T}(I) \subseteq O$ hold ?

Warning:

In general, $\mathcal{T}(I)$ is not recognizable ...

Common Transformation Languages:

XSLT 2.0

W3C, 2002

fxt

A. Berlea, 2001

XDuce

H. Hosoya, B.C. Pierce, 2003

Common Transformation Languages:

XSLT 2.0

W3C, 2002

no type checking :-()

fxt

A. Berlea, 2001

no type checking :-()

XDuce

H. Hosoya, B.C. Pierce, 2003

approximate checking only :-{

Ingredients of Transformation Languages:

Match Patterns: Which rule is applicable?

Select Patterns: Which node is transformed next?

Named Rules: Functions :-)

Accumulating Parameters

Key Observation 1:

Such transformations can adequately be described by

Stay Macro Tree Transducers ...

Fundamental properties of mtt's	Engelfriet, Vogler 1985
k -Pebble Tree Transducers	Milo, Suciu, Vianu 2000
Decomposition of k -ptt's	Engelfriet, Maneth 2003
Composition results	Voigtländer 2004
Decomposing XML transformations	Maneth, Perst, S. 2005

Key Observation 2:

Inverse **Stay Macro Tree Transducers** effectively preserve **recognizability**, i.e., for smtt M and recognizable O ,

$$M^{-1}(O) = \{t \in \mathcal{T} \mid M(t) \cap O \neq \emptyset\}$$

is again recognizable :-)

Engelfriet, Vogler 1985

Corollary

Type-checking is **decidable** for arbitrary compositions of
macro tree transducers :-)

Corollary

Type-checking is **decidable** for arbitrary compositions of
macro tree transducers :-)

... but:

The complexity is **a tower of exponentials** :-)

Wanted:

- Large **sub-classes** of (useful) transformations with **efficient type-checking**

Martens, Neven 2004

- **General algorithm** which is efficient for the efficient sub-cases :-)

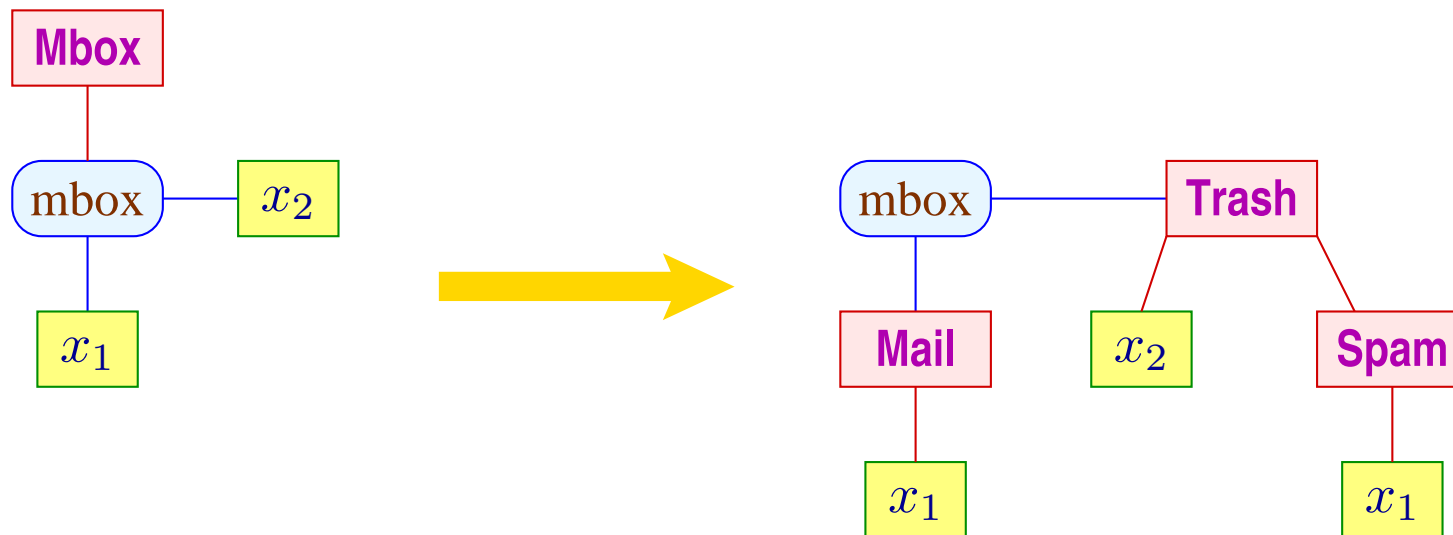
Outline:

- Stay Macro Tree Transducers
- Deciding emptiness
- Intersection construction
- Contextfree Tree Grammars
- *b*-bounded SMTTs
- Extensions and Summary

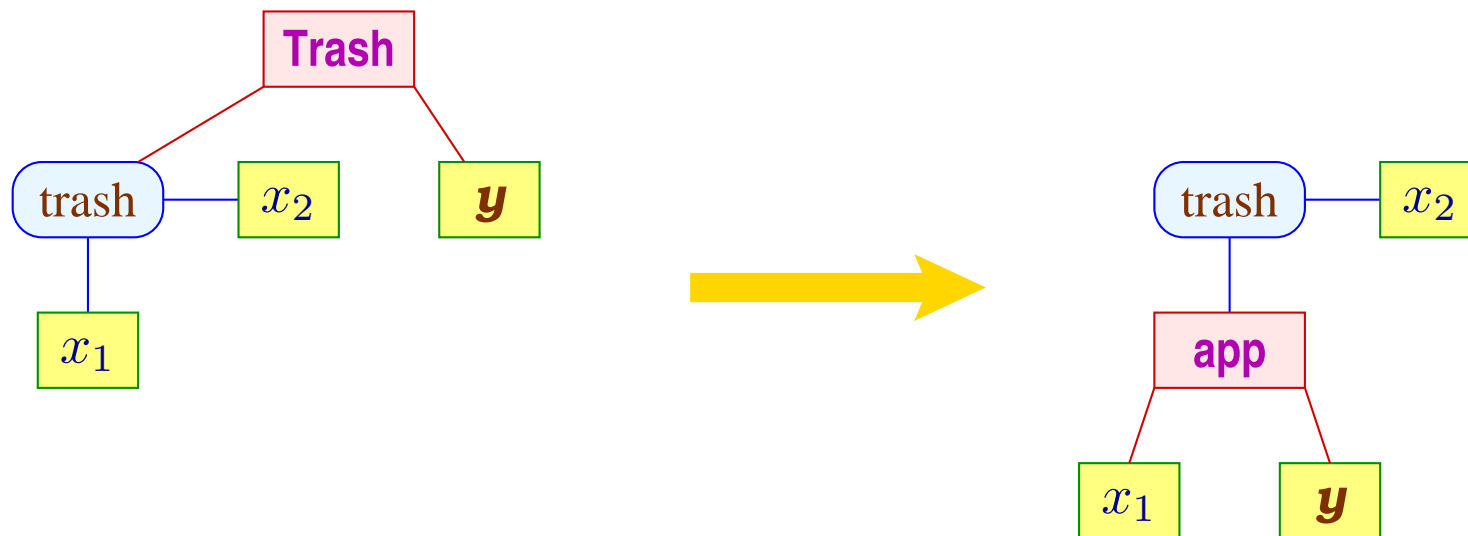
Stay Macro Tree Transducers

- First-order functional program;
- topdown traversal over the first argument;
- possible accumulation in extra parameters.

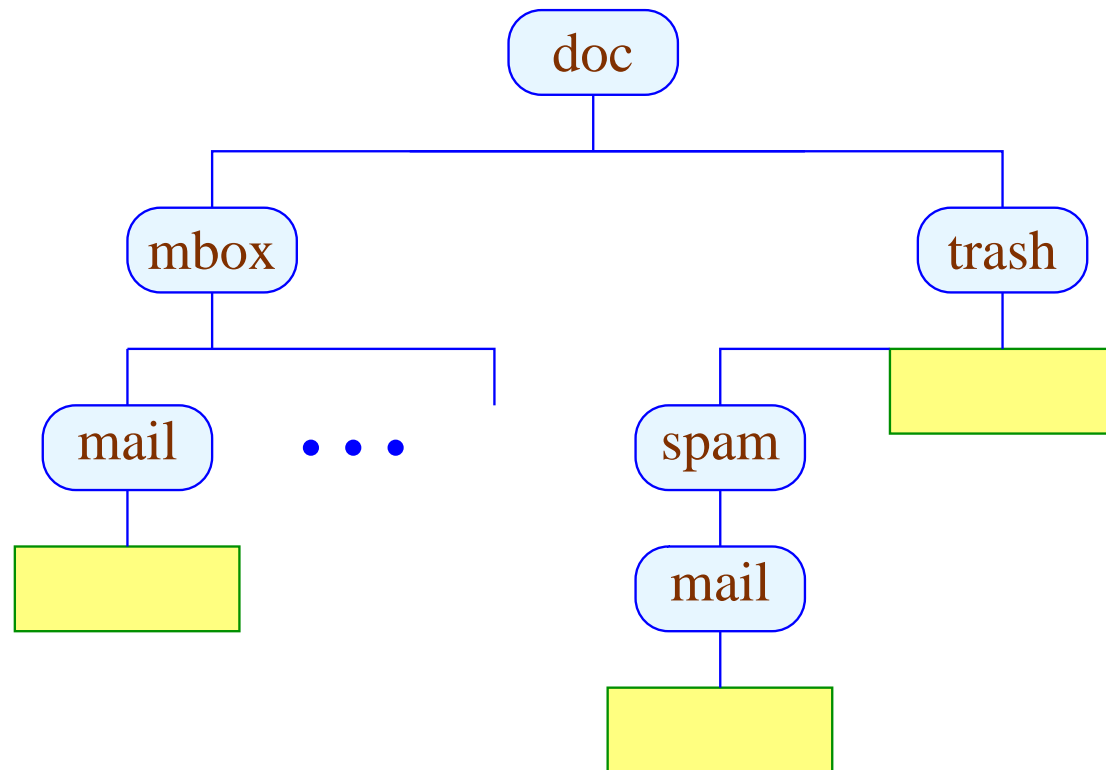
Our Example:



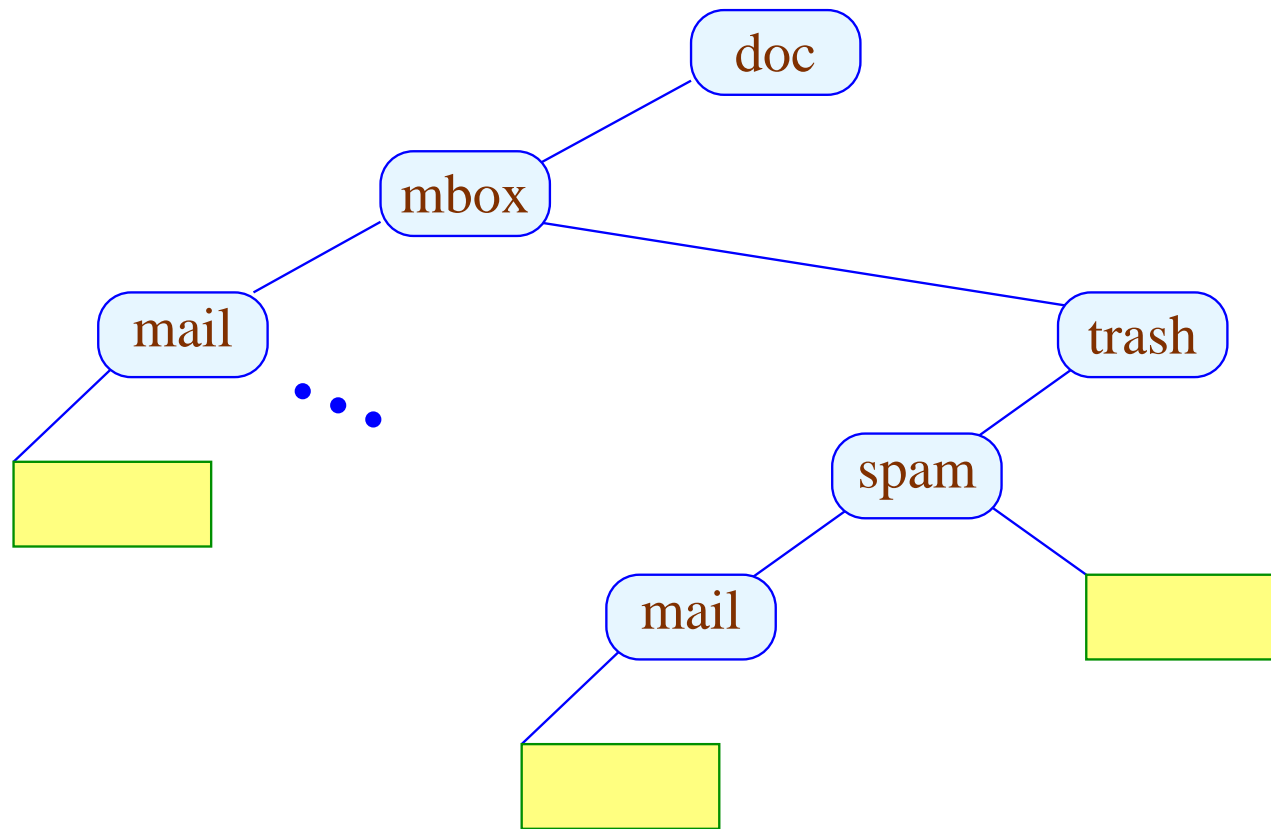
Our Example (cont.):



Unranked Trees = Binary Trees



Unranked Trees = Binary Trees



Formally, rules look as follows:

$$\begin{aligned} q(\langle \mathbf{a} \rangle x_1 \langle /\mathbf{a} \rangle x_2, y_1, \dots, y_k) &\rightarrow A \quad \text{or} \\ q(x_0, y_1, \dots, y_k) &\rightarrow A \quad \text{or} \\ q(\mathbf{e}, y_1, \dots, y_k) &\rightarrow A \end{aligned}$$

where

- q is a function name;
- x_1, x_2 are content and right context,
- y_1, \dots, y_k are the accumulating parameters;
- A is the action to be taken ...

Formally, rules look as follows:

$$q(\mathbf{a}(x_1, x_2), y_1, \dots, y_k) \rightarrow A \quad \text{or}$$

$$q(x_0, y_1, \dots, y_k) \rightarrow A \quad \text{or}$$

$$q(\mathbf{e}, y_1, \dots, y_k) \rightarrow A$$

where

- q is a function name;
- x_1, x_2 are content and right context,
- y_1, \dots, y_k are the accumulating parameters;
- A is the action to be taken ...

Possible actions are described by means of the grammar:

$$A ::= y_j \mid \mathbf{a}(A_1, A_2) \\ \mid \mathbf{e} \\ \mid q(x_i, A_1, \dots, A_k)$$

Semantics:

Operational: CBV, i.e., Inside-out Rewriting.

Denotational: Recursively defined Functions:

$$\llbracket q \rrbracket : \mathcal{T} \rightarrow 2^{\mathcal{T}(Y)}$$

Accidentally, both semantics coincide :-))

Outline:

- Stay Macro Tree Transducers
- Deciding emptiness
- Intersection construction
- Contextfree Tree Grammars
- b -bounded SMTTs
- Extensions and Summary

Properties:

- Smtts can be nondeterministic and **partial**.
- Smtt definable transformations are closed under recognizable restriction of the **input language**.

Idea: product construction :-)

- Emptyness is **DEXPTIME-complete ...**

Outline:

- Stay Macro Tree Transducers
- Deciding emptiness
- Intersection construction
- Contextfree Tree Grammars
- b -bounded SMTTs
- Extensions and Summary

Theorem:

Output languages of smtts are closed under intersection with recognizable sets, i.e.,

for smtt M and deterministic fta A ,

$$M_A(t) = M(t) \cap \mathcal{L}(A)$$

for some smtt M_A with

$$|M_A| \leq |M| \cdot |A|^{k+1+d}$$

d = max. number of states in a rhs.

k = max. number of accumulating pars.

Idea: (Generalized Triple Construction)

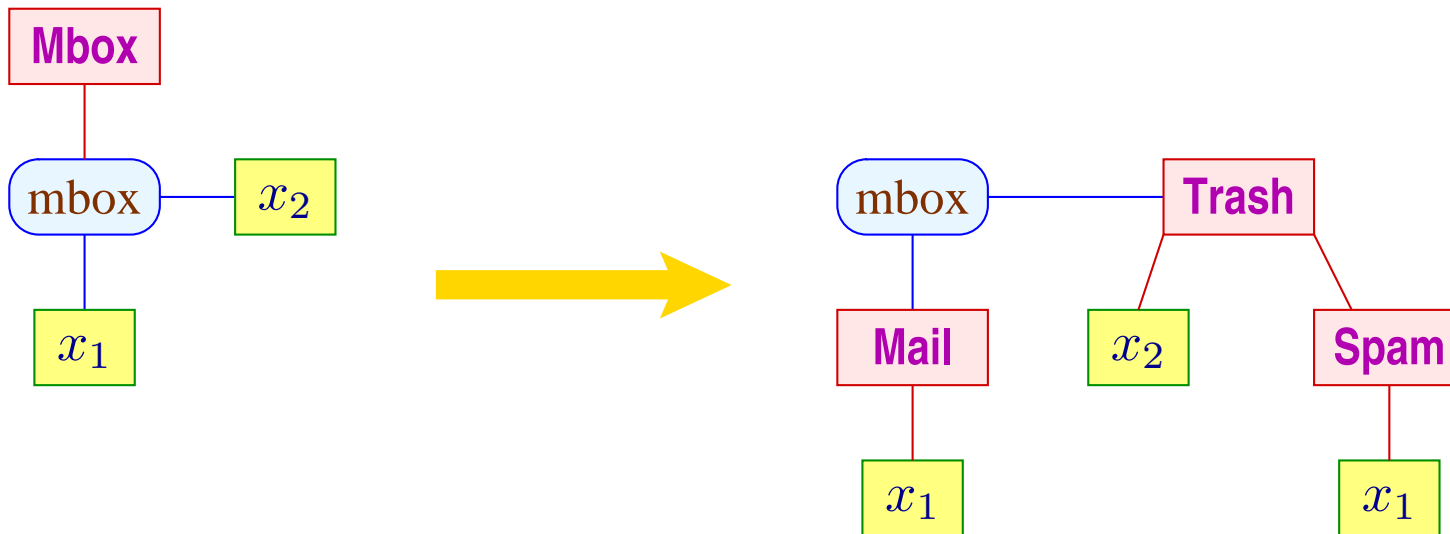
... new states $\langle q, p_0 \dots p_k \rangle$ with:

- q state of smtt M with k parameters;
- p_0, \dots, p_k states of dfta A

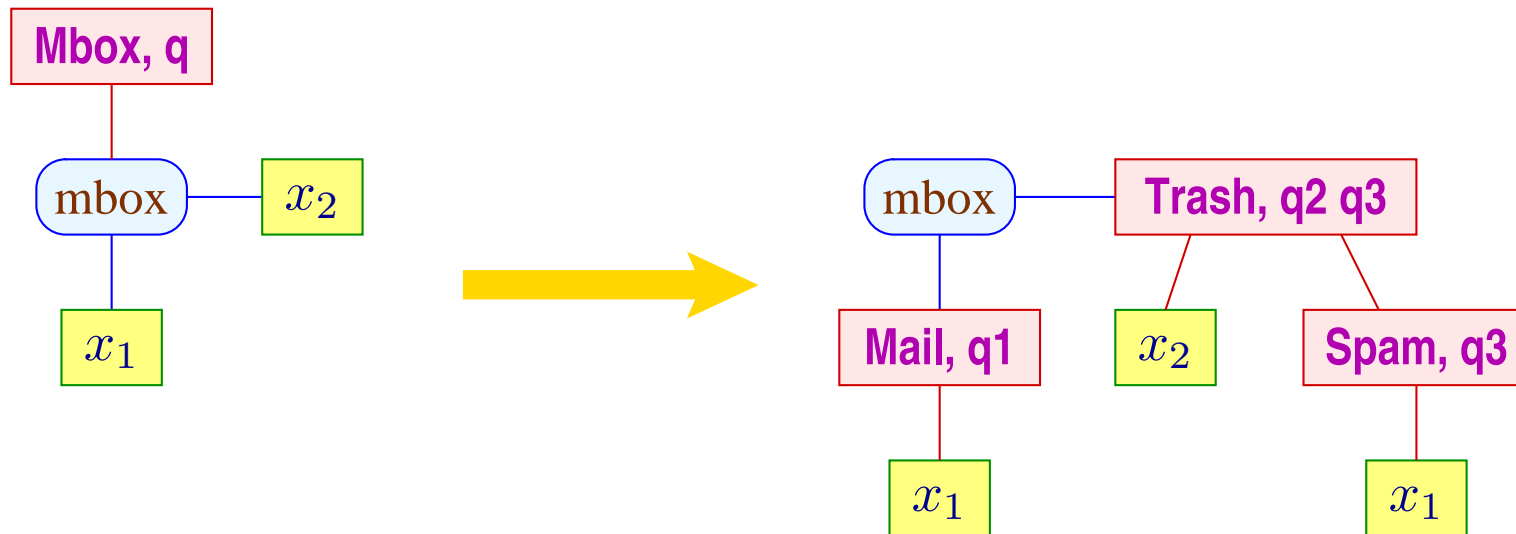
... such that:

$$\begin{aligned} \llbracket \langle q, p_0 \dots p_k \rangle (t) \rrbracket = \\ \{s \in \llbracket q \rrbracket (t) \mid \delta_A(s, p_1 \dots p_k) = p_0\} \end{aligned}$$

In our Example:



In our Example:



Remark:

- Combining emptiness procedure with construction for intersection gives a type checking algorithm :-)
- ... which is no better than pre-image computation :-)
- **Wanted:**
Subclasses where we can do better ...

Outline:

- Stay Macro Tree Transducers
- Deciding emptiness
- Intersection construction
- Contextfree Tree Grammars
- b -bounded SMTTs
- Extensions and Summary

Single-Use SMTTs:

Every x_i occurs at most **once** in a rhs **:-)**

Example:

$$\text{app}(@ (x_1, x_2), y) \rightarrow \text{app}(x_1, \text{app}(x_2, y))$$

$$\text{app}(a(x_1, x_2), y) \rightarrow a(\text{app}(x_1, e), \text{app}(x_2, y))$$

$$\text{app}(e, y) \rightarrow y$$

Observation:

For single-use mts, the output language is **context-free !!!**

Example:

$$\text{app}(@ (x_1, x_2), y) \rightarrow \text{app}(x_1, \text{app}(x_2, y))$$

$$\text{app}(a(x_1, x_2), y) \rightarrow a(\text{app}(x_1, e), \text{app}(x_2, y))$$

$$\text{app}(e, y) \rightarrow y$$

Observation:

For single-use mts, the output language is **context-free !!!**

Example:

$$\text{app}(\quad y) \rightarrow \text{app}(\quad \text{app}(\quad y))$$

$$\text{app}(\quad y) \rightarrow \text{a}(\text{app}(\quad \text{e}), \text{app}(\quad y))$$

$$\text{app}(\quad y) \rightarrow y$$

Proposition

- Emptiness for **contextfree** tree grammars (cftg's) is linear :-)
- The intersection construction can be organized such that only **useful** nonterminals are constructed :-))
- The output language of an mtt can be naturally **approximated** by a cftg :-)))

Idea:

Construct a **Datalog** program with predicates $q/(k + 1)$:

The rule:

$$q(y_1, y_2) \rightarrow \mathbf{a}(p_1(\mathbf{b}(p_2(y_1), y_2)), y_1)$$

gives rise to the clause:

$$\begin{aligned} q(Y_0, Y_1, Y_2) : - & \delta_A(Y_0, \mathbf{a}, X_1, Y_1), \\ & p_1(X_1, X_2), \\ & \delta_A(X_2, \mathbf{b}, X_3, Y_2), \\ & p_2(X_3, Y_1). \end{aligned}$$

Idea (cont.):

- **Topdown** answering the query

$$? - q_0(p_f).$$

// q_0 initial mtt state

// p_f accepting dfta state

computes all facts $q(p_0, \dots, p_k)$ where
 $\langle q, p_0 \dots p_k \rangle$ occurs in a terminal derivation :-)

- Efficient techniques for topdown querying are known ;-)

Outline:

- Stay Macro Tree Transducers
- Deciding emptiness
- Intersection construction
- Contextfree Tree Grammars
- *b*-bounded SMTTs
- Extensions and Summary

Generalization: Input-bounded SMTTs

Every sub-document is transformed at most b times.

Our Example:

Every subdocument is transformed at most **twice** :-)

Proposition:

- If M is b -bounded, then so is M_A :-)
- If M is b -bounded, emptiness is decidable in time

$$\mathcal{O}(|M|^b)$$

Theorem:

Assume M is b -bounded and A a dfta.

Then translation emptiness of M_A can be decided in time

$$\mathcal{O}(|M|^b \cdot |A|^{(k+3) \cdot b})$$

Practical Algorithm:

- Compute approximating cftg G .
- Compute set of useful nonterminals for intersection grammar G_A .
- If $\mathcal{L}(G_A) = \emptyset$ then return.

Otherwise, compute M_A and decide emptiness through **local fixpoint iteration** :-)

Outline:

- Stay Macro Tree Transducers
- Deciding emptiness
- Intersection construction
- Contextfree Tree Grammars
- b -bounded SMTTs
- Extensions and Summary

Extensions:

- direct support for concatenation, i.e.,

$$A ::= \dots \mid A_1 A_2; \mid \dots$$

\implies Stay Macro Forest Transducer

- representation of output types needed which is compatible with concatenations as well

\implies Forest Monoids

Summary:

- Ingredients of **adaptive** type-checking algorithm:
 - contextfree approximation;
 - generalized triple construction;
 - **top-down** querying Datalog;
 - **locally solving** Boolean systems.
- Algorithm behaves well on **common** transformations ;-)
- An implementation is on the way :-))