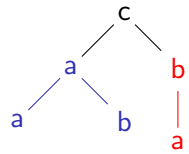
Stackless Processing of Streamed Trees

Corentin Barloy, Filip Murlak, Charles Paperman

Highlights 2021

Processing streamed trees

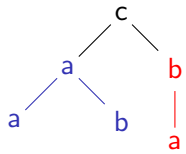
XML encoding of trees:



$\langle c \rangle$
  $\langle a \rangle$
    $\langle a \rangle \langle / a \rangle$
    $\langle b \rangle \langle / b \rangle$
  $\langle / a \rangle$
  $\langle b \rangle$
    $\langle a \rangle \langle / a \rangle$
  $\langle / b \rangle$
$\langle / c \rangle$

Processing streamed trees

XML encoding of trees:

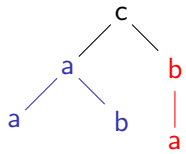


$\langle c \rangle$
  $\langle a \rangle$
    $\langle a \rangle \langle / a \rangle$
    $\langle b \rangle \langle / b \rangle$
  $\langle / a \rangle$
  $\langle b \rangle$
    $\langle a \rangle \langle / a \rangle$
  $\langle / b \rangle$
$\langle / c \rangle$

RPQs: the path from the root belongs to a given regular language.

Processing streamed trees

XML encoding of trees:

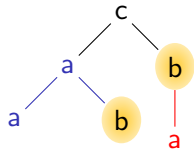


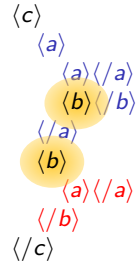
$\langle c \rangle$
  $\langle a \rangle$
    $\langle a \rangle \langle / a \rangle$
    $\langle b \rangle \langle / b \rangle$
  $\langle / a \rangle$
  $\langle b \rangle$
    $\langle a \rangle \langle / a \rangle$
  $\langle / b \rangle$
$\langle / c \rangle$

RPQs: the path from the root belongs to a given regular language.
For instance, the RPQ associated to ca^*b.

Processing streamed trees

XML encoding of trees:





RPQs: the path from the root belongs to a given regular language.
For instance, the RPQ associated to ca^*b.

Evaluation in constant memory VS linear memory

- We have an effective characterisation of the RPQs that can be evaluated in constant memory.

Evaluation in constant memory VS linear memory

- We have an effective characterisation of the RPQs that can be evaluated in constant memory.
- It is very limited: $//a/b$ and $//a//b$ are not doable.

Evaluation in constant memory VS linear memory

- We have an effective characterisation of the RPQs that can be evaluated in constant memory.
- It is very limited: $//a/b$ and $//a//b$ are not doable.
- All RPQs can be evaluated with a stack, but this is costly.

Evaluation in logarithmic memory (Stackless automata)

Main ingredients:
- a finite state machine,
- a counter that stores the current depth in the tree,
- a finite number of registers where the counter values can be stored,
- can compare register values with the current depth.

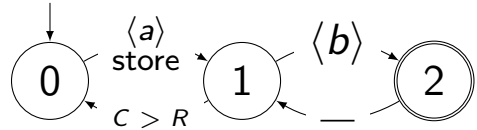
Evaluation in logarithmic memory (Stackless automata)

Main ingredients:
- a finite state machine,
- a counter that stores the current depth in the tree,
- a finite number of registers where the counter values can be stored,
- can compare register values with the current depth.

Evaluating:

$(a+b+c)^*a(a+b+c)^*b$

//a//b



Main result

Theorem

We can decide whether a given RPQ can be evaluated with a stackless automaton

Main result

Theorem

We can decide whether a given RPQ can be evaluated with a stackless automaton

(`//a//b` is doable but still not `//a/b`)

Conclusion

- Similar characterisations for the validation problem.

Conclusion

- Similar characterisations for the validation problem.
- Ongoing work on leveraging schemas for querying streamed trees.

Conclusion

- Similar characterisations for the validation problem.
- Ongoing work on leveraging schemas for querying streamed trees.
- Ongoing work on vectorization.