Visibly Pushdown Languages, by Rajeev Alur and P. Madhusudan

William Blum

January 30, 2005

Visibly pushdown languages:

- rich enough for modelling programs
- nice properties

A visibly pushdown automaton is a pushdown automaton where the input symbol determines whether the automaton push or pop from the stack.

1 Definition

The alphabet Σ is partitioned into three different alphabets: one for call symbols, one for return symbols, and one for internal actions.

$$\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_{int}$$

If a call symbol is read, the automaton has to push an element on the stack. If it is a return symbol, it pops from the stack. Otherwise, for internal actions, it does not push or pop.

A visibly pushdown automaton on finite words over the alphabet $\hat{\Sigma} = \langle \Sigma_c, \Sigma_r, \Sigma_{int} \rangle$ is a tuple $M = (Q, Q_{in}, \Sigma, \delta, Q_F)$. Q is the set of states, Q_{in} the set of initial states. Γ is the alphabet use for the stack (finite) containing the emtpy stack symbol $\perp$. $\delta \subset (Q \times \Sigma_C \times Q \times (\Gamma \setminus \{\perp\})) \cup (Q \times \Sigma_r \times \Gamma \times Q) \cup (Q \times \Sigma_{int} \times Q)$, Q_F is the set of final states.

There are three types of transition depending on the kind of symbol read.

The stack is nonempty finite sequence of element of Γ.

The languages recognized by VPA are the *visibly pushdown languages*: VPL.

VPA languages are a strict subclass of context-free languages (take $L = \{a^n b a^n \text{ for } n \in \mathbb{N}\}$).

However for any context-free language, it is possible to associate a VPA over a different alphabet: every symbol of the initial alphabet is augmented with some information about the stack state. There is an isomorphism between the set of CFL and the set of CFL transformed into VPL.

1.1 Theorem 1: closure

VPL are closed under:

- union
- intersection
- complementation
- renaming
- concatenation
- Kleene-*

1.2 Theorem 2

Non deterministic VPA can be determinized.

1.3 Theorem 3

Universality and inclusion problem undecidable for Context-free languages (CFL) becomes EXPTIME-complete for VPL languages.

2 Logical characterization

Let MSO_μ be the MSO theory (Monadic Second Order logic) augmented with a binary matching predicate $\mu(x, y)$. Then MSO_μ is equivalent to VPL.

2.1 Theorem 4

L is VPL iff there is an MSO_μ sentence ψ (over the same alphabet as L) that defines L.

3 Regular Tree Languages

We fixed an alphabet $\hat{\Sigma}$, then we can associate a tree to any word of Σ^+. These tree are called stack trees: $STree$.

Regular tree languages are the languages recognized by a special kind of automata over trees.

3.1 Theorem 5

The class VPL coincides with regular tree languages

4 Visibly pushdown ω-languages : ω-VPL

ω-VPL: VPA on ω-words using Bchi or Muller acceptance condition.

Decision problems: same complexity as for VPL.

4.1 Theorem 6 and 8 - Closure

The resulting class is closed under union, intersection, renaming, complementation.

4.2 Theorem 7

Nondeterministic Bchi VPA are stricly more expressive than deterministic Bchi or Muller VPA.

This is the main difference between VPL and ω-VPL.

4.3 Theorem 9 : $\mathbf{MSO}_\mu$ characterization

ω-VPL are characterized by MSO_μ formulas.

4.4 Theorem 10

Same as for VPL where stack-trees are defined using Bchi automaton on trees.