Visibly Pushdown Transducers with Look-Ahead

Emmanuel Filiot and Frédéric Servais

Université Libre de Bruxelles
University of Hasselt

SOFSEM 2012

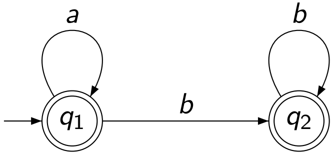
Plan

- Transducers: Finite State and Pushdown
- Visibly Pushdown Transducers
- VPT with Look-ahead
- Open Problems and Future Work

Transducers

Automata with output

Finite State Automaton



$L(A)$: $a^m b^n$ for all $m, n \geq 0$

Define regular languages which are closed under (nearly) everything and all decision problems are decidable.

Finite State Transducers

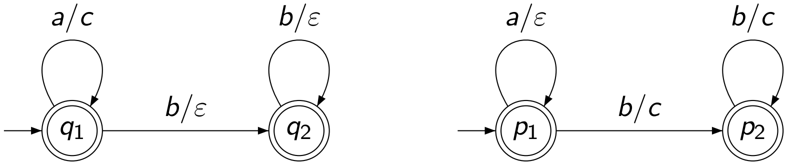


Figure: Two finite state transducers: T_1 (left) and T_2 (right).

$$
\begin{array}{llll}
R(T_1): & a^m b^n & \rightarrow c^m & \text{for all } m, n \geq 0 \\
R(T_2): & a^m b^n & \rightarrow c^n & \text{for all } m, n \geq 0
\end{array}
$$

Finite State Transducers

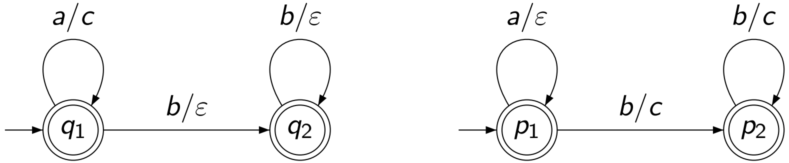


Figure: Two finite state transducers: T_1 (left) and T_2 (right).

$$\begin{array}{llll} R(T_1): & a^m b^n & \rightarrow c^m & \text{for all } m, n \geq 0 \\ R(T_2): & a^m b^n & \rightarrow c^n & \text{for all } m, n \geq 0 \end{array}$$

What about closure and decision problems ? Not as good as for regular languages.

Finite State Transducers

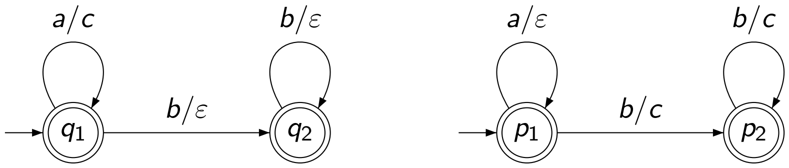


Figure: Two finite state transducers: T_1 (left) and T_2 (right).

$$R(T_1): \quad a^m b^n \quad \to c^m \quad \text{for all } m, n \geq 0$$

$$R(T_2): \quad a^m b^n \quad \to c^n \quad \text{for all } m, n \geq 0$$

Not closed under intersection:

$$R(T_1) \cap R(T_2) = \{(a^n b^n, c^n) \mid n \geq 0\}$$

Finite State Transducers

- Closed under union, composition, lookahead.
- Not closed under intersection, complement.
- Decidable emptiness, functionality, determinizability, type checking.
- Undecidable inclusion, equivalence.

Finite State Transducers

- Closed under union, composition, lookahead.
- Not closed under intersection, complement.
- Decidable emptiness, functionality, determinizability, type checking.
- Undecidable inclusion, equivalence.

Functional (and finite-valued) finite state transducers:

- Closed under (union), composition, lookahead.
- Not closed under intersection, complement.
- Decidable emptiness, (functionality), determinizability, type checking.
- $\boxed{\text{Decidable inclusion, equivalence.}}$

Pushdown Transducers

Adding a stack

Pushdown Transducers

- Closed under union.
- Not closed under intersection, complement, composition, (lookahead).
- Decidable emptiness.
- Undecidable inclusion, equivalence, functionality, determinizability, type checking.

Pushdown Transducers

- Closed under union.
- Not closed under intersection, complement, composition, (lookahead).
- Decidable emptiness.
- Undecidable inclusion, equivalence, functionality, determinizability, type checking.

Functional (and finite-valued) pushdown transducers:

- Closed under (union).
- Not closed under intersection, complement, composition, (lookahead).
- Decidable emptiness.
- Undecidable inclusion, equivalence, functionality, determinizability, type checking.

Visibly Pushdown Automata

R. Alur and P. Madhusudan 2004

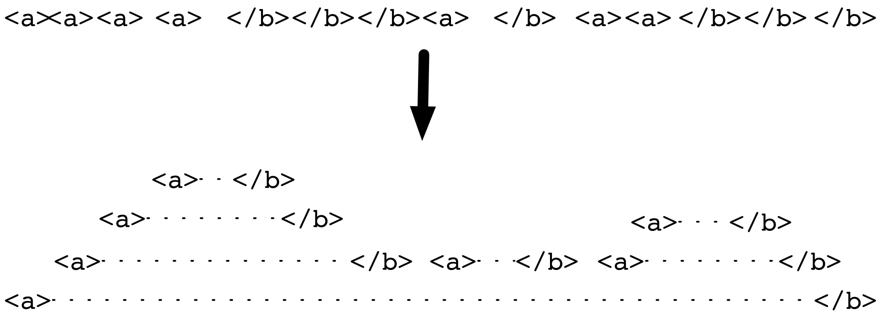
The alphabet drives the stack

Looking for a good subclass of pushdown automata

Partition the input alphabet into two types of symbols:

- *Call* symbols (opening): `<a>`, `<b>`, `<c>`...
- *Return* symbols (closing): `</a>`, `</b>`, `</c>`...

This partition induces a *nesting structure* on the words.



Visibly Pushdown Automata (VPA)

A Visibly Pushdown Automaton is a Pushdown Automaton such that:

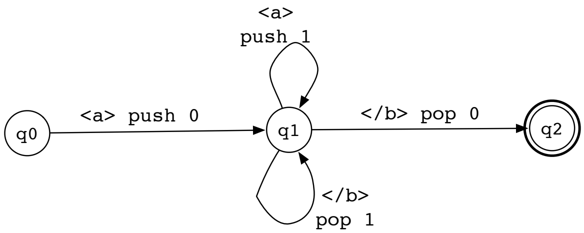
- When it reads a call it must push one symbol on the stack.
- When it reads a return it must pop the top of the stack.

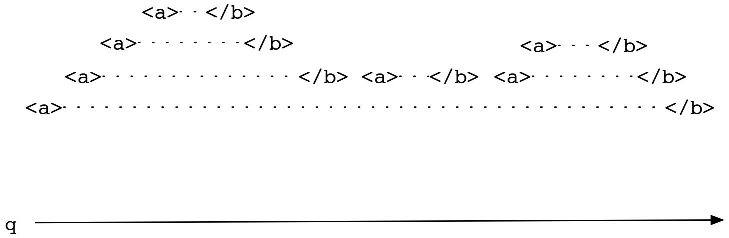
Product construction is possible because the stack operations are synchronized on the input word (stack can be simulated).

Proposition - Visibly Pushdown Languages [Alur and Madhu., 2004]

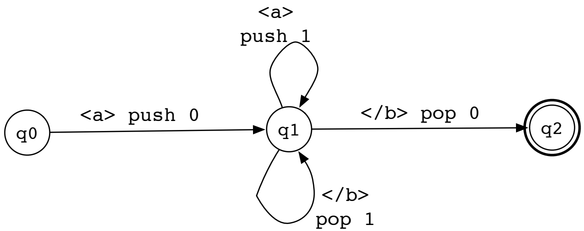
The class of VPL is closed under all Boolean operations.
Equivalence, inclusion, emptiness, universality are all decidable.

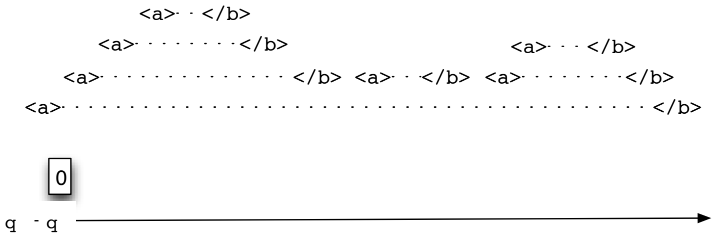
Visibly Pushdown Automata (VPA)



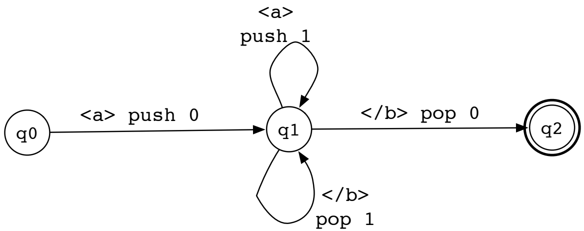


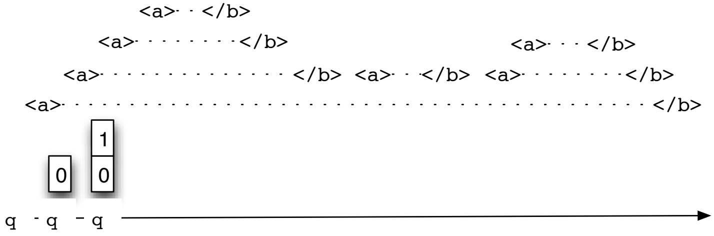
Visibly Pushdown Automata (VPA)



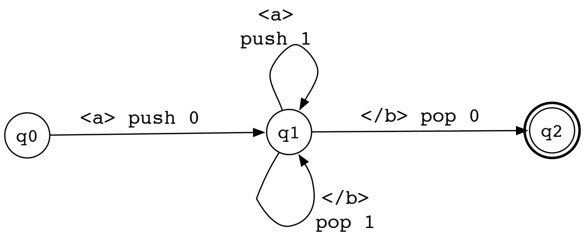


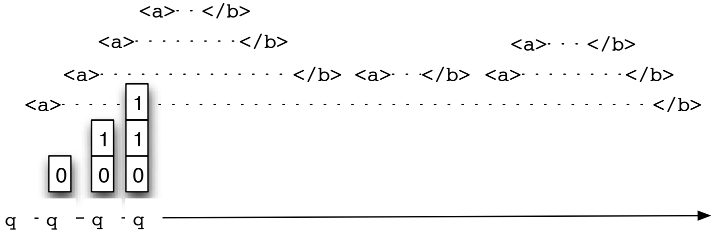
Visibly Pushdown Automata (VPA)



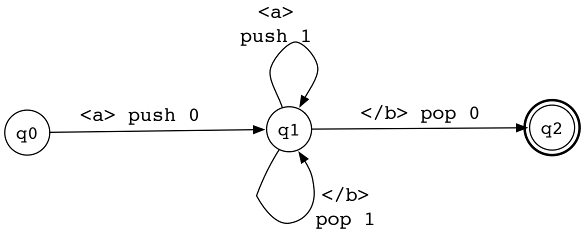


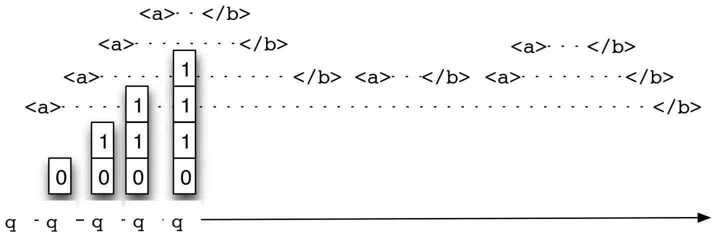
Visibly Pushdown Automata (VPA)



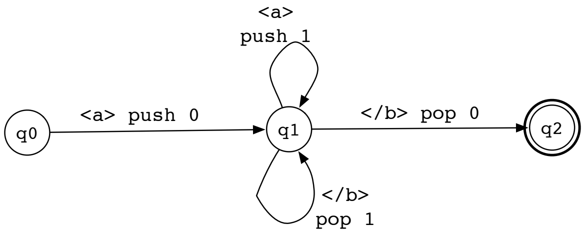


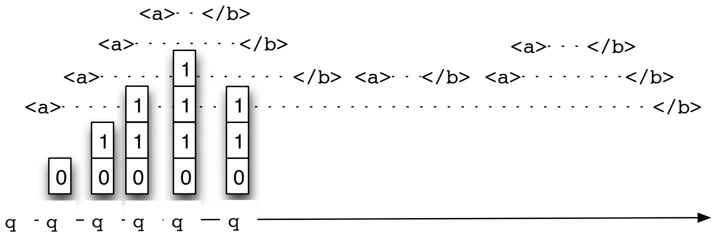
Visibly Pushdown Automata (VPA)



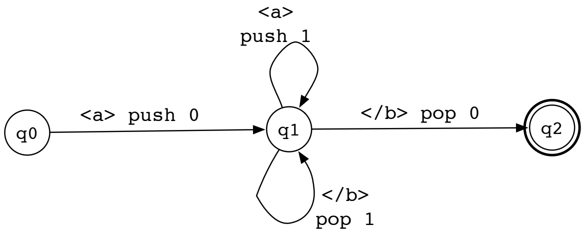


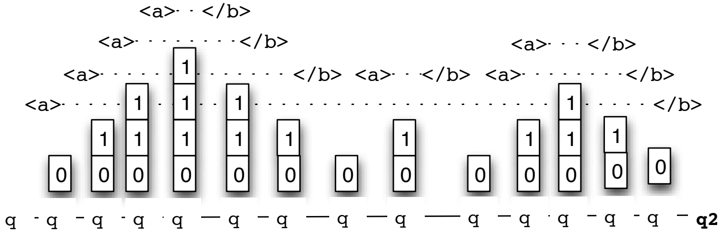
Visibly Pushdown Automata (VPA)





Visibly Pushdown Automata (VPA)





VPA properties

Closure	$\overline{L}$	$L_1 \cup L_2$	$L_1 \cap L_2$	$L_1 L_2$	$L \cap \text{REG}$
Finite state	yes	yes	yes	yes	yes
Visibly pushdown	yes	yes	yes	yes	yes
Pushdown	no	yes	no	yes	yes

Decision problems	emptiness membership	equivalence inclusion	universality
Finite state	PTIME	PSPACE-C	PSPACE-C
Visibly pushdown	PTIME	EXPTIME-C	EXPTIME-C
Pushdown	PTIME	undec	undec

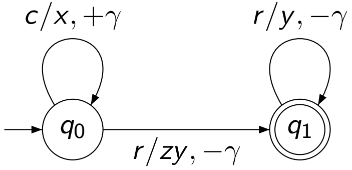
Visibly Pushdown Transducers

Visibly Pushdown Transducers

Input alphabet:

- Call symbols: $\{c\}$
- Return symbols: $\{r\}$

Output alphabet: $\{x, y, z\}$.



$$c^n r^n \to x^n z y^n \quad \text{for all} \quad 1 \leq n$$

Visibly Pushdown Transducers - Properties

- Closed under union, (composition), lookahead (this paper).
- Not closed under intersection, complement.
- Decidable emptiness, functionality, (determinizability ?), type checking.
- Undecidable inclusion, equivalence.

Visibly Pushdown Transducers - Properties

- Closed under union, (composition), lookahead (this paper).
- Not closed under intersection, complement.
- Decidable emptiness, functionality, (determinizability ?), type checking.
- Undecidable inclusion, equivalence.

Functional (and finite-valued) finite state transducers:

Closed under (union), composition, lookahead (this paper).

- Not closed under intersection, complement.
- Decidable emptiness, (functionality), (determinizability ?), type checking.
- $\boxed{\text{Decidable inclusion, equivalence.}}$

Functional (and *k*-valued) Transducers

Finite State Transducers:

- Closed under (union), composition, lookahead (this paper).
- Not closed under intersection, complement.
- Decidable emptiness, functionality, determinizability, type checking, inclusion, equivalence.

Pushdown Transducers:

- Closed under (union).
- Not closed under intersection, complement, composition, (lookahead).
- Decidable emptiness.
- Undecidable inclusion, equivalence, functionality, determinizability, type checking.

Visibly Pushdown Transducers:

- Closed under (union), composition, lookahead (this paper).
- Not closed under intersection, complement.
- Decidable emptiness, (functionality), (determinizability ?), type checking, inclusion, equivalence.

Look-Ahead

Useful syntactic sugar

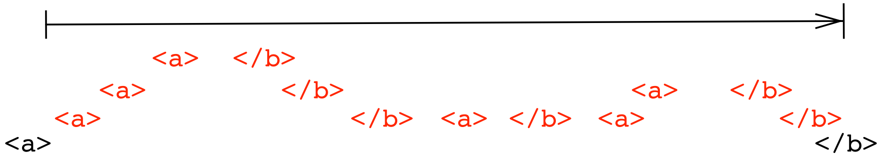
Determinism vs Functional

- Determinism is too restrictive to define all functional transduction.
- To stay determinist but express all functional transductions we need some look-ahead.
- What look-aheads are necessary to capture all functional VPT?
- Are VPT with look-ahead more expressive than VPT?

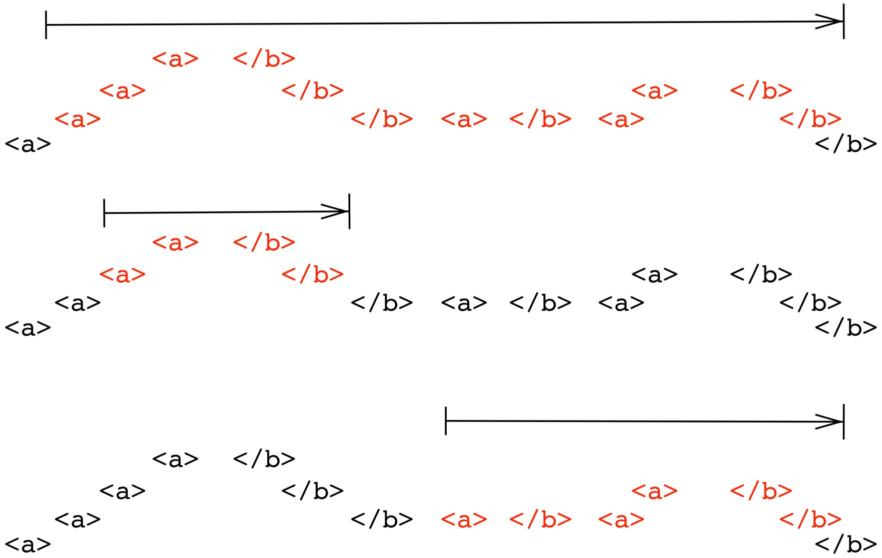
VPT with Look-Ahead

Definition

A VPT with look-ahead is a VPT s.t. call transitions are guarded by VPL. A call transition guarded by L can be fired if the longest well-nested sub-word starting at the call is in L.



VPT with Look-Ahead



VPT with or without Look-Ahead

Theorem

VPT *and* VPT *with look-ahead are equally expressive*
(but exponentially more succinct).

- Challenge: Unbounded number of running look-aheads.
- Unbounded 1: `<c`$_1$`></r`$_1$`> <c`$_2$`></r`$_2$`> ... <c`$_n$`> </r`$_n$`>`
  - when reading `<c`$_1$`>` a new look-ahead is triggered, this look-ahead will run until `<r`$_n$`>`.
  - $\rightarrow$ after reading k successive `<c> </r>` there are (at least) k simultaneous running look-aheads.
- Unbounded 2: `<c> <c> <c>...<r> <r>...<r>`.
- Idea 1: Simulate all look-aheads with a subset construction.
- Idea 2: Deal with the stack using summaries (Alur 2004).
- Cost: Exponential blow-up.

Deterministic VPT with Look-Ahead

Theorem

Functional VPT *and deterministic* VPT *with look-ahead are equally expressive.*

- Challenge: Unbounded number of runs.
- Idea: All accepting runs have the same output (functional).
  Order the runs of the VPT (lexicographic ordering).
  Choose the smallest accepting one using look-ahead.
- Careful: when entering a new nesting level, thanks to look-ahead choose the smallest run that is *compatible* with the chosen *global* run!
- Cost: Exponential blow-up.

VPT with Look-Ahead

Corollary

Functional VPT *and unambiguous* VPT *are equally expressive.*

- Idea: apply successively the two previous theorems.

  $$\text{functional VPT} \quad \rightarrow \quad \text{deterministic VPT}_{\text{la}} \quad \rightarrow \quad \text{VPT}$$

  The resulting VPT is unambiguous.
- Cost: Doubly exponential blow-up.

Theorem

Equivalence and inclusion of VPT_{la} *is* EXPTIME-C.
(Same as for VPT *despite being exponentially more succinct).*
Functionality, emptiness are EXPTIME-C.

Conclusion

Finally.

Conclusion

We showed:

- Closure under look-ahead.
- Characterization of functional VPT by deterministic VPT_{la}.
- Characterization of functional VPT by unambiguous VPT.
- Complexity of decision problems for VPT_{la}.
- Discussion on variants of look-ahead (shorter, longer...).

$\rightarrow$ VPT form a robust class of transducers.

Open Problems and Future Work

Open problems

- Deciding determinizability, determinization procedure (*coming soon*).
- Deciding equivalence of k-valued transducers.
- Deciding finite-valuedness.
- k-valued VPT $=$ k-ambiguous VPT?