

# 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

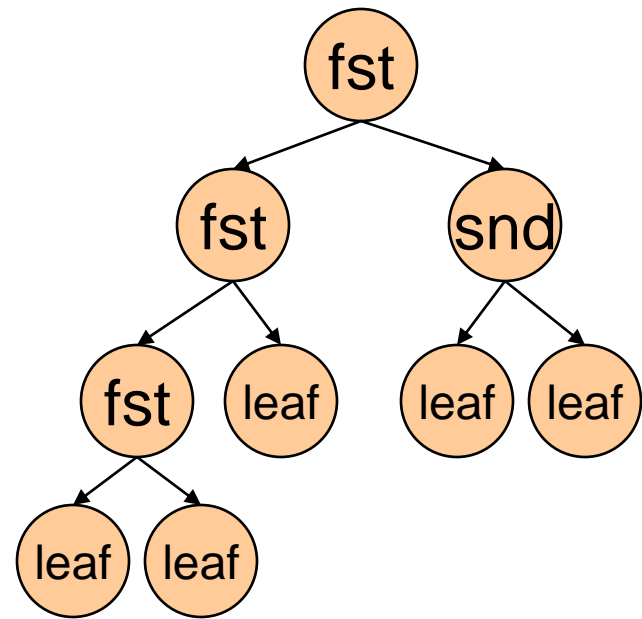
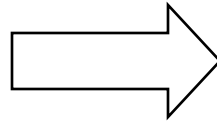
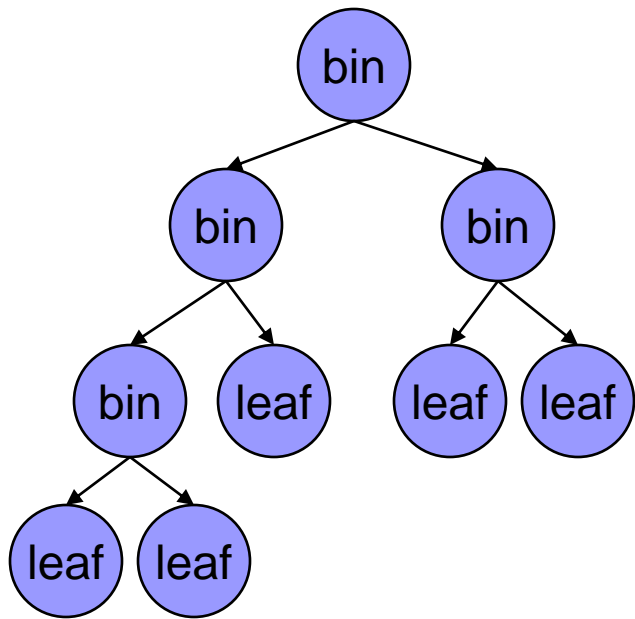
## ■ Top-down **Tree Transducer**

[Rounds 70, Thatcher 70]

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

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

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



$\langle q, \underline{\text{bin}}(x_1, x_2) \rangle \rightarrow \underline{\text{fst}}( \langle q, x_1 \rangle, \langle p, x_2 \rangle )$

$\langle q, \underline{\text{leaf}} \rangle \rightarrow \underline{\text{leaf}}$

$\langle p, \underline{\text{bin}}(x_1, x_2) \rangle \rightarrow \underline{\text{snd}}( \langle q, x_1 \rangle, \langle p, x_2 \rangle )$

$\langle p, \underline{\text{leaf}} \rangle \rightarrow \underline{\text{leaf}}$

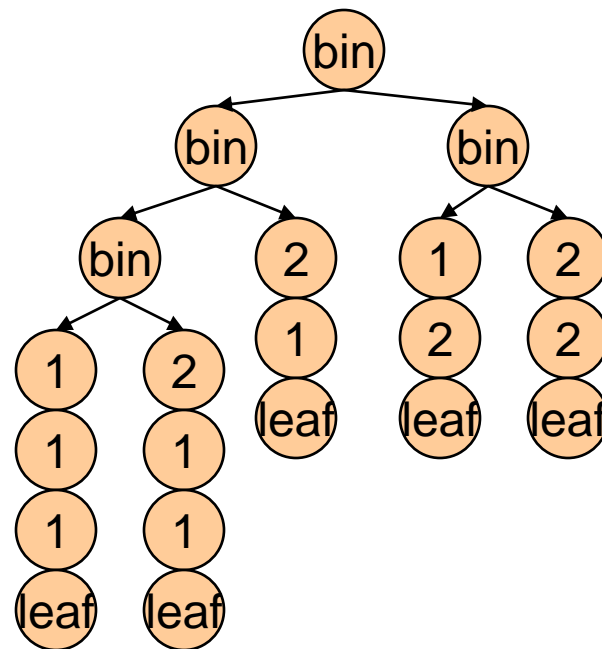
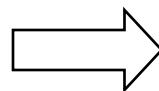
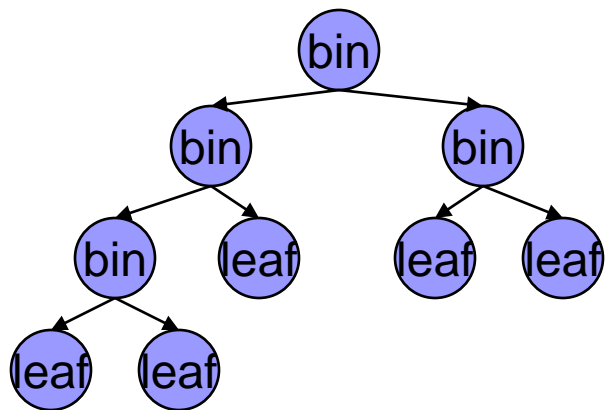
# Models of Tree Translation

## ■ Macro Tree Transducer (MTT)

[Engelfriet 80, Courcell&Franchi-Zannettacci 82]

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

$$\langle q, \underline{\text{bin}}(x_1, x_2) \rangle \rightarrow \underline{\text{bin}}( \langle p, x_1 \rangle (\underline{\text{leaf}}), \langle p, x_2 \rangle (\underline{\text{leaf}}) )$$
$$\langle p, \underline{\text{bin}}(x_1, x_2) \rangle (y) \rightarrow \underline{\text{bin}}( \langle p, x_1 \rangle (\underline{1}(y)), \langle p, x_2 \rangle (\underline{2}(y)) )$$
$$\langle p, \underline{\text{leaf}} \rangle (y) \rightarrow y$$



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

$\langle p, \underline{\text{bin}}(x_1, x_2) \rangle (y_1) \rightarrow$   
 $\underline{\text{bin}}( \langle p, x_1 \rangle (\underline{1}(y_1)), \langle p, x_2 \rangle (\underline{2}(y_1)) )$

$\langle p, \underline{\text{leaf}} \rangle (y_1) \rightarrow y_1$

# Today's Topic

## ■ Multi-Return Macro Tree Transducer

[Inaba, Hosoya, and Maneth 08]

### □ Macro Tree Transducer + Multiple return trees

$\langle q, \underline{\text{bin}}(x_1, x_2) \rangle(y_1)$	$\rightarrow$	$\text{let } (z_1, z_2) = \langle q, x_1 \rangle(\underline{1}(y_1)) \text{ in}$ $\text{let } (z_3, z_4) = \langle p, x_2 \rangle(\underline{2}(y_1)) \text{ in}$ $(\underline{\text{bin}}(z_1, z_3), \underline{\text{fst}}(z_2, z_4))$
$\langle q, \underline{\text{leaf}} \rangle(y_1)$	$\rightarrow$	$(\underline{\text{leaf}}, y_1)$
$\langle p, \underline{\text{bin}}(x_1, x_2) \rangle(y_1)$	$\rightarrow$	$\text{let } (z_1, z_2) = \langle q, x_1 \rangle(\underline{1}(y_1)) \text{ in}$ $\text{let } (z_3, z_4) = \langle p, x_2 \rangle(\underline{2}(y_1)) \text{ in}$ $(\underline{\text{bin}}(z_1, z_3), \underline{\text{snd}}(z_2, z_4))$
$\langle p, \underline{\text{leaf}} \rangle(y_1)$	$\rightarrow$	$(\underline{\text{leaf}}, y_1)$



# 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)
  - $\Sigma$  : Ranked set of input alphabet
  - $\Delta$  : Ranked set of output alphabet
  - $q_0$  : Initial state of rank-0
  - $R$  : Set of rules of the following form:

$$\langle q, \underline{\sigma}(x_1, \dots, x_k) \rangle (y_1, \dots, y_m) \rightarrow \text{RHS}$$
$$\begin{aligned} \text{RHS} ::= & \underline{\delta}(\text{RHS}, \dots, \text{RHS}) \\ & | \langle q', x_i \rangle (\text{RHS}, \dots, \text{RHS}) \\ & | y_i \end{aligned}$$

# 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  $\langle q, \underline{\sigma}(\dots) \rangle(\dots) \rightarrow \dots$
- **Nondeterministic** otherwise
- **Total** if there's at least one rule of the form  $\langle q, \underline{\sigma}(\dots) \rangle(\dots) \rightarrow \dots$  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

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

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

$$\begin{aligned} \langle q_0, \underline{a}(x) \rangle &\rightarrow \langle q_1, x \rangle ( \langle q_2, x \rangle ) \\ \langle q_1, \underline{e} \rangle (y) &\rightarrow \underline{b}(y, y) \\ \langle q_2, \underline{e} \rangle &\rightarrow \underline{c} \\ \langle q_2, \underline{e} \rangle &\rightarrow \underline{d} \end{aligned}$$
$$\begin{aligned} \langle q_0, \underline{a}(\underline{e}) \rangle &\Rightarrow \langle q_1, \underline{e} \rangle ( \langle q_2, \underline{e} \rangle ) \Rightarrow \langle q_1, \underline{e} \rangle ( \underline{c} ) \Rightarrow \underline{b}(\underline{c}, \underline{c}) \\ &\Rightarrow \langle q_1, \underline{e} \rangle ( \underline{d} ) \Rightarrow \underline{b}(\underline{d}, \underline{d}) \\ &\Rightarrow \underline{b}( \langle q_2, \underline{e} \rangle, \langle q_2, \underline{e} \rangle ) \end{aligned}$$

# 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 nondeterministically 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)
  - $\Sigma$  : Ranked set of input alphabet
  - $\Delta$  : Ranked set of output alphabet
  - $q_0$  : Initial state of rank (0, 1)
  - $R$  : Set of rules of the following form:

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

**RHS** ::= **LET\*** (TC, ..., TC)

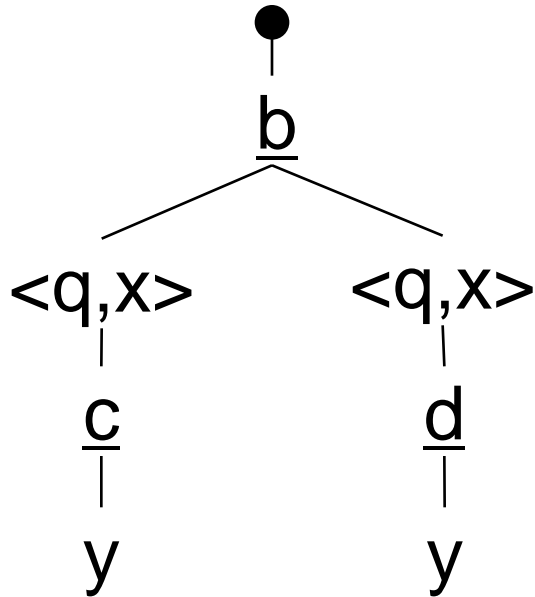
**LET** ::= let  $(z_1, \dots, z_n) = \langle q, x_i \rangle (\text{TC}, \dots, \text{TC})$  in

**TC** ::=  $\underline{\delta}(\text{TC}, \dots, \text{TC}) \mid y_i \mid z_i$

# MTT vs mr-MTT $\doteq$ Tree vs DAG

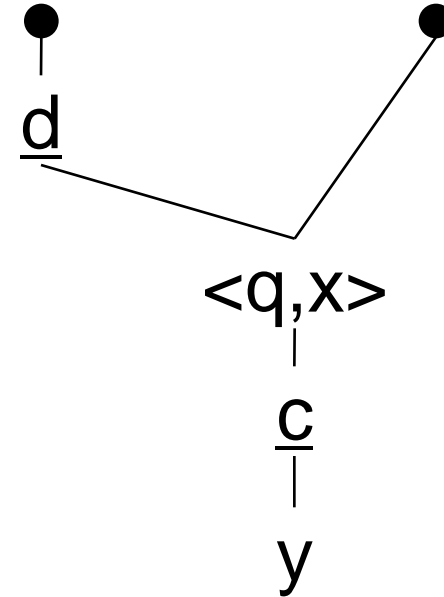
## ■ MTT

$\langle q, \underline{a}(x) \rangle(y) \rightarrow$   
 $\underline{b}(\langle q, x \rangle(\underline{c}(y)), \langle q, x \rangle(\underline{d}(y)))$



## ■ mr-MTT

$\langle q, \underline{a}(x) \rangle(y) \rightarrow$   
let  $(z_1, z_2) = \langle q, x \rangle(\underline{c}(y))$  in  
 $(\underline{d}(z_1), z_2)$



# 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  $d \in \mathbb{N}$ ) : 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 add any power to MTTs?

- Answer:

Yes, it does! (for nondeterministic MTTs)

# Expressiveness of Det. Mr-MTT

- $DMT = DMM$  (Corollary 5)

- Intuition: State Splitting

a state  $q$  returning  $n$ -tuple of trees

$\doteq$

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

# Expressiveness of Nondet. 1-MM

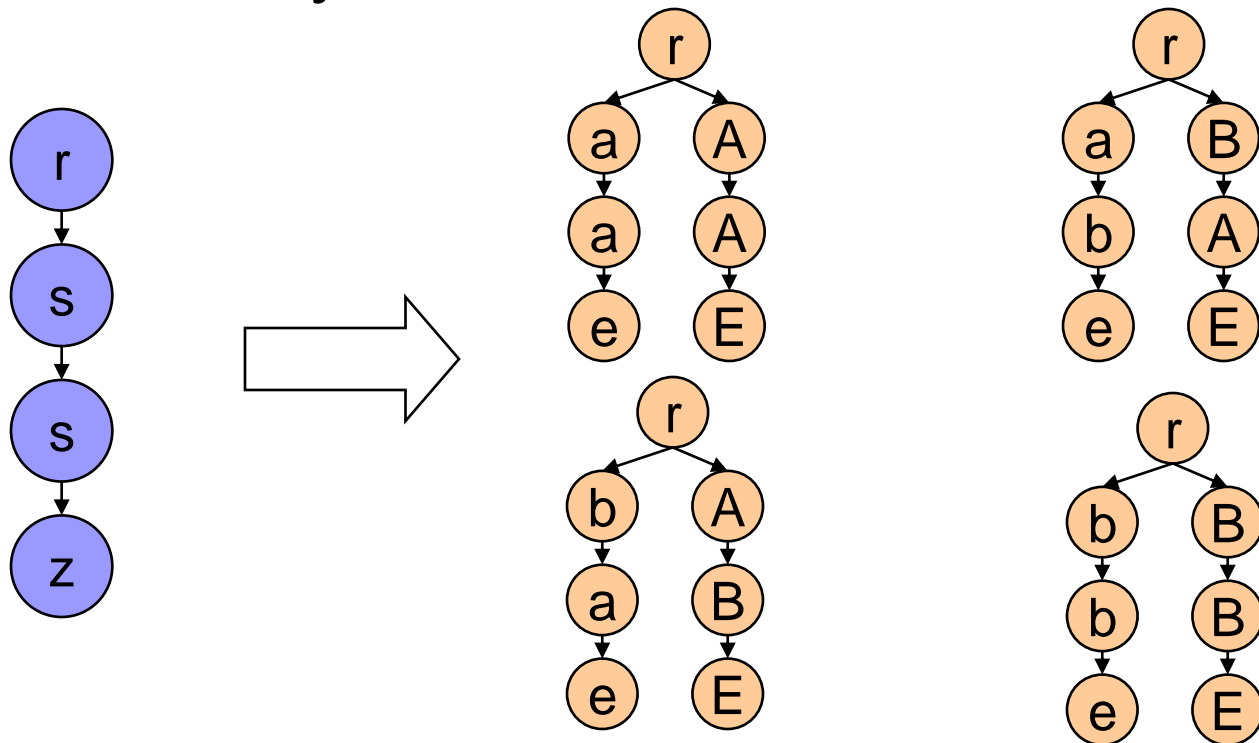
- $MT \not\subseteq 1\text{-MM}$  (Proposition 12)
- Intuition: copying by 'let' variables adds some power

$$\langle q_0, \underline{b}(x_1, x_2) \rangle \rightarrow$$
$$\text{let } z = \langle q, x_1 \rangle(\underline{a}, \underline{a}) \text{ in}$$
$$\langle q, x_2 \rangle( z, z )$$
$$\langle q_0, \underline{b}(x_1, x_2) \rangle \rightarrow$$
$$\langle q, x_2 \rangle( \langle q, x_1 \rangle(\underline{a}, \underline{a}), \langle q, x_1 \rangle(\underline{a}, \underline{a}) )$$

# Expressiveness of 2-MM

- **1-MM  $\not\subseteq$  2-MM (Theorem 13)**

- Witnessed by the ‘twist’ translation in the paper



# Expressiveness of d-MM

- Conjecture

- $d\text{-MM} \subsetneq (d+1)\text{-MM}$  for every  $d \geq 1$

# 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

## ■ $DT ; MM \subseteq MM$

□ 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  $P \times Q$  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').

## ■ $MM ; DtT \subseteq MM$

□ 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  $q \in Q$  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
    - $MM \subseteq 1-MM ; LDtT$
    - $1-MM \subseteq LHOM ; MT$

# Characterization of mr-MTTs (Simulating multiple return values)

## ■ $MM \subseteq 1\text{-}MM$ ; LDtT (Lemma 2)

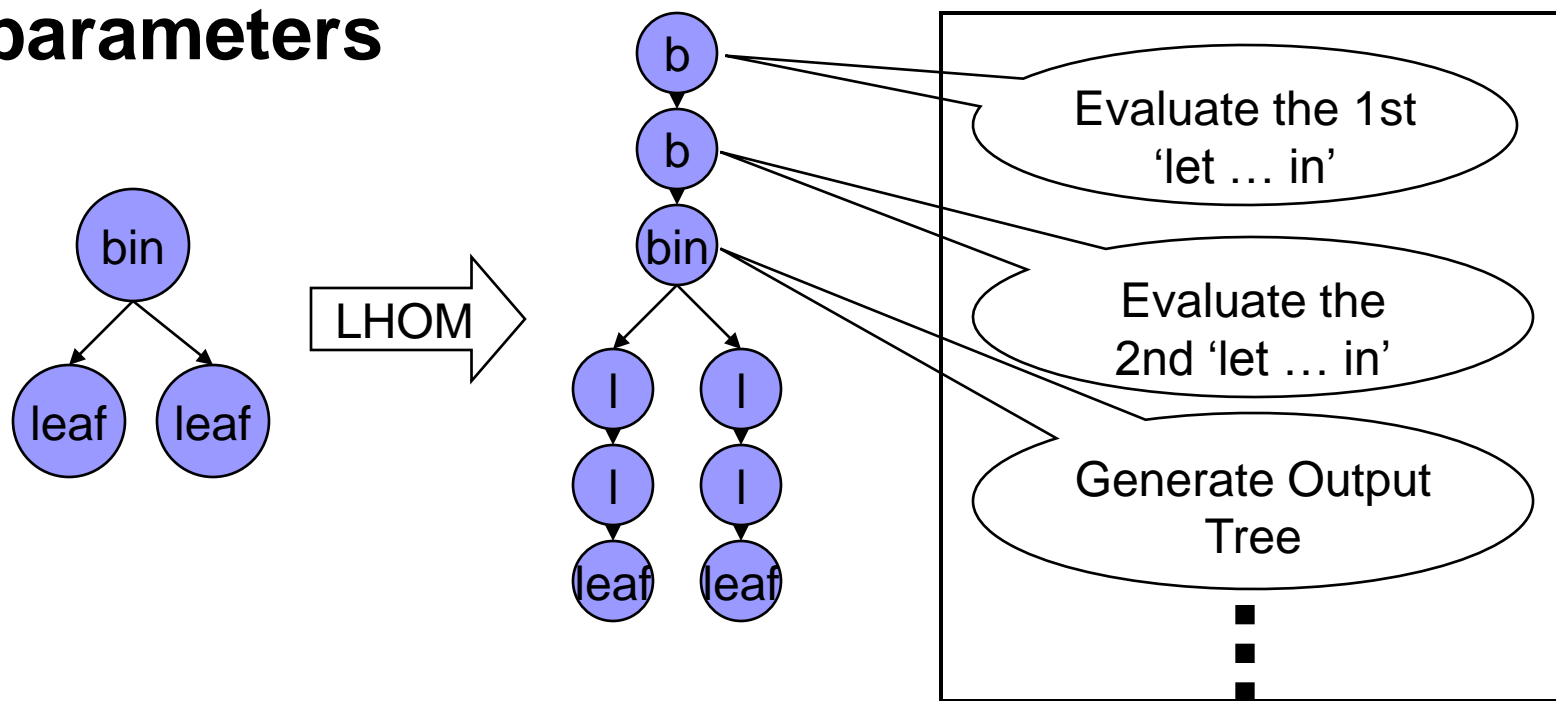
- Intuition: the 1-MM outputs symbolic representations of tupling and projection operations, and the LDtT carries them out

$$\langle q, \underline{b}(x) \rangle \rightarrow$$
$$\text{let } (z_1, z_2) = \langle q, x \rangle \text{ in}$$
$$(\underline{a}(z_1), \underline{b}(z_2))$$
$$\langle q, \underline{b}(x) \rangle \rightarrow$$
$$\text{let } z = \langle q, x \rangle \text{ in}$$
$$\underline{\tau}(\underline{a}(\underline{1}^{\text{st}}(z)), \underline{b}(\underline{2}^{\text{nd}}(z)))$$

# Characterization of mr-MTTs (Simulating 'let'-variable bindings)

## ■ $1\text{-MM} \subseteq \text{LHOM} ; \text{MT}$ (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
  - $MM = LHOM ; MT ; LDtT$