



THE COMPLEXITY OF TRANSLATION MEMBERSHIP FOR MACRO TREE TRANSDUCERS

Kazuhiro Inaba (The University of Tokyo)

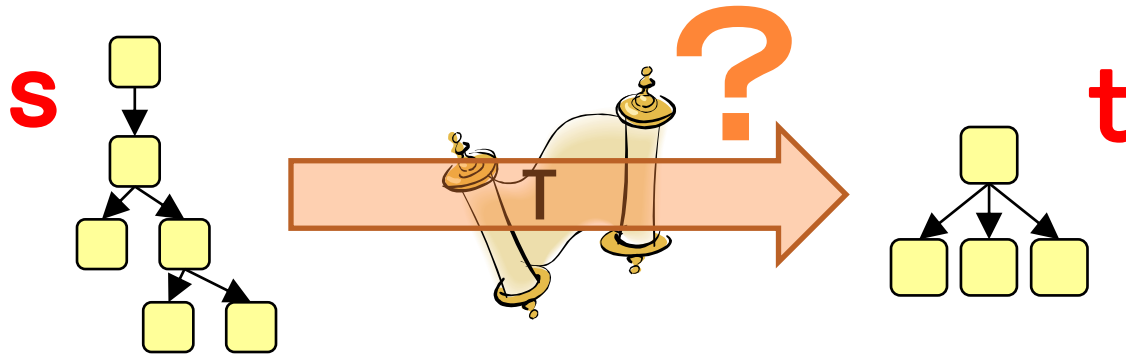
Sebastian Maneth (NICTA & University of New South Wales)

PLAN-X 2009, Savannah

TRANSLATION MEMBERSHIP?

- “Translation Membership Problem” for a tree-to-tree translation τ :

- Input: Two trees s and t
- Output: “YES” if τ translates s to t (“NO” otherwise)



- We are especially interested in **nondeterministic translations** where $\tau(s)$ is a *set* of trees (i.e., the translation membership problem asks “ $t \in \tau(s)$?”)



APPLICATIONS

- Dynamic assertion testing / Unit testing

```
assert(  
    run_my_xslt( load_xml("test-in.xml") )  
    == load_xml("test-out.xml") );
```

- How can we check the assertion efficiently?
- How can we check it when the translation depends on external effects (randomness, global options, or data from external DB...)?
 - → “Is there a configuration realizes the input/output pair?”
- Sub-problem of larger decision problems
 - Membership test for the domain of the translation
[Inaba&Maneth 2008]



KNOWN RESULTS ON COMPLEXITIES OF TRANSLATION MEMBERSHIP

- If τ is a Turing Machine
 - ⋯ Undecidable
- If τ is a finite composition of top-down/bottom-up tree transducers
 - ⋯ Linear space [Baker 1978]
 - Cubic Time [This Work]
- If τ is a finite composition of deterministic macro tree transducers
 - ⋯ Linear time (Easy consequence of [Maneth 2002])



OUTLINE

- Macro Tree Transducers (MTTs)
 - IO and OI -- Two Evaluation Strategies
- MTT_{OI} Translation Membership is NP-complete
 - ... also for finite compositions of MTT_{IO}/MTT_{OI} 's
- MTT_{IO} Translation Membership is in PTIME!!
 - ... also for several extensions of MTT_{IO} !!
- Conclusion and Open Problems



MACRO TREE TRANSDUCER (MTT)

$$\text{start}(A(x_1)) \quad \rightarrow \quad \text{double}(x_1, \text{double}(x_1, E))$$
$$\text{double}(A(x_1), y_1) \rightarrow \text{double}(x_1, \text{double}(x_1, y_1))$$
$$\text{double}(B, y_1) \quad \rightarrow \quad F(y_1, y_1)$$
$$\text{double}(B, y_1) \quad \rightarrow \quad G(y_1, y_1)$$

- An MTT $M = (Q, q_0, \Sigma, \Delta, R)$ is a set of first-order functions of type $\text{Tree}(\Sigma) * \text{Tree}(\Delta)^k \rightarrow \text{Tree}(\Delta)$
- Each function is inductively defined on the 1st parameter
 - Dispatch based on the label of the current node
 - Functions are applied only to the direct children of the current node
 - Not allowed to inspect other parameter trees



(M)TT IN THE XML WORLD

- Simulation of XSLT, XML-QL [Milo&Suciu&Vianu 2000]
 - Expressive fragment of XSLT and XML-QL can be represented as a composition of pebble tree transducers (which is a model quite related to macro tree transducers)
- TL – XML Translation Language [MBPS 2005]
 - A translation language equipping Monadic Second Order Logic as its query sub-language, representable by 3 compositions of MTTs.
- Exact Type Checking [MSV00, Tozawa 2001, Maneth&Perst&Seidl 2007, Frisch&Hosoya 2007, ...]
- Streaming [Nakano&Mu 2006]
- Equality Test [Maneth&Seidl 2007]
- ...



IO AND OI

```
double( A(x1), y1) → double( x1, double(x2, y1) )  
double( B, y1 ) → F( y1, y1 )  
double( B, y1 ) → G( y1, y1 )
```

- IO (inside-out / call-by-value):
evaluate the arguments first and then call the function

start(A(B)) → double(B, double(B, E))

→ double(B, F(E, E))

→ F(F(E, E), F(E, E))

or → G(F(E, E), F(E, E))

or

→ double(B, G(E, E))

→ F(G(E, E), G(E, E))

or → G(G(E, E), G(E, E))




IO AND OI

```
double( A(x1), y1) → double( x1, double(x2, y1) )
double( B, y1 ) → F( y1, y1 )
double( B, y1 ) → G( y1, y1 )
```

- OI (outside-in / call-by-name): call the function first and evaluate each argument when it is used

```
start(A(B)) → double( B, double(B, E) )
  → F( double(B, E), double(B, E) )
    → F( F(E,E), double(B, E) ) → F( F(E,E), F(E,E) )
      → F( F(E,E), G(E,E) )
    → F( G(E,E), double(B, E) ) → F( G(E,E), F(E,E) )
      → F( G(E,E), G(E,E) )
  → G( double(B, E), double(B, E) )
    → G( F(E,E), double(B, E) ) → G( F(E,E), F(E,E) )
      → G( F(E,E), G(E,E) )
    → G( G(E,E), double(B, E) ) → G( G(E,E), F(E,E) )
      → G( G(E,E), G(E,E) )
```



IO OR OI?

- Why we consider two strategies?

- IO is usually a more precise approximation of originally deterministic programs:

```
// f(A(x))    → if «complex_choice» then e1 else e2
f(A(x))      → e1
f(A(x))      → e2
g(A(x))      → h(x, f(x))
h(A(x), y)   → B(y, y)
```

- OI has better closure properties and a normal form:
 - For a composition sequence of OI MTTs, there exists a certain normal form with a good property, while not in IO. (explained later)



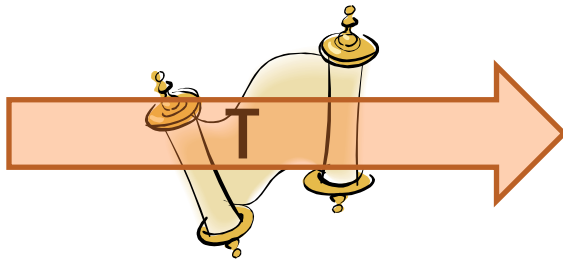
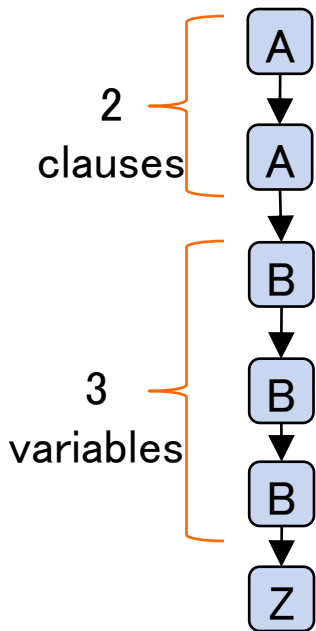
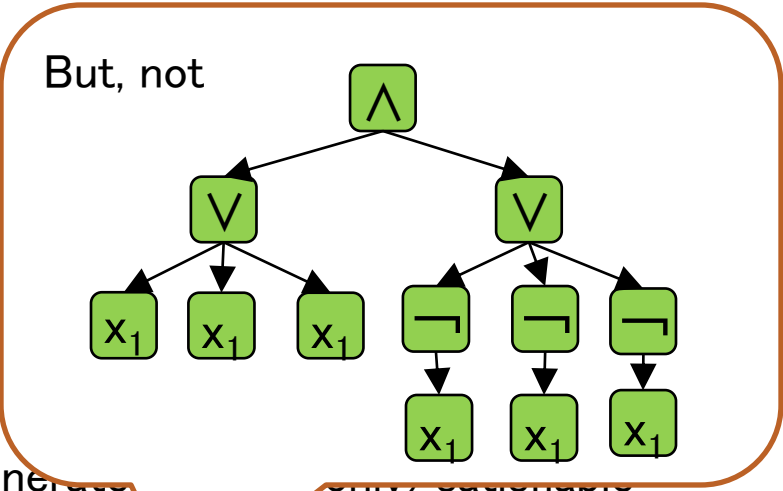
RESULTS



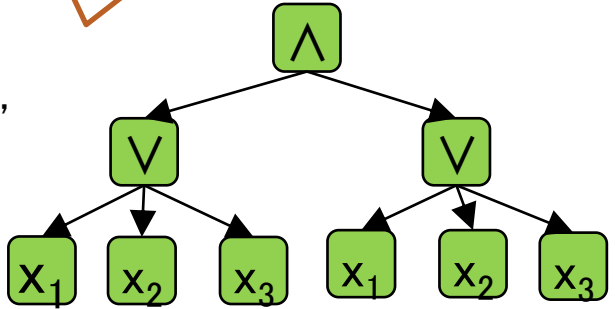
TRANSLATION MEMBERSHIP

MTT_{OI} Translation Membership

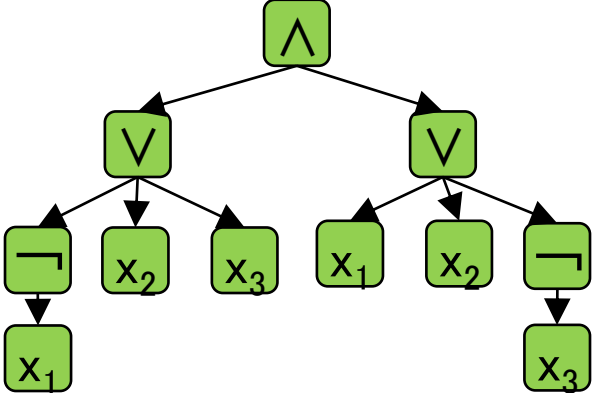
- Proof is by reduction from the
 - There is an MTT_{OI} translation that takes natural numbers (c and v), and generates a 3-CNF boolean formula with c clauses and v variables.



e.g.,



or



TRANSLATION MEMBERSHIP FOR MTT_{OI}

○ ‘Path-linear’ MTT_{OI} Translation Membership is in NP

[Inaba&Maneth 2008]

- Path-linear \Leftrightarrow No nested state calls to the same child node

```
f( A(x1, x2) ) → g(x1, g(x2, B)) // ok
f( A(x1, x2) ) → g(x1, g(x1, B)) // bad
f( A(x1, x2) ) → h(x1, g(x2, B), g(x2, C)) // ok
```

- Proof is by the “compressed representation”
 - The set τ (s) can be represented as a single “sharing graph” (generalization of a DAG) of size $O(|s|)$ [Maneth&Bussato 2004]
 - Navigation (up/1st child/next sibling) on the representation can be done in P only if the $MTT \tau$ is a path-linear.
- Corollary: MTT_{OI} Translation Membership is in NP
 - Proof is by the ‘Garbage-Free’ form in the next page...



K-COMPOSITIONS OF MTTs:

TRANSLATION MEMBERSHIP FOR MTT_{OI}^k AND MTT_{IO}^k

○ MTT_{OI}^k ($k \geq 1$) Translation Membership is NP-complete

- Proof is by the Garbage-Free Form [Inaba&Maneth 2008]

Any composition sequence of MTT_{OI} 's

$T = T_1 ; T_2 ; \dots ; T_k$ can be transformed to a
“Garbage-Free” sequence of **path-linear** MTT_{OI} 's

$T = \rho_1 ; \rho_2 ; \dots ; \rho_{2k}$ where for any (s,t) with $t \in T(s)$,
there exists intermediate trees

$s_1 \in \rho_1(s), s_2 \in \rho_2(s_1), \dots, t \in \rho_{2k}(s_{2k-1})$ such that $|s_i| \leq c |t|$

→ by NP-oracle we can guess all s_i 's

○ MTT_{IO}^k ($k \geq 2$) Translation Membership is NP-complete

- Proof is by Simulation between IO and OI [Engelfriet&Vogler 1985]

○ $MTT_{OI} \subseteq MTT_{IO} ; MTT_{IO}$ and $MTT_{IO} \subseteq MTT_{OI} ; MTT_{OI}$

MAIN RESULT:

TRANSLATION MEMBERSHIP FOR MTT_{IO}

- MTT_{IO} Translation Membership is in PTIME
(for an mtt with k parameters, $O(n^{k+2})$)
 - Proof is based on the Inverse Type Inference
[Engelfriet&Vogler 1985, Milo&Suciu&Vianu 2000]

For an MTT τ and a tree t , the inverse image $\tau^{-1}(t)$ is a regular tree language

- Instead of “ $t \in \tau(s)$ ”, check “ $s \in \tau^{-1}(t)$ ”
 - First, construct the bottom-up tree automaton recognizing $\tau^{-1}(t)$
 - Then, run the automaton on s .

PITFALL

The automaton may have $2^{|t|}$ states in the worst case.

PTIME SOLUTION

Do not fully instantiate the automaton. Run it while constructing it **on-the-fly**.



EXAMPLE (1)

$$\begin{array}{l}
 \circ \tau = \begin{array}{l}
 \text{st}(A(x_1)) \rightarrow \text{db}(x_1, \text{db}(x_2, E)) \\
 \text{db}(A(x_1), y_1) \rightarrow \text{db}(x_1, \text{db}(x_2, y_1)) \\
 \text{db}(B, y_1) \rightarrow F(y_1, y_1) \\
 \text{db}(B, y_1) \rightarrow G(y_1, y_1)
 \end{array}
 \end{array}
 \quad
 \begin{array}{l}
 s = A(B), \\
 t = F(G(E,E), G(E,E))
 \end{array}$$

State of the inverse-type automaton :: $\{\text{st}\} \cup (\{\text{db}\} \times V(t)) \rightarrow 2^{V(t)}$

We assign the state q_A such that:

$$\begin{array}{l}
 q_A(\text{st}) = q_B(\text{db}, q_B(\text{db}, E)) = q_B(\text{db}, \{G(E,E)\}) = \{F(G(E,E), G(E,E))\} \\
 q_A(\text{db}, E) = q_B(\text{db}, q_B(\text{db}, E)) = \{F(G(E,E), G(E,E))\} \\
 q_A(\text{db}, G(E,E)) = q_B(\text{db}, q_B(\text{db}, G(E,E))) = \{\} \\
 q_A(\text{db}, F(G(E,E), G(E,E))) = \dots = \{\}
 \end{array}$$

We assign the state q_B such that:

$$\begin{array}{l}
 q_B(\text{st}) = \{\} \\
 q_B(\text{db}, E) = \{G(E,E)\} \quad // F(E,E) \notin V(t) \\
 q_B(\text{db}, G(E,E)) = \{F(G(E,E), G(E,E))\} \quad // G(G,G) \notin V(t) \\
 q_B(\text{db}, F(G(E,E), G(E,E))) = \{\}
 \end{array}$$



EXAMPLE (2)

$$\begin{array}{l}
 \circ \tau = \begin{array}{l}
 \text{st}(A(x_1)) \rightarrow \text{db}(x_1, \text{db}(x_1, E)) \\
 \text{db}(A(x_1), y_1) \rightarrow \text{db}(x_1, \text{db}(x_1, y_1)) \\
 \text{db}(B, y_1) \rightarrow F(y_1, y_1) \\
 \text{db}(B, y_1) \rightarrow G(y_1, y_1)
 \end{array}
 \end{array}
 \quad
 \begin{array}{l}
 s = A(B), \\
 t = \underline{F(G(E,E), F(E,E))}
 \end{array}$$

State of the inverse-type automaton :: $\{\text{st}\} \cup (\{\text{db}\} \times V(t)) \rightarrow 2^{V(t)}$

• w

$$\begin{array}{l}
 q_A(\text{st}) = q_B(\text{db}, q_B(\text{db}, E)) = q_B(\text{db}, \{G(E,E), F(E,E)\}) = \{\} \\
 q_A(\text{db}, E) = q_B(\text{db}, q_B(\text{db}, E)) = \{\} \\
 q_A(\text{db}, G(E,E)) = q_B(\text{db}, q_B(\text{db}, G(E,E))) = \{\} \\
 q_A(\text{db}, F(E,E)) = q_B(\text{db}, q_B(\text{db}, F(E,E))) = \{\} \\
 q_A(\text{db}, F(G(E,E), G(E,E))) = \dots = \{\}
 \end{array}$$

$$\begin{array}{l}
 q_B(\text{st}) = \{\} \\
 q_B(\text{db}, E) = \{G(E,E), F(E,E)\} \\
 q_B(\text{db}, G(E,E)) = \{\} // F(G,G) \text{ and } G(G,G) \notin V(t) \\
 q_B(\text{db}, F(E,E)) = \{\} // F(F,F) \text{ and } G(F,F) \notin V(t) \\
 q_B(\text{db}, F(G(E,E), F(E,E))) = \{\} // \dots
 \end{array}$$

A

B

NOTE

○ Complexity:

- At each node of s , one function of type $\{st\} \cup (\{db\} \times V(t)) \rightarrow 2^{V(t)}$ is computed
- $\{st\} \cup (\{db\} \times V(t)) \rightarrow 2^{V(t)} \equiv 2^{V(t) \times (\{st\} \cup (\{db\} \times V(t)))}$
- Each function is of size $O(|V(t)|^2)$, which is computed per each node ($O(|s|)$ times) (and, computation of each entry of the function requires $O(|t|^2)$ time) $\rightarrow O(|s| |t|^4)$ time

- MTT_{OI} also has regular inverse image, but the inverse-type automaton may have $2^{2^{|t|}}$ many states in the worst case

\rightarrow Computing even a single state requires EXPTIME



SEVERAL EXTENSIONS

As long as the inverse type is sufficiently small, we can apply the same technique.

○ Variants of MTTs with PTIME Translation Membership

- MTT_{IO} with TAC-look-ahead
 - Rules are chosen not only by the label of the current node, but by a regular look-ahead and (dis)equality-check on child subtrees

$f(A(x_1, x_2))$	s.t. $x_1 \equiv x_2$	$\rightarrow C(f(x_1))$
$f(A(x_1, x_2))$	s.t. x_1 has even number of nodes	$\rightarrow D(f(x_1), f(x_2))$
$f(A(x_1, x_2))$	otherwise	$\rightarrow E(f(x_1), f(x_2))$

- Multi-Return MTT_{IO}
 - Each function can return multiple tree fragments (tuples of trees)

$f(A(x_1, x_2))$	$\rightarrow \text{let } (z_1, z_2) = g(x_1) \text{ in } D(z_1, C(z_2))$
$g(A(x_1, x_2))$	$\rightarrow (f(x_1), f(x_2))$

- Finite-copying MTT_{OI}
 - OI, but each parameter is copied not so many times.



CONCLUSION AND OPEN PROBLEMS

- Complexity of Translation Membership is
 - NP-complete for
 - MTT_{OI}^k ($k \geq 1$), MTT_{IO}^k ($k \geq 2$)
 - Higher-Order MTT, Macro Forest TT, ...
 - PTIME for
 - MTT_{IO} (+ look-ahead and multi-return)
- Open Problems
 - MTT_{OI} with at most one accumulating parameter
 - Our encoding of SAT used **3** parameters, which actually can be done with **2**. How about **1**?
 - MTT_{IO} with holes [Maneth&Nakano PLAN-X08]
 - It is an extension of IO MTTs, but has more complex inverse-type.



THANK YOU!

