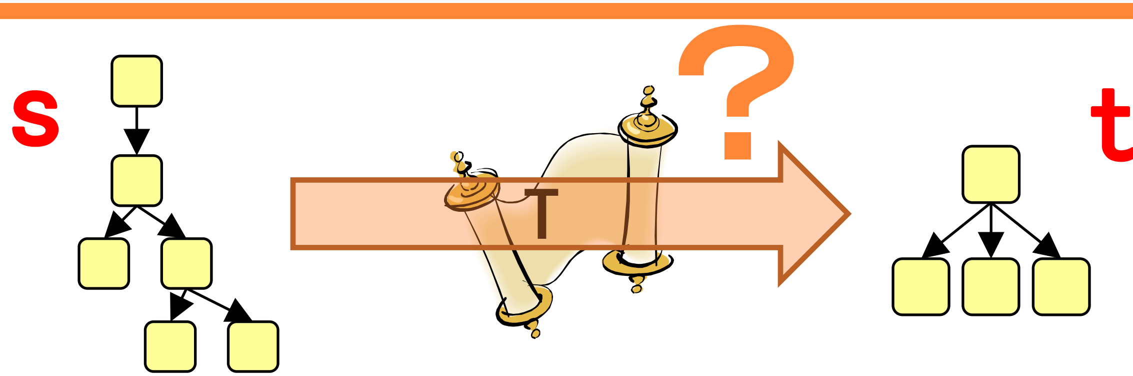
The Complexity of Translation Membership for Macro Tree Transducers

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 τ (s) is a *set* of trees
  (i.e., the translation membership problem asks "t ∈τ (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 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 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)

```
start( A(x₁) )      → double( x₁, double(x₁, E) )

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

- An MTT M = (Q, q_0, Σ , Δ , R) is a set of first-order functions of type Tree(Σ) * Tree(Δ)^k → Tree(Δ)
- 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:

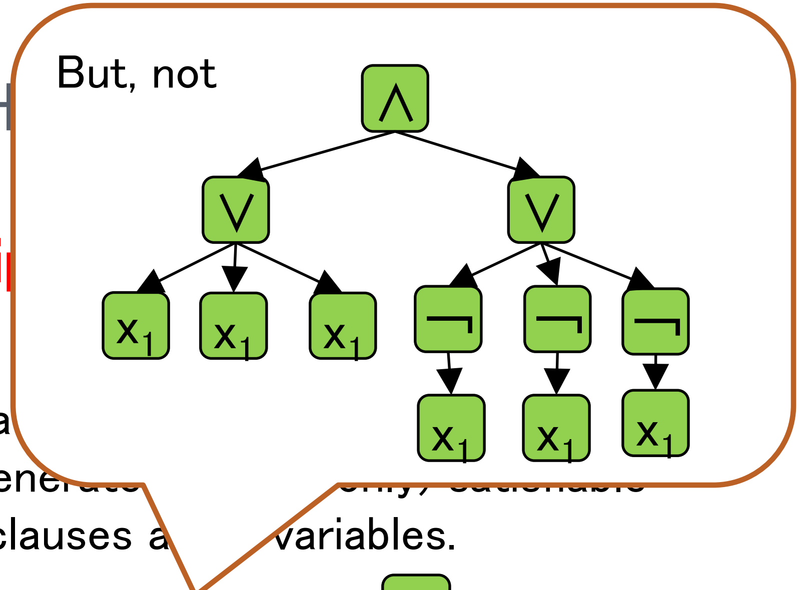
```
// f(A(x))    → if ≪complex_choice≫ then e1 else e2
f(A(x))       → e1
f(A(x))       → e2
g(A(x))       → h(x, f(x))
h(A(x), y)    → B(y, y)
```

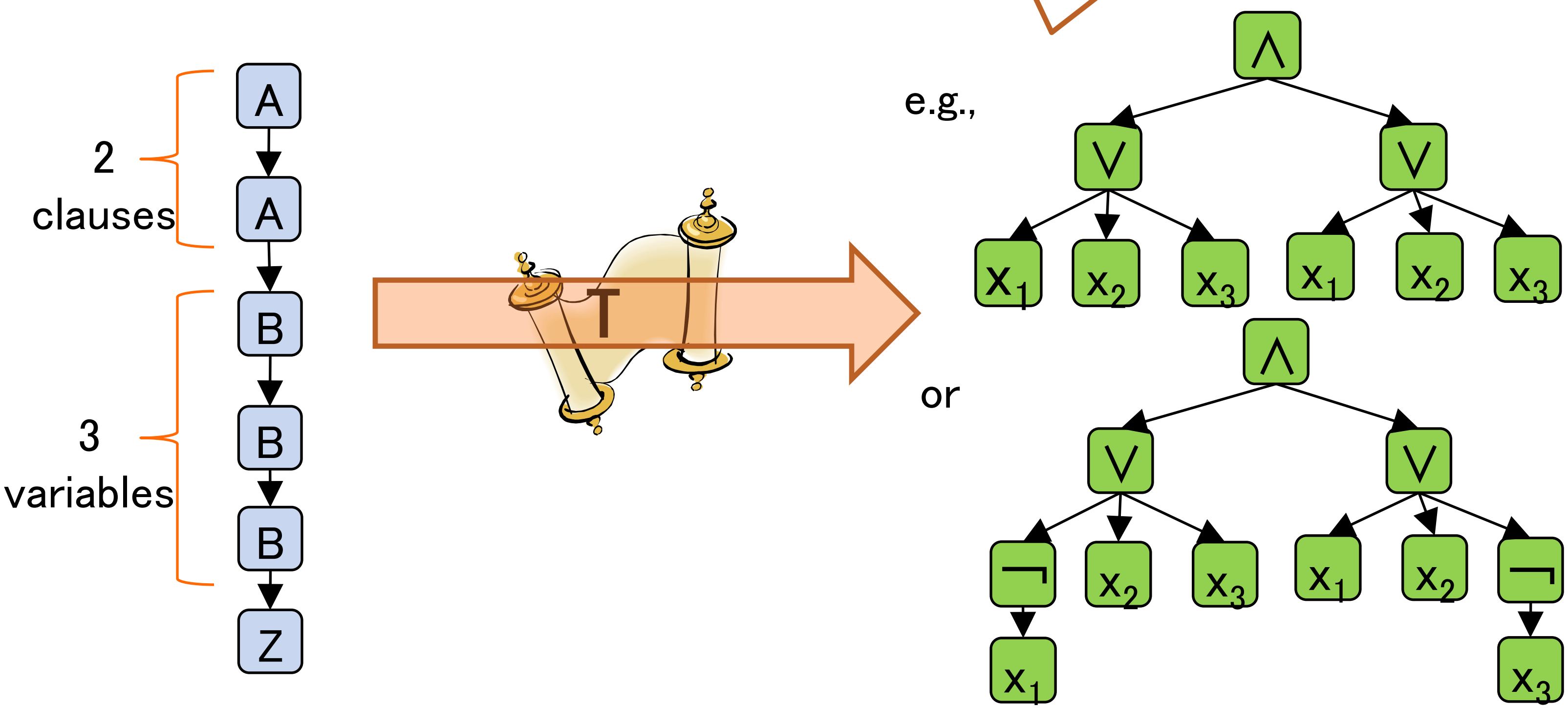
  - OI has better closure properties and a normal form:
    - For a composition sequence of OI MTTs, there exists a certain normal form with a good property, while not in IO. (explained later)

Results

TRANSLATION MEMBERSH

- **MTT$_{OI}$ Translation Membershi**
  - Proof is by reduction from the
    - There is an MTT$_{OI}$ translation tha natural numbers (c and v), and gener 3-CNF boolean formulas with c clauses a variables.





Translation Membership for MTT_{OI}

- **'Path-linear' MTT_{OI} Translation Membership is in NP**
  [Inaba&Maneth 2008]
  - Path-linear ⇔ 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 τ 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} ($k \geqq 1$) Translation Membership is NP-complete**
  - Proof is by the Garbage-Free Form [Inaba&Maneth 2008]

> **Any composition sequence of MTT_{OI}'s**
> **$\tau = \tau_1 ; \tau_2 ; \cdots ; \tau_k$ can be transformed to a**
> **"Garbage-Free" sequence of path-linear MTT_{OI}'s**
> **$\tau = \rho_1 ; \rho_2 ; \cdots ; \rho_{2k}$ where for any (s,t) with $t \in \tau(s)$,**
> **there exists intermediate trees**
> **$s_1 \in \rho_1(s), s_2 \in \rho_2(s_1), \cdots, t \in \rho_{2k}(s_{2k-1})$ such that $|s_i| \leqq c\,|t|$**

➔ by NP-oracle we can guess all s_i's

- **MTT_{IO}^{k} ($k \geqq 2$) 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.

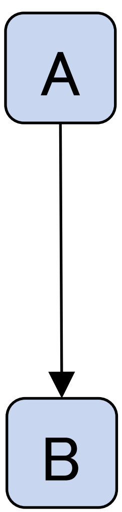
EXAMPLE (1)

- T =

```
st( A(x₁) )       → db( x₁, db(x₂, E) )
db( A(x₁), y₁)    → db( x₁, db(x₂, y₁) )
db( B, y₁ )       → F( y₁, y₁ )
db( B, y₁ )       → G( y₁, y₁ )
```

$s = A(B)$,
$t = F(G(E,E),\ G(E,E))$

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



We assign the state q_A such that:

$q_A\ (st) = q_B\,(db,\ q_B(db,E)) = q_B\,(db,\ \{G(E,E)\}) = \{F(G(E,E),\ G(E,E))\}$

$q_A\ (db,\ E) = q_B\,(db,\ q_B(db,E)) = \{F(G(E,E),\ G(E,E))\}$

$q_A\ (db,\ G(E,E)) = q_B\,(db,\ q_B(db,G(E,E))) = \{\}$

$q_A\ (db,\ F(G(E,E),\ G(E,E))) = \cdots = \{\}$

We assign the state q_B such that:

$q_B\ (st) = \{\}$

$q_B\ (db,\ E) = \{\ G(E,E)\ \}$ // $F(E,E) \notin V(t)$

$q_B\ (db,\ G(E,E)) = \{\ F(G(E,E),\ G(E,E))\ \}$ // $G(G,G) \notin V(t)$

$q_B\ (db,\ F(G(E,E),\ G(E,E))) = \{\}$

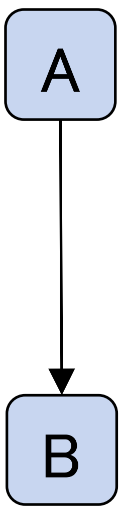
EXAMPLE (2)

- $T =$

$$
\begin{array}{lll}
st(\ A(x_1)\ ) & \to & db(\ x_1,\ db(x_1,\ E)\ ) \\
db(\ A(x_1),\ y_1) & \to & db(\ x_1,\ db(x_1,\ y_1)\ ) \\
db(\ B,\ y_1\ ) & \to & F(\ y_1,\ y_1\ ) \\
db(\ B,\ y_1\ ) & \to & G(\ y_1,\ y_1\ )
\end{array}
$$

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

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



$$
\begin{array}{lll}
q_A\ (st) = q_B\,(db,\ q_B(db,E)) & = q_B\,(db,\ \{G(E,E),F(E,E)\}) & = \{\} \\
q_A\ (db,\ E) & = q_B\,(db,\ q_B(db,E)) & = \{\} \\
q_A\ (db,\ G(E,E)) & = q_B\,(db,\ q_B(db,G(E,E))) & = \{\} \\
q_A\ (db,\ F(E,E)) & = q_B\,(db,\ q_B(db,F(E,E))) & = \{\} \\
q_A\ (db,\ F(G(E,E),\ G(E,E))) & = \cdots & = \{\}
\end{array}
$$

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

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 $2^\wedge 2^\wedge |t|$ 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**
  - $\mathrm{MTT}_{\mathrm{IO}}$ 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(x₁,x₂) ) s.t. x₁≡x₂              → C( f(x₁) )
f( A(x₁,x₂) ) s.t. x₁ has even number of nodes
                                      → D( f(x₁), f(x₂) )
f( A(x₁,x₂) ) otherwise               → E( f(x₁), f(x₂) )
```

  - Multi-Return $\mathrm{MTT}_{\mathrm{IO}}$
    - Each function can return multiple tree fragments (tuples of trees)

```
f( A(x₁,x₂) ) → let (z₁,z₂) = g(x₁) in D( z₁, C(z₂) )
g( A(x₁,x₂) ) → ( f(x₁), f(x₂) )
```

  - Finite-copying $\mathrm{MTT}_{\mathrm{OI}}$
    - 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} ($k \geqq 1$), MTT_{IO}^{k} ($k \geqq 2$)
    - 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!