Streaming Tree Transducers

Loris D'Antoni

University of Pennsylvania

Joint work with Rajeev Alur

Outline

1. Deterministic bottom-up MSO equivalent model for ranked tree transformations
2. Deterministic left-to-right MSO equivalent model for tree transformations

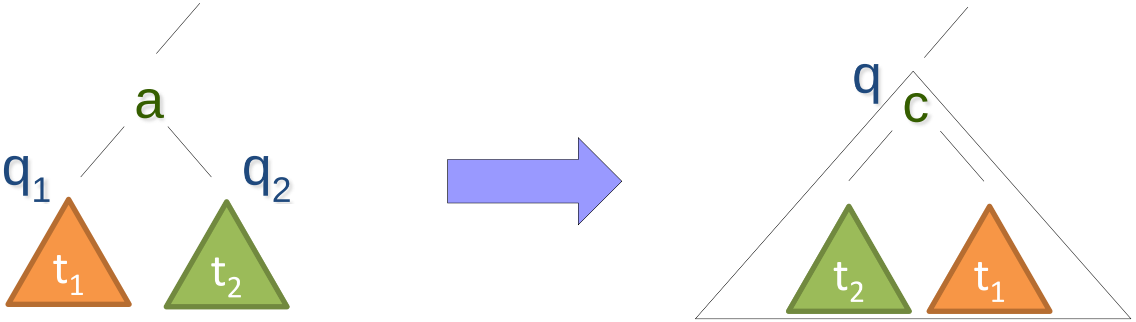
Motivations

- A tree transducer maps a tree over an input alphabet to a tree over an output alphabet
- Desirable properties of a class of transducers C
  - Closure properties:
    - Composition: given T_1, T_2 in C, their composition $T_1 o T_2$ belongs to C (for free if MSO equivalence);
    - Regular look-ahead: ability to ask question about the remaining input, without needing to read it.
  - Fast Execution:
    - single pass over the input tree
    - deterministic
  - Expressiveness: possibly MSO equivalent
  - Fast algorithms: equivalence, type checking…

Example of Transformations

- Insert/delete nodes
- Copy a sub-tree K times
- Swap sub-trees based on some regular pattern
  - Given an address book, where each entry has a tag that denotes whether the entry is "private" or "public", sort the address book based on this tag: all private entries should appear before public entries
- NO actual sorting:
  - we want to be MSO equivalent

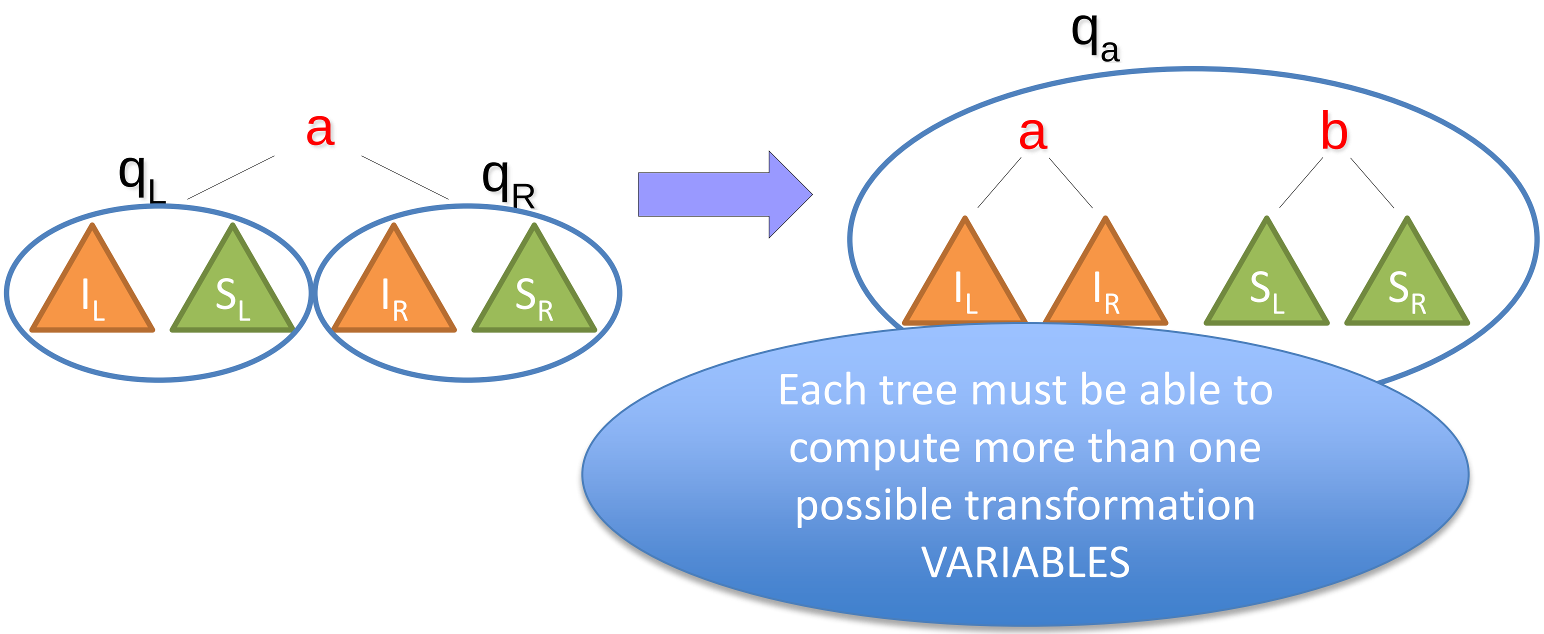
Bottom-up Ranked Tree Transducers



- When processing a tree a(x_1,x_2) the transducer
  - reads the state q_i reached by each child x_i (while going bottom-up)
  - reads the symbol a of the current node
  - Uses the transformations t_1, t_2 computed by the x_1, x_2 to produce a new output
  - Updates the state to q

Multiple Variables Needed

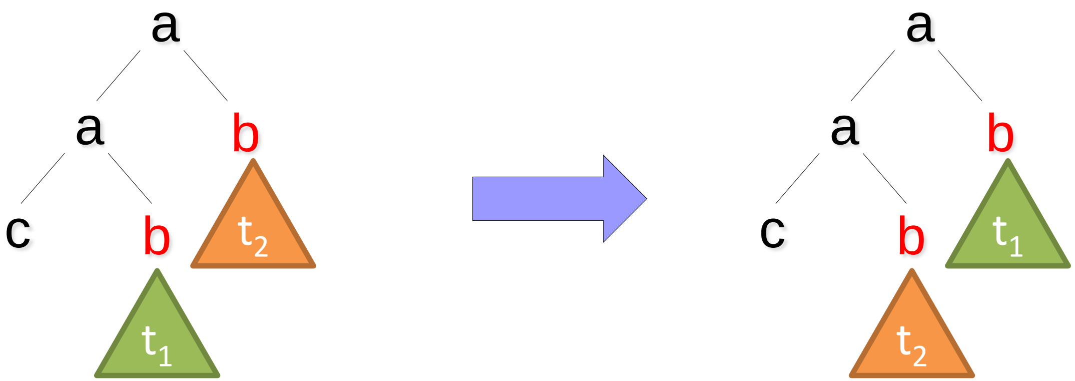
- If the root is labeled with a
  - compute the identity function,
  - otherwise replace each a with b and each b with an a



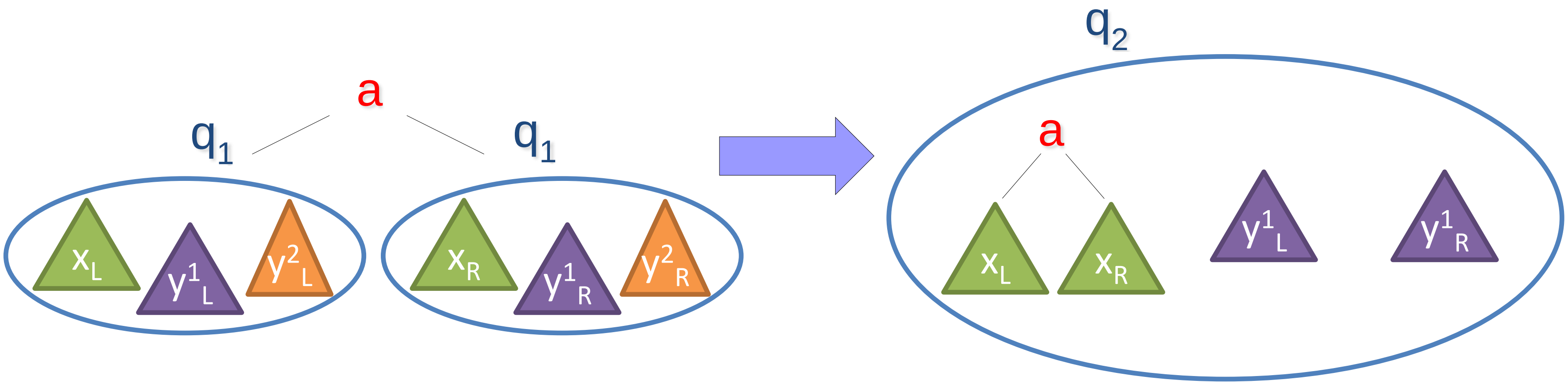
Each tree must be able to compute more than one possible transformation VARIABLES

Holes in Variables Needed 1/3

- Tree Swap: swap the first two sub-trees with root labeled with a b (in-order traversal)

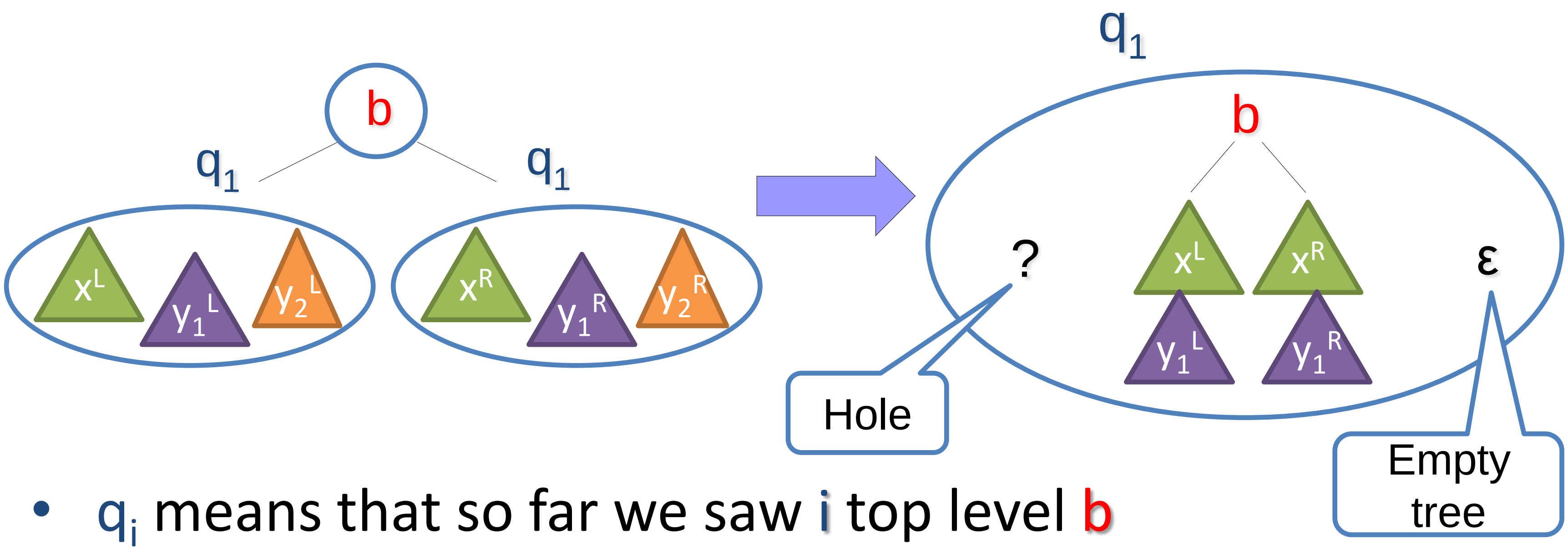


Holes in Variables Needed 2/3



- q_i means that so far we saw i top level b
- y^i contains the i-th b-rooted sub-tree
- x contains the tree processed so far but has i holes in place of the top-level b-rooted sub-trees

Holes in Variables Needed 2/3



Hole

Empty tree

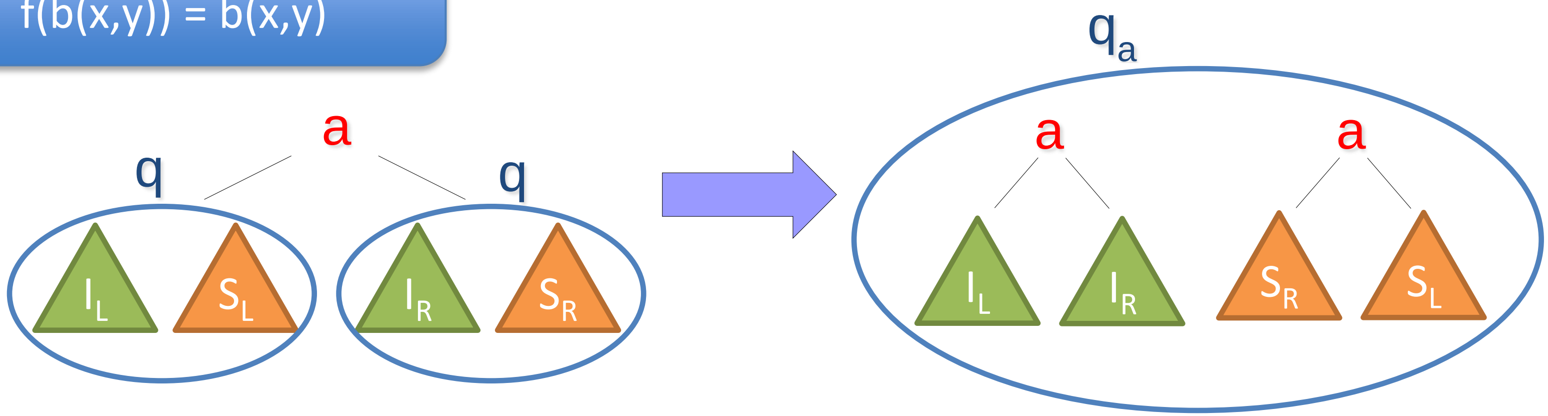
- q_i means that so far we saw i top level b
- y^i contains the i-th b-rooted sub-tree
- x contains the tree processed so far but has i holes in place of the top-level b-rooted sub-trees

Conflict Relation 1/3

- Recursive swap:
  - f(a(x,y)) = a(f(y),f(x))
  - f(b(x,y)) = b(x,y)
- Easy to compute top-down
- Bottom-up it needs two variables

Conflict Relation 2/3

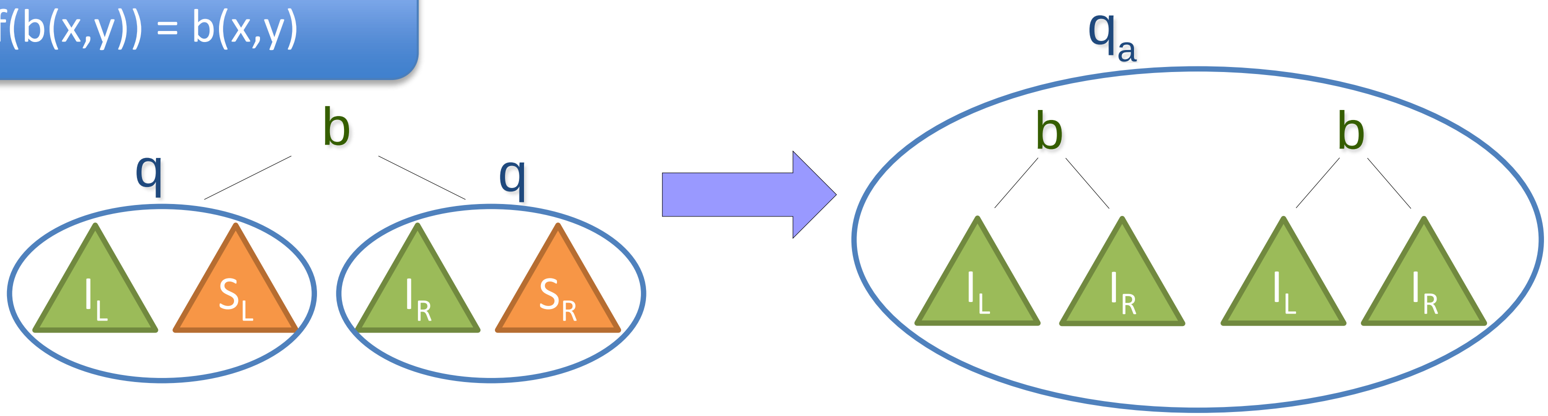
f(a(x,y)) = a(f(y),f(x))
f(b(x,y)) = b(x,y)



- Two variables
  - I computes the identity: case in which we have not hit the last b yet
  - S computes the swap: case in which we have hit the last b

Conflict Relation 3/3

f(a(x,y)) = a(f(y),f(x))
f(b(x,y)) = b(x,y)



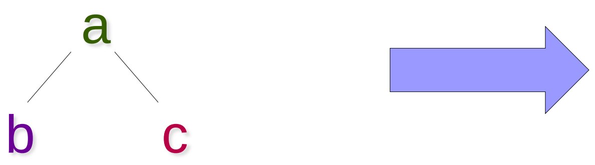
- The variable l is used twice
  - This could cause the output tree to be of exponential size in the size of the input tree (NO MSO)
  - We need the ability of copying but we need to limit it
  - INTUITION: only one of the two trees we are computing will appear in the final output (will explain later)

Streaming Tree Transducers: Design Principles

- Execution: single left-to-right pass in linear time
- Key to expressiveness:
  - multiple variables
  - variables can be stored on stack
  - explicit way of combining sub-trees in the assignments of variables (hole substitution)
- Key to analyzability:
  - single-use restricted updates
  - write-only output
  - Can compute multiple possible partial outputs

Streaming Tree Transducers 1/3

- The input and output trees are represented as nested words



<a <b b> <c c> a>

- Each node is represented by an open tag <a and a close tag a>
- This requires a stack to model the current depth in the input tree (pushdown machine)
- Enables uniform representation of string, ranked trees, unranked trees, and forests

Streaming Tree Transducers 2/3

- STT from Σ to Γ:
  - Q : set of states
  - P : set of stack states
  - X : set of variables
  - ~ : conflict relation over X
  - Variables can contain a hole ?
  - δ : transition function. Updates state when reading input symbol in a given state
  - U : variable update function. Updates variable values when reading an input symbol in a given state.
  - O : output function for combining variables and producing final output

The limited form of coyping

Streaming Tree transducers 3/3

Transition function δ:

- Open Tags:
  - δ(q,<a) → (q',p) (push state p on the stack)
  - x := ?
  - x_p := <b x b> (x stored on the stack as x_p)
- Close tags:
  - δ(q,a>,p) → q’
  - x := <b x x_p b> (x_p popped from the stack)
- Internal:
  - δ(q,a) → q’
  - x := <b x b>

The Conflict Relation

- We want to be able to express the assignment

X:=X
Y:=X

- However x and y must not be combined later
  - we can create an output of size exponential in the input
- SOLUTION: Conflict relation: x ~ y
  - x and y can never appear on the RHS of the same variable assignment
  - Example: z:=a(x,y) is not allowed

STT Properties

- MSO equivalent (closure under composition and regular lookahead)
- Output computed in single left-to-right linear time pass over the input
- Functional equivalence decidable in NExpTime:
  - compute a exponential size PDA over {0,1} that accepts a string with same number of 0s and 1s iff two STTs are not equivalent. Use Parikh Image
- Type checking decidable in ExpTime:
  - given two tree language I and O and an STT S check whether S(I) is included in O

Loris D’Antoni
University of Pennsylvania
lorisdan@seas.upenn.edu

Thank you!
Questions?