About Pushdown Automata

The difference between DPDA and NPDA

Ruirui Zhang
201800301072
May 18, 2020

Abstract

Computational theory is a branch of theoretical computer science in which a pushdown automaton (PDA) is a type of automaton that USES a stack.

Push-down automata are used in theories about what machines can calculate. They are more capable than finite state machines, but not as capable as Turing machines. Deterministic push-down automata can recognize all deterministic context-free languages, while non-deterministic push-down automata can recognize all context-free languages. The former is usually used in parser design.

This paper first explains the definition of DPDA and NPDA and the related problems, and then briefly discusses the connection and difference between DPDA and NPDA.[1]

Keywords: PDA, NPDA, DPDA

1 Definition of DPDA and NPDA

1.1 Definition of DCFL

Before understanding DPDA and NPDA, we need to understand context-free languages and deterministic context-free languages, abbreviated CFL and DCFL.
In formal language theory, deterministic context-free languages (DCFL) are a proper subset of context-free languages. They are the context-free languages that can be accepted by a deterministic pushdown automaton. DCFLs are always unambiguous, meaning that they admit an unambiguous grammar. There are non-deterministic unambiguous CFLs, so DCFLs form a proper subset of unambiguous CFLs.

DCFLs are of great practical interest, as they can be parsed in linear time, and various restricted forms of DCFGs admit simple practical parsers. They are thus widely used throughout computer science.

1.2 Definition of PDA

The PDA is an automaton equivalent to the CFG in language-defining power. A PDA is described normally as a 7-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ where:

- Q is a finite set of states
- Σ is a finite set of input symbols
- Γ is a finite set of stack symbols
- q_0 is the start state
- Z_0 is the starting stack symbol
- A is the set of accepting states
- δ is a transition function, where δ: $(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \longrightarrow \mathcal{P}(Q \times \Gamma^*)$ where $*$ is the Kleene star, meaning that Γ^* is the set of all finite strings (including the empty string) of elements of Γ ,and $\mathcal{P}(X)$ is the power set of a set X.

M is deterministic if it satisfies both the following conditions:

1. For any $q \in Q, a \in \Sigma \cup \{\varepsilon\}, x \in \Gamma$, the set $\delta(q, a, x)$ has at most one element.
2. For any $q \in Q, x \in \Gamma$, if $\delta(q, \varepsilon, x) \neq \emptyset$, then $\delta(q, a, x) = \emptyset$ for every $a \in \Sigma$.

If, in every situation, at most one such transition action is possible, then the automaton is called a **deterministic pushdown automaton** (**DPDA**). In general, if several actions are possible, then the automaton is called a **general**, or **nondeterministic**, **PDA**. [2]
There are two possible acceptance criteria: acceptance by empty stack and acceptance by final state. The two are not equivalent for the DPDA (although they are for the NPDA). The languages accepted by empty stack are those languages that are accepted by final state and are prefix-free: no word in the language is the prefix of another word in the language.

2 Diffence between DPDA and NPDA

2.1 Languages recognized

The language identified by DPDA is a deterministic context-free language, while the language identified by NPDA is a context-free language. From the perspective of recognition language, DPDA recognition language is a subset of NPDA recognition language, while DFA and NFA recognition language are the same - both are regular languages, so we can think that DPDA and NPDA are not equivalent, that is, NPDA is not weaker than DPDA.

If L(A) is a language accepted by a NPDA A, it can also be accepted by a DPDA if and only if there is a single computation from the initial configuration until an accepting one for all strings belonging to L(A). If L(A) can be accepted by a NPDA it is a context free language and if it can be accepted by a DPDA it is a deterministic context-free language (DCFL).
But in fact, not all context-free languages are deterministic. This makes the DPDA a strictly weaker device than the NPDA.
For example, the language L_p of even-length palindromes on the alphabet of 0 and 1 has the context-free grammar $S \longrightarrow 0S0|1S1|$.
If a DPDA for this language exists, and it sees a string 0^n, it must use its stack to memoize the length n, in order to be able to distinguish its possible continuations $0^n110^nL_p$ and $0^n110^{n+2}L_p$. Hence, after reading 0^n110^n, comparing the post-"11" length to the pre-"11" length will make the stack empty again. For this reason, the strings $0^n110^n0^n110^nL_p$and $0^n110^n0^{n+2}110^{n+2}L_p$ cannot be distinguished.[3]

2.2 Properties

2.3 Closure

Closure properties of deterministic context-free languages (accepted by DPDA by final state) are drastically different from the context-free languages.
For example, the languages which DPDAs recognize are closed under complementation, but not closed under union.
As a consequence of the complementation it is decidable whether a DPDA accepts all words over its input alphabet, by testing its complement for emptiness. This is not possible for context-free grammars, hence not for NPDA.[3]

2.4 Equivalence problem

For nondeterministic PDA, equivalence is undecidable. But it is decidable for DPDA.
Géraud Sénizergues (1997) proved that the equivalence problem for deterministic PDA (i.e. given two deterministic PDA A and B, is L(A)=L(B)?) is decidable[4], a proof that earned him the 2002 Gödel Prize. [5]

2.5 Simulation

The main advantage of DPDAs is that we can simulate them much more easily with our deterministic computers (real hardware is always deterministic).
In fact, simulating general DPDAs is not fast enough for most purposes, and so when parsing code we usually use LALR grammars which are weaker than DPDAs.
On the contrary, simulating NPDAs is harder than simulating DPDAs on actual computers.

Conclusion

There are many differences and connections between DPDA and NPDA. There is no doubt that NPDA is better than DPDA, but in terms of specific hardware implementation, DPDA has a great advantage. In the process of learning and research, we need to choose DPDA or NPDA according to the specific situation.

References

1. Sipsermichael. Introduction to the theory of computation. *Sigact News*, 1996.

2. Wikipedia. Pushdown automaton. `https://en.wikipedia.org/wiki/Pushdown_automaton`.

3. Wikipedia. Deterministic pushdown automaton. `https://en.wikipedia.org/wiki/Deterministic_pushdown_automaton`.

4. Geraud Senizergues. L(a)=l(b)? a simplified decidability proof. *Theoretical Computer Science*, 281(1):555–608, 2002.

5. Geraud Senizergues. The equivalence problem for deterministic pushdown automata is decidable. pages 671–681, 1997.