deterministic pushdown automaton*

CWoo†

2013-03-22 2:52:02

A pushdown automaton $M = (Q, \Sigma, \Gamma, T, q_0, \perp, F)$ is usually called "non-deterministic" because the image of the transition function T is a subset of $Q \times \Gamma^*$, which may possibly contain more than one element. In other words, the transition from one configuration to the next is not uniquely determined. When there is uniqueness, M is called "deterministic".

Formally, a *deterministic pushdown automaton*, or *DPDA* for short, is a non-deterministic pushdown automaton $M = (Q, \Sigma, \Gamma, T, q_0, \perp, F)$ where the transition function T has the following properties: for any $p \in Q$, $a \in \Sigma$, and $A \in \Gamma$,

1. $T(p, a, A) \cup T(p, \lambda, A)$ is at most a singleton,
2. $T(p, a, A) \cap T(p, \lambda, A) = \varnothing$.

The properties can be interpreted as follows: given any configuration of M, if there is a transition to the next configuration, the transition must be unique. The second property just insures that $T(p, a, A) \neq T(p, \lambda, A)$, so that when a λ-transition is possible for a given (p, A), no other transitions are possible for the same (p, A).

The way a DPDA works is exactly the same as an NPDA, with several modes of acceptance: acceptance on final state, acceptance on empty stack, and acceptance on final state and empty stack. However, unlike a NPDA, these acceptance methods are not equivalent. It can be shown that the set $\mathscr{E}$ of languages accepted on empty stack is a proper subset of the set $\mathscr{F}$ of languages determined on final state. In fact, every language in $\mathscr{E}$ is prefix-free, while some languages in $\mathscr{F}$ are not.

Nevertheless, any regular language can be accepted by a DPDA on empty stack, and any language accepted by a DPDA on final state is unambiguous, and, as a result, $\mathscr{F}$ is a proper subset of the family of all context-free languages. This is quite unlike the case for finite automata: every non-deterministic finite automaton is equivalent to a deterministic finite automaton. A language in $\mathscr{F}$ called a *deterministic language*.

---

*⟨*DeterministicPushdownAutomaton*⟩ created: ⟨*2013-03-2*⟩ by: ⟨*CWoo*⟩ version: ⟨*41787*⟩ Privacy setting: ⟨*1*⟩ ⟨*Definition*⟩ ⟨*03D10*⟩ ⟨*68Q42*⟩ ⟨*68Q05*⟩

Some examples: the set of palindromes $\{u \in \Sigma^* \mid u = \mathrm{rev}(u)\}$ is unambiguous, but not deterministic. The language $\{a^m b^n \mid m \geq n \geq 0\}$ is deterministic, but not prefix-free, and hence can not be accepted by any DPDA on empty stack. The language $\{a^n b^n \mid n \geq 0\}$ can be accepted by a DPDA on empty stack, but is not regular.

Any formal grammar that generates a deterministic language is said to be *deterministic context-free*. A deterministic context-free grammar can be described by what is known as the $LR(k)$ grammars.

The family of deterministic languages is closed under complementation, intersection with a regular language, but not arbitrary (finite) intersection, and hence not union.

Remark. The reason why $\mathscr{E} \neq \mathscr{F}$ can be traced back to the definition of a DPDA: it allows for the following possibilities for a DPDA M:

- M completely stops reading an input word because either there are no available transitions from one configuration to the next:

  $$T(p, a, A) \cup T(p, \lambda, A) = \varnothing,$$

  or the stack is emptied before the last input symbol is read: a configuration (p, u, λ) is reached and u is not empty.

- M consumes the last input symbol, and continues processing because of λ-transitions.

Some authors consider these imperfections of M as being "non-deterministic", and put additional constraints on M, such as making sure T is a total function, the stack is never empty, and delimiting input strings.

References

[1] A. Salomaa *Computation and Automata, Encyclopedia of Mathematics and Its Applications, Vol. 25*. Cambridge (1985).

[2] S. Ginsburg, *The Mathematical Theory of Context-Free Languages*, McGraw-Hill, New York (1966).

[3] D. C. Kozen, *Automata and Computability*, Springer, New York (1997).

[4] J.E. Hopcroft, J.D. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley, (1969).