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 $\tau$ :

- Input: Two trees $s$ and $t$
- Output: “YES” if $\tau$ 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

```plaintext
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 form 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 $\tau$ is a Turing Machine
  - ... Undecidable

- If $\tau$ is a finite composition of top-down/bottom-up tree transducers
  - ... Linear space [Baker 1978]
  - → Cubic Time [This Work]

- If $\tau$ is a finite composition of deterministic macro tree transducers
  - ... Linear time (Easy consequence of [Maneth 2002])
Macro Tree Transducers (MTTs)
  • IO and OI — Two Evaluation Strategies

MTT_{OI} Translation Membership is NP-complete
  • … also for finite compositions of \( \text{MTT}_{IO}/\text{MTT}_{OI} \) ’s

MTT_{IO} Translation Membership is in PTIME!!
  • … also for several extensions of \( \text{MTT}_{IO} \)!!

Conclusion and Open Problems
MACRO TREE TRANSDUCER (MTT)

An MTT \( M = (Q, q_0, \Sigma, \Delta, R) \) is a set of first-order functions of type \( \text{Tree}(\Sigma) \times \text{Tree}(\Delta)^k \rightarrow \text{Tree}(\Delta) \)

Each function is inductively defined on the 1\(^{\text{st}}\) 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

\begin{align*}
\text{start(} \ A(x_1) \ \text{)} & \rightarrow \ \text{double(} \ x_1, \ \text{double(} \ x_1, \ E) \ \text{)} \\
\text{double(} \ A(x_1), y_1 \ \text{)} & \rightarrow \ \text{double(} \ x_1, \ \text{double(} \ x_1, y_1) \ \text{)} \\
\text{double(} \ B, y_1 \ \text{)} & \rightarrow \ \text{F(} \ y_1, y_1 \ \text{)} \\
\text{double(} \ B, y_1 \ \text{)} & \rightarrow \ \text{G(} \ y_1, y_1 \ \text{)}
\end{align*}
(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

\[
\text{start}(A(B)) \Rightarrow \text{double}( B, \text{double}(B, E) ) \Rightarrow F( \text{double}(B, E), \text{double}(B, E) ) \Rightarrow F( F(E,E), \text{double}(B, E) ) \Rightarrow F( F(E,E), F(E,E) ) \Rightarrow F( F(E,E), G(E,E) ) \Rightarrow F( G(E,E), F(E,E) ) \Rightarrow F( G(E,E), G(E,E) ) \Rightarrow G( \text{double}(B, E), \text{double}(B, E) ) \Rightarrow G( F(E,E), \text{double}(B, E) ) \Rightarrow G( F(E,E), F(E,E) ) \Rightarrow G( F(E,E), G(E,E) ) \Rightarrow G( G(E,E), F(E,E) ) \Rightarrow G( G(E,E), G(E,E) )
\]
Why we consider two strategies?

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

```plaintext
// f(A(x)) → if "complex_choice" then e1 else e2
f(A(x)) → e1
f(A(x)) → e2
```

- 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\textsubscript{OI} Translation Membership**
  - Proof is by reduction from the 3-SAT problem.
    - There is an MTT\textsubscript{OI} translation that takes an input encoding two natural numbers (c and v), and generates all (and only) satisfiable 3-CNF boolean formulas with c clauses and v variables.
  - Example:
    - \( \text{e.g., } x_1 \lor x_2 \lor x_3 \lor \neg x_1 \lor \neg x_3 \lor x_2 \)
    - But, not
    - \( \text{or } x_1 \land x_2 \land x_3 \land \neg x_1 \land \neg x_3 \land x_2 \land x_1 \land x_2 \land \neg x_3 \land x_2 \land x_3 \land \neg x_2 \land \neg x_3 \land x_2 \)

- 2 clauses
  - A
    - A
      - B
        - B
          - Z

- 3 variables
  - A
    - A
      - B
        - B
          - B
            - Z
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

  \[
  \begin{align*}
  f( A(x_1, x_2) ) &\rightarrow g(x_1, g(x_2, B)) \quad \text{// ok} \\
  f( A(x_1, x_2) ) &\rightarrow g(x_1, g(x_1, B)) \quad \text{// bad} \\
  f( A(x_1, x_2) ) &\rightarrow h(x_1, g(x_2, B), g(x_2, C)) \quad \text{// ok}
  \end{align*}
  \]

  - Proof is by the “compressed representation”
    - The set \( \tau(s) \) can be represented as a single “sharing graph” (generalization of a DAG) of size \( O(|s|) \) [Maneth&Bussato 2004]
    - Navigation (up/1\(^{st}\) 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
  - $T = T_1 ; T_2 ; \cdots ; T_k$ can be transformed to a "Garbage-Free" sequence of path-linear $MTT_{OI}$’s
  - $T = \rho_1 ; \rho_2 ; \cdots ; \rho_{2k}$ where for any $(s,t)$ with $t \in T(s)$, there exists intermediate trees
    - $s_1 \in \rho_1(s)$, $s_2 \in \rho_2(s_1)$, …, $t \in \rho_{2k}(s_{2k-1})$ 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 \( \tau \) 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)

\( \mathbf{T} = \{ \text{st}( A(x_1) ) \rightarrow \text{db}( x_1, \text{db}(x_2, E) ) \}
\)

\( \text{db}( A(x_1), y_1 ) \rightarrow \text{db}( x_1, \text{db}(x_2, y_1) ) \)

\( \text{db}( B, y_1 ) \rightarrow F( y_1, y_1 ) \)

\( \text{db}( B, y_1 ) \rightarrow G( y_1, y_1 ) \)

\[ s = A(B), \quad t = F(G(E,E), G(E,E)) \]

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:
  \[
  \begin{align*}
  q_A(\text{st}) &= \emptyset \\
  q_A(\text{db}, E) &= q_B(\text{db}, q_B(\text{db}, E)) = \emptyset \\
  q_A(\text{db}, G(E,E)) &= q_B(\text{db}, q_B(\text{db}, G(E,E))) = \emptyset \\
  q_A(\text{db}, F(G(E,E), G(E,E))) &= \emptyset 
  \end{align*}
  \]

- We assign the state \( q_B \) such that:
  \[
  \begin{align*}
  q_B(\text{st}) &= \emptyset \\
  q_B(\text{db}, E) &= \{ G(E,E) \} \quad // F(E,E) \notin V(t) \\
  q_B(\text{db}, G(E,E)) &= \{ F(G(E,E), G(E,E)) \} \quad // G(G,G) \notin V(t) \\
  q_B(\text{db}, F(G(E,E), G(E,E))) &= \emptyset 
  \end{align*}
  \]
EXAMPLE (2)

- \( T = \{ \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)

s = A(B),
\text{t = } F(G(E,E), F(E,E))

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

\begin{align*}
q_A(\text{st}) &= q_B(\text{db}, q_B(\text{db}, E)) = q_B(\text{db}, \{ G(E,E), F(E,E) \}) = \emptyset \\
q_A(\text{db}, E) &= q_B(\text{db}, q_B(\text{db}, E)) = \emptyset \\
q_A(\text{db}, G(E,E)) &= q_B(\text{db}, q_B(\text{db}, G(E,E))) = \emptyset \\
q_A(\text{db}, F(E,E)) &= q_B(\text{db}, q_B(\text{db}, F(E,E))) = \emptyset \\
q_A(\text{db}, F(G(E,E), G(E,E))) &= \ldots = \emptyset \\
\end{align*}

\begin{align*}
q_B(\text{st}) &= \emptyset \\
q_B(\text{db}, E) &= \{ G(E,E), F(E,E) \} \\
q_B(\text{db}, G(E,E)) &= \emptyset \quad // F(G,G) \text{ and } G(G,G) \notin V(t) \\
q_B(\text{db}, F(E,E)) &= \emptyset \quad // F(F,F) \text{ and } G(F,F) \notin V(t) \\
q_B(\text{db}, F(G(E,E), F(E,E))) &= \emptyset \quad // \ldots
\end{align*}
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 \( (O(|s|) \text{ times}) \) (and, computation of each entry of the function requires \( O(|t|^2) \) time) \( \rightarrow O( |s| |t|^4 ) \) time

- \( \text{MTT}_{OI} \) also has regular inverse image, but the inverse-type automaton may have \( 2^2^{|t|} \) many states in the worst case
  - Computing even a single state requires EXPTIME
SEVERAL EXTENSIONS

- Variants of MTTs with PTIME Translation Membership
  - **MTT\textsubscript{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
    
    \[
    \begin{array}{ll}
    f( A(x_1, x_2) ) \text{ s.t. } x_1 \equiv x_2 & \rightarrow C( f(x_1) ) \\
    f( A(x_1, x_2) ) \text{ s.t. } x_1 \text{ has even number of nodes} & \rightarrow D( f(x_1), f(x_2) ) \\
    f( A(x_1, x_2) ) \text{ otherwise} & \rightarrow E( f(x_1), f(x_2) )
    \end{array}
    \]
  
  - **Multi-Return MTT\textsubscript{IO}**
    - Each function can return multiple tree fragments (tuples of trees)
    
    \[
    \begin{array}{l}
    f( A(x_1, x_2) ) \rightarrow \text{let } (z_1, z_2) = g(x_1) \text{ in } D( z_1, C(z_2) ) \\
    g( A(x_1, x_2) ) \rightarrow ( f(x_1), f(x_2) )
    \end{array}
    \]

  - **Finite-copying MTT\textsubscript{OI}**
    - OI, but each parameter is copied not so many times.

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

- Complexity of Translation Membership is
  - NP-complete for
    - $\text{MTT}_{0i}^k$ ($k \geq 1$), $\text{MTT}_{1o}^k$ ($k \geq 2$)
    - Higher-Order MTT, Macro Forest TT, ...
  - PTIME for
    - $\text{MTT}_{1o}$ (+ look-ahead and multi-return)

- Open Problems
  - $\text{MTT}_{0i}$ with at most one accumulating parameter
    - Our encoding of SAT used 3 parameters, which actually can be done with 2. How about 1?
  - $\text{MTT}_{1o}$ with holes [Maneth\&Nakano PLAN–X08]
    - It is an extension of IO MTTs, but has more complex inverse-type.
THANK YOU!