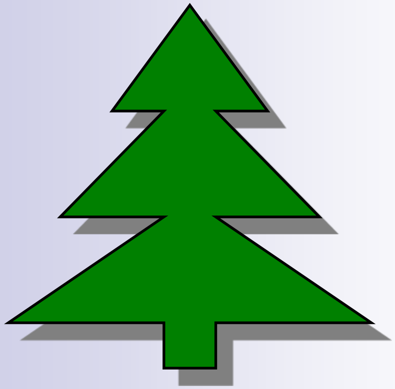
Multi-Return Macro Tree Transducers

CIAA 2008, San Francisco



The Univ. of Tokyo Kazuhiro Inaba

The Univ. of Tokyo Haruo Hosoya

NICTA, and UNSW Sebastian Maneth

Tree to Tree Translations

- Applications
  - Compiler
  - Natural Language Processing
  - XML Query/Translation
    - XSLT, XQuery, XDuce, …
  - …
- Models
  - Tree Transducer
    - Top-down / bottom-up
    - with/without lookahead …
  - Attributed Tree Transducer
  - MSO Tree Translation
  - Pebble Tree Transducer
  - Macro Tree Transducer
  - …
  - **Multi-Return Macro Tree Transducer**

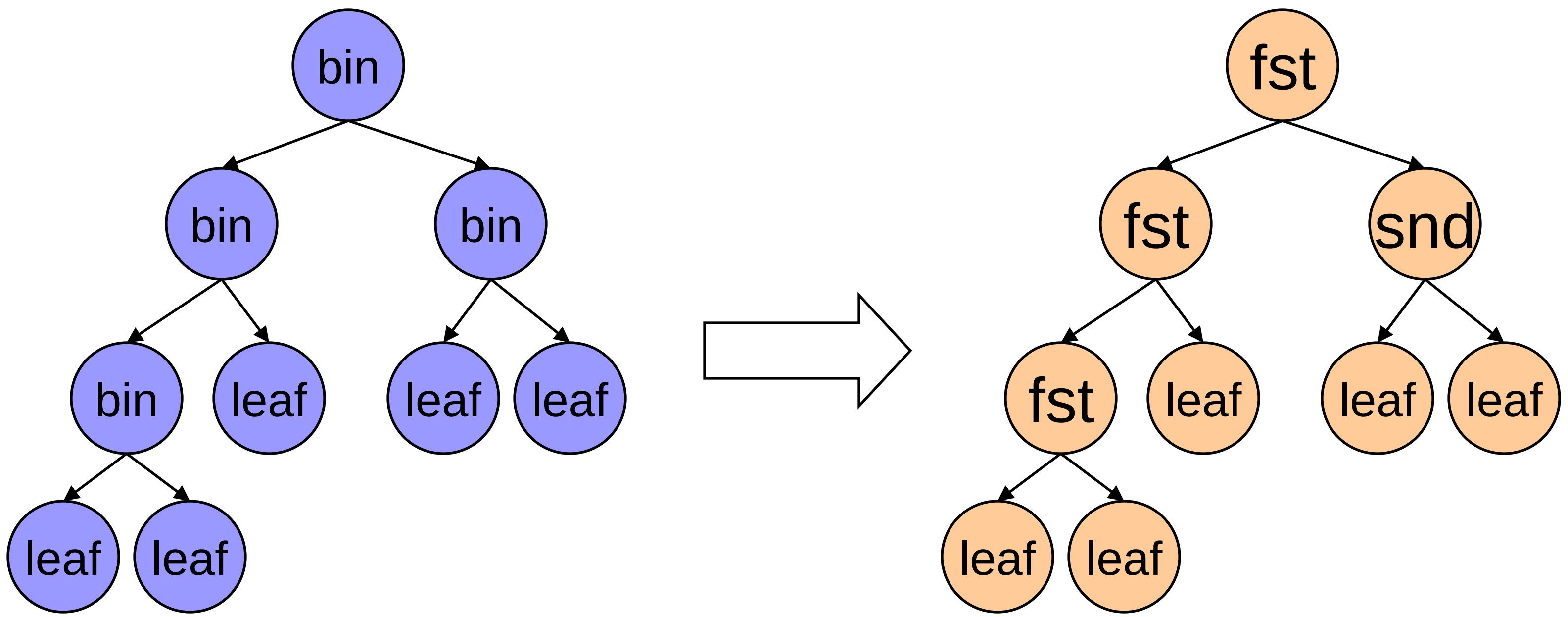
Models of Tree Translation

- Top-down **Tree Transducer**

  [Rounds 70, Thatcher 70]

  - Finite-state translation defined by structural (mutual) recursion on the input tree

$$
\begin{aligned}
&\langle q, \underline{\mathtt{bin}}(x_1,x_2)\rangle && \rightarrow \underline{\mathtt{fst}}(\, \langle q,x_1\rangle, \langle p,x_2\rangle \,) \\
&\langle q, \underline{\mathtt{leaf}}\rangle && \rightarrow \underline{\mathtt{leaf}} \\
\\
&\langle p, \underline{\mathtt{bin}}(x_1,x_2)\rangle && \rightarrow \underline{\mathtt{snd}}(\, \langle q,x_1\rangle, \langle p,x_2\rangle \,) \\
&\langle p, \underline{\mathtt{leaf}}\rangle && \rightarrow \underline{\mathtt{leaf}}
\end{aligned}
$$



$$\langle q, \underline{\mathtt{bin}}(x_1,x_2)\rangle \to \underline{\mathtt{fst}}(\, \langle q,x_1\rangle, \langle p,x_2\rangle \,)$$
$$\langle q, \underline{\mathtt{leaf}}\rangle \to \underline{\mathtt{leaf}}$$

$$\langle p, \underline{\mathtt{bin}}(x_1,x_2)\rangle \to \underline{\mathtt{snd}}(\, \langle q,x_1\rangle, \langle p,x_2\rangle \,)$$
$$\langle p, \underline{\mathtt{leaf}}\rangle \to \underline{\mathtt{leaf}}$$

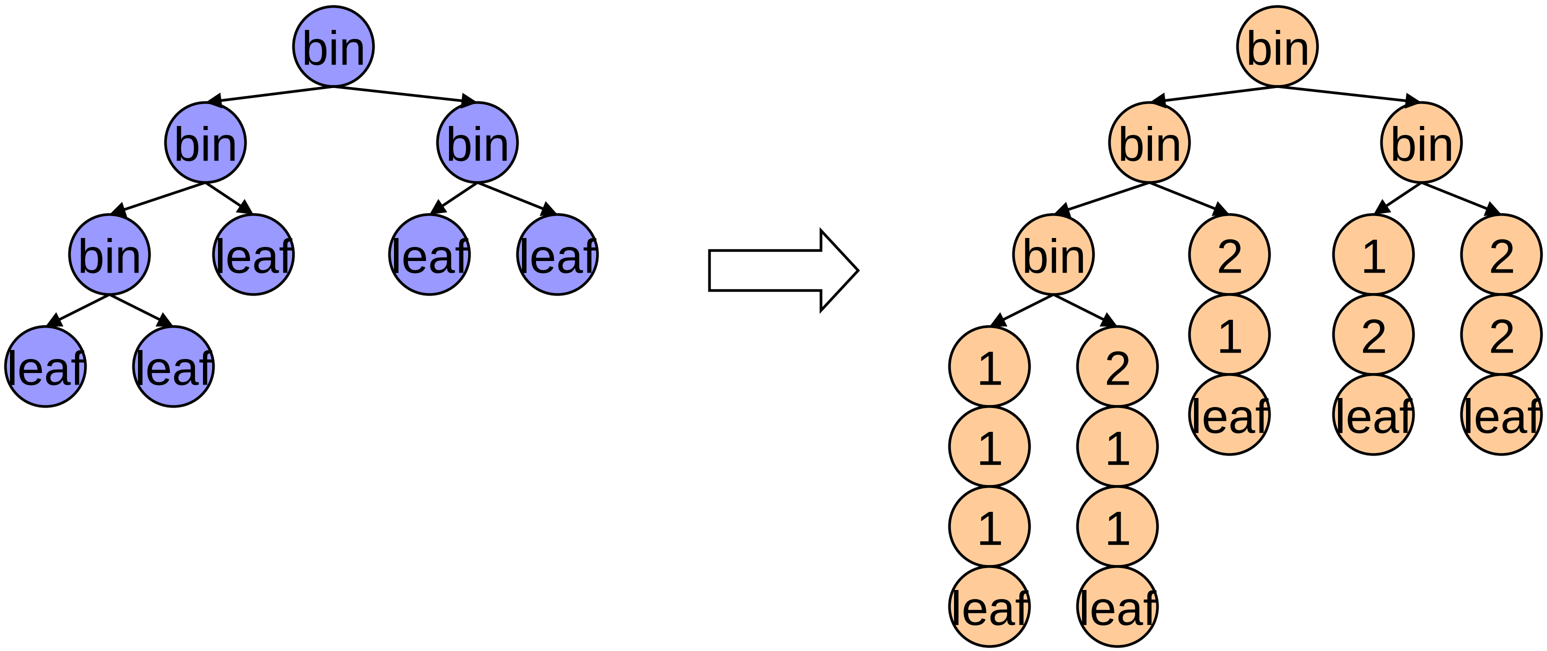
Models of Tree Translation

- Macro Tree Transducer (MTT)

  [Engelfriet 80, Courcell&Franchi-Zannettacci 82]

  - Tree Transducer + Context parameters
  - Strictly more expressive than tree transducers

$$
\begin{aligned}
&\langle q, \underline{\mathtt{bin}}(x_1,x_2)\rangle &&\rightarrow \underline{\mathtt{bin}}(\ \langle p,x_1\rangle(\underline{\mathtt{leaf}}),\langle p,x_2\rangle(\underline{\mathtt{leaf}})\ )\\
\\
&\langle p, \underline{\mathtt{bin}}(x_1,x_2)\rangle(y) &&\rightarrow \underline{\mathtt{bin}}(\ \langle p,x_1\rangle(\underline{1}(y)),\langle p,x_2\rangle(\underline{2}(y))\ )\\
&\langle p, \underline{\mathtt{leaf}}\rangle(y) &&\rightarrow y
\end{aligned}
$$



$\langle q, \underline{\mathtt{bin}}(x_1,x_2)\rangle \to \underline{\mathtt{bin}}(\ \langle p,x_1\rangle(\underline{\mathtt{leaf}}),\langle p,x_2\rangle(\underline{\mathtt{leaf}})\ )$

$\langle p, \underline{\mathtt{bin}}(x_1,x_2)\rangle(y_1) \to$
$\underline{\mathtt{bin}}(\ \langle p,x_1\rangle(\underline{1}(y_1)),\langle p,x_2\rangle(\underline{2}(y_1))\ )$
$\langle p, \underline{\mathtt{leaf}}\rangle(y_1) \to y_1$

Today's Topic

- Multi-Return Macro Tree Transducer

  [Inaba, Hosoya, and Maneth 08]

  - Macro Tree Transducer + Multiple return trees

```
<q, bin(x1,x2)>(y1) →    let (z1,z2) = <q,x1>(1(y1)) in
                         let (z3,z4) = <p,x2>(2(y1)) in
                           (bin(z1,z3), fst(z2,z4))
<q, leaf>(y1)       →    (leaf, y1)

<p, bin(x1,x2)>(y1) →    let (z1,z2) = <q,x1>(1(y1)) in
                         let (z3,z4) = <p,x2>(2(y1)) in
                           (bin(z1,z3), snd(z2,z4))
<p, leaf>(y1)       →    (leaf, y1)
```

Outline of the Talk

- Overview
- Definitions of MTTs and mr-MTTs
- Properties of mr-MTTs
  - Expressiveness
  - Closure under DtT composition
- Characterization of mr-MTTs

Definition of Macro Tree Transducers (MTTs)

- A MTT is a tuple $M = (Q, \Sigma, \Delta, q_0, R)$ where
  - Q : Ranked set of states (rank = # of parameters)
  - Σ : Ranked set of input alphabet
  - Δ : Ranked set of output alphabet
  - q_0 : Initial state of rank-0
  - R : Set of rules of the following form:

$$\langle q, \underline{\sigma}(x_1,\ldots,x_k)\rangle(y_1, \ldots, y_m) \rightarrow \mathrm{RHS}$$

$$\begin{aligned}\mathrm{RHS} ::= \; & \underline{\delta}(\ \mathrm{RHS}, \ldots, \mathrm{RHS}\ ) \\ \mid \; & \langle q', x_i\rangle(\ \mathrm{RHS}, \ldots, \mathrm{RHS}\ ) \\ \mid \; & y_i\end{aligned}$$

Definition of MTTs

- An MTT is
  - Deterministic if for every pair of $q \in Q$, $\underline{\sigma} \in \Sigma$, there exists at most one rule of the form $\langle q,\underline{\sigma}(\ldots)\rangle(\ldots) \rightarrow \ldots$
  - Nondeterministic otherwise
  - Total if there's at least one rule of the form $\langle q,\underline{\sigma}(\ldots)\rangle(\ldots) \rightarrow \ldots$ for each of them
  - Linear if in every right-hand side, each input variable x_i occurs at most once

Translation realized by MTTs

- The translation realized by M is

$$\tau_M = \{ (s,t) \in T_\Sigma \times T_\Delta \mid \langle q_0, s \rangle \Rightarrow^* t \}$$

where $\Rightarrow$ is the rewriting relation

  - By interpreting R as the set of rewrite rules
  - We consider only the Call-by-Value (Inside-Out) rewriting order in this work

Inside-Out (IO) Evaluation

- Example

$$<q_0,\ \underline{a}(x)> \rightarrow <q_1,x>(\ <q_2,x>\ )$$
$$<q_1,\ \underline{e}>(y) \rightarrow \underline{b}(y,\ y)$$
$$<q_2,\ \underline{e}> \rightarrow \underline{c}$$
$$<q_2,\ \underline{e}> \rightarrow \underline{d}$$

$$<q_0,\ \underline{a}(\underline{e})> \Rightarrow <q_1,\underline{e}>(\ <q_2,\underline{e}>\ ) \Rightarrow <q_1,\underline{e}>(\ \underline{c}\ ) \Rightarrow \underline{b}(\underline{c},\underline{c})$$
$$\Rightarrow <q_1,\underline{e}>(\ \underline{d}\ ) \Rightarrow \underline{b}(\underline{d},\underline{d})$$
$$\cancel{\Rightarrow \underline{b}(\ <q_2,\underline{e}>,\ <q_2,\underline{e}>\ )}$$

Why Nondeterminism and Why IO?

- IO-Nondeterminism in XML translation languages
  - In pattern matching (XDuce)
    - `match(e) with pat1 -> e1 | pat2 -> e2`
      - If e matches both pat1 and pat2, then it nondeterminisitically chooses e1 or e2
  - Approximation of Turing-complete languages (XSLT, …)
    - `if` (complicated-condition) `then e1 else e2`
      - (complicated-condition) may not be able to be modeled by MTTs

Multi-Return Macro Tree Transducer (mr-MTT)

- An mr-MTT is a tuple $M = (Q, \Sigma, \Delta, q_0, R)$ where
  - Q : Doubly ranked set of states (#params, #retvals)
  - Σ : Ranked set of input alphabet
  - Δ : Ranked set of output alphabet
  - q_0 : Initial state of rank (0, 1)
  - R : Set of rules of the following form:

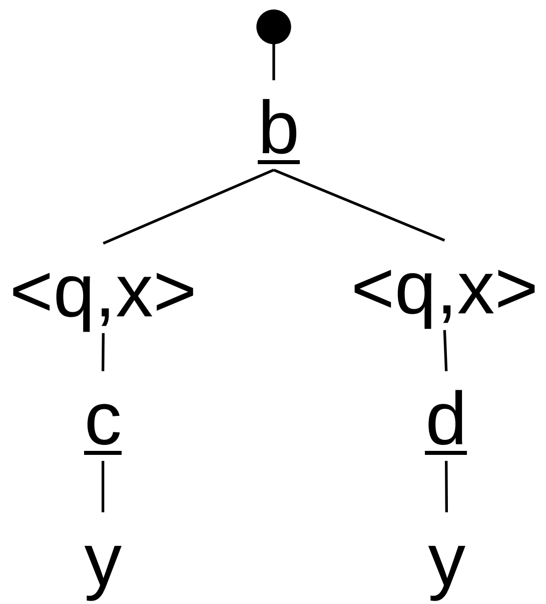
$$\langle q, \underline{\sigma}(x_1,\ldots,x_k)\rangle(y_1, \ldots, y_m) \rightarrow \mathrm{RHS}$$

$$\begin{aligned}
\mathrm{RHS} &::= \mathrm{LET}^{*}\ (\mathrm{TC}, \ldots, \mathrm{TC}) \\
\mathrm{LET} &::= \mathtt{let}\ (z_1,\ldots,z_n) = \langle q,x_i\rangle(\mathrm{TC}, \ldots, \mathrm{TC})\ \mathtt{in} \\
\mathrm{TC} &::= \underline{\delta}(\mathrm{TC}, \ldots, \mathrm{TC}) \mid y_i \mid z_i
\end{aligned}$$

MTT vs mr-MTT ≒ Tree vs DAG

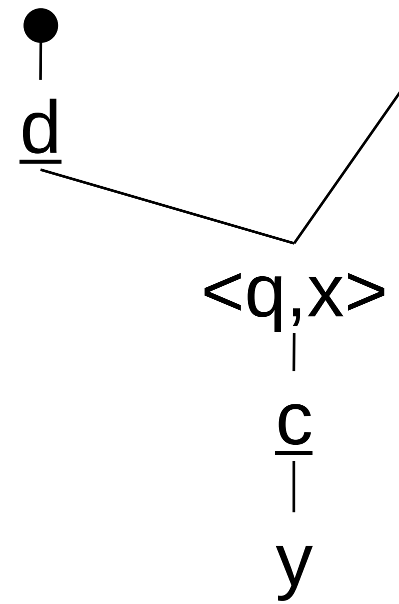
- MTT

```
<q, a(x)>(y) →
  b(<q,x>(c(y)), <q,x>(d(y)))
```



- mr-MTT

```
<q, a(x)>(y) →
 let (z1,z2) = <q,x>(c(y)) in
     (d(z1), z2)
```



Notations

- T : the class of translation realized by top-down TTs
- MT : the class of translations realized by MTTs
- MM : the class of translations realized by mr-MTTs
- *d*-MM (for $d \in N$) : the class of translations realizable by mr-MTTs whose return-tuples are at most length *d*

- Prefix D stands for "deterministic", t for "total", and L for "linear". E.g.,
  - DMT : the class of translations realized by deterministic MTTs
  - LDtT : the class of translations realized by linear deterministic total TTs

Good Properties of mr-MTTs

Expressiveness

- Question:
  Does the ‘multi-return’ feature really adds any power to MTTs?

- Answer:
  Yes, it does! (for nondetermistic MTTs)

Expressiveness of Det. Mr-MTT

- DMT = DMM (Corollary 5)

- Intuition: State Splitting

  a state q returning n-tuple of trees

  $\doteqdot$

  n states $q_1 \ldots q_n$ where q_i returns the i-th component of the return value of q.

Expressiveness of Nondet. 1-MM

- MT $\subsetneq$ 1-MM (Proposition 12)

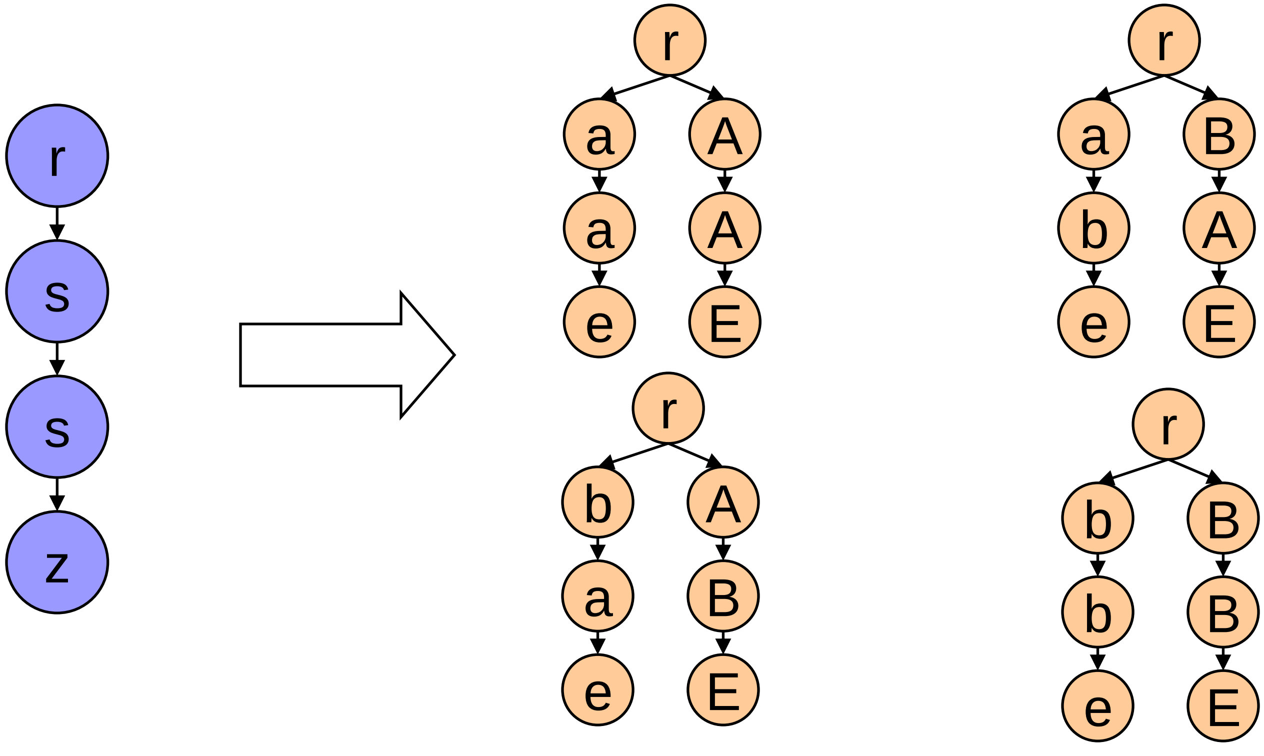
- Intuition: copying by 'let' variables adds some power

```
<q0,b(x1,x2)> →
     let z = <q,x1>(a,a) in
        <q,x2>( z, z )
```

```
<q0,b(x1,x2)> →
     <q,x2>( <q,x1>(a,a), <q,x1>(a,a) )
```

Expressiveness of 2-MM

- 1-MM ⊊ 2-MM (Theorem 13)
  - Witnessed by the ‘twist’ translation in the paper



Expressiveness of d-MM

- Conjecture
  - d-MM $\subsetneq$ (d+1)-MM for every d $\geqq$ 1

Closure under composition

- MTTs are very poor in composition:
  - LHOM ; MT $\not\subseteq$ MT
  - MT ; DtT $\not\subseteq$ MT

- For mr-MTTs:
  - DT ; MM $\subseteq$ MM
  - MM ; DtT $\subseteq$ MM (Theorem 11)

Proof Sketch

- DT ; MM ⊆ MM
  - Proof. Product construction
    P : the set of states of the DT
    Q : the set of states of the lhs MM
    → MM with set of states P × Q can simulate the composition (rules for the state (p,q) are obtained by ‘applying’ q to rules for p, in which we need variable-bindings by ‘let’).
- MM ; DtT ⊆ MM
  - Proof. (A variant of) product construction
    Q : states of lhs MM,   P : states of DtT
    → MM with set of states Q, where ranks of each q∈Q is multiplied by |P| (a state with *m* params & *d* retvals becomes *m*|P| params & *d*|P| retvals).

Characterization of mr-MTTs

- Question:
  How precisely powerful than MTTs?

- Answer:
  $MM \subseteq LHOM \,;\, MT \,;\, LDtT$
  - proven through two lemmas
    - $MM \subseteq 1\text{-}MM \,;\, LDtT$
    - $1\text{-}MM \subseteq LHOM \,;\, MT$

Characterization of mr-MTTs (Simulating multiple return values)

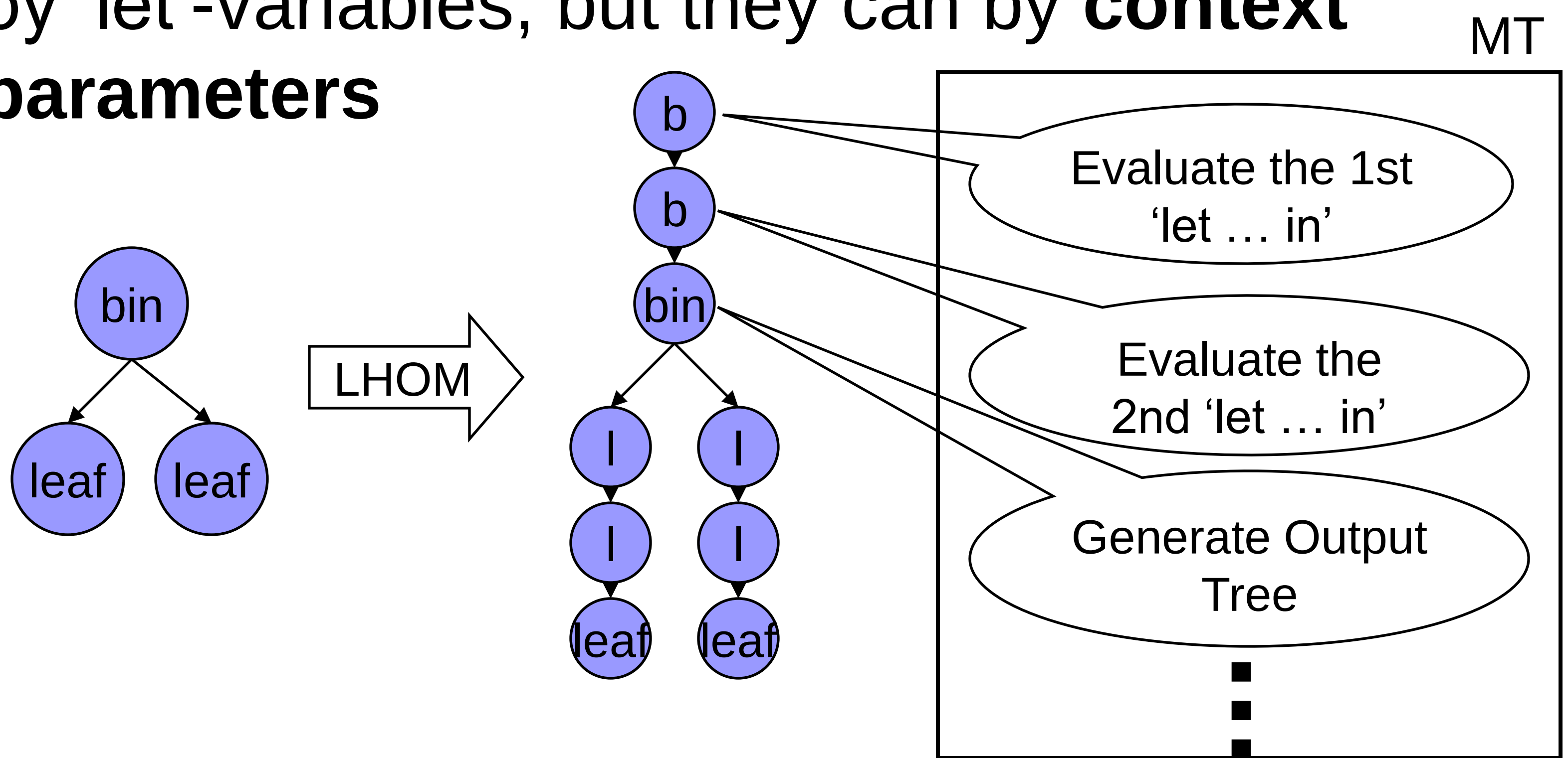
- MM $\subseteq$ 1-MM ; LDtT (Lemma 2)
  - Intuition: the 1-MM outputs symbolic representations of tupling and projection operations, and the LDtT carries them out

$$
\begin{array}{l}
\langle q,\ \underline{b}(x)\rangle \to \\
\quad \mathtt{let}\ (z_1, z_2) = \langle q, x\rangle\ \mathtt{in} \\
\qquad (\underline{a}(z_1),\ \underline{b}(z_2))
\end{array}
$$

$$
\begin{array}{l}
\langle q,\ \underline{b}(x)\rangle \to \\
\quad \mathtt{let}\ z = \langle q, x\rangle\ \mathtt{in} \\
\qquad \underline{\tau}(\underline{a}(\underline{1^{st}}(z)),\ \underline{b}(\underline{2^{nd}}(z)))
\end{array}
$$

Characterization of mr-MTTs (Simulating ‘let’-variable bindings)

- 1-MM ⊆ LHOM ; MT   (Lemma 3)
  - Intuition: MTTs cannot bind and copy trees by ’let’-variables, but they can by **context parameters**



Conclusion

- Multi-Return Macro Tree Transducers
  = Macro tree transducers with multiple return-values

- Expressiveness
  - DMT = DMM
  - MT $\subsetneq$ 1-MM $\subsetneq$ 2-MM
- Closure under Composition
  - DT ; MM $\subseteq$ MM
  - MM ; DtT $\subseteq$ MM
- Characterization
  - MM = LHOM ; MT ; LDtT