Grace Rademacher

CSE 30151

Extra Credit Assignment

12/16/2024

Nondeterminism

Last exam, question 4 stumped me. It asked us to develop a PDA to accept language L = $\{a^k r \mid r \in \{a,b\}^*, |r| \leq k\}$. We didn't need to create a CFG first. I quickly became confused as to how to show when a^k ended and r started. I spent a lot of time on that question until I quickly jotted down some messy PDA that attempted to show exactly how to deterministically tell when the strings switched. I had failed to realize that this question was quite simple–just craft a nondeterministic PDA. I clearly did not fully understand how to use the "crystal ball" and nondeterminism. Instead of the horribly drawn deterministic PDA I had attempted, a nondeterministic PDA could just transition to the next state when the r string starts. I never realized it could be that simple!

Nondeterministic PDAs are much more powerful than deterministic ones. In converting from a CFG to a PDA, there is much ambiguity: the same string can have multiple valid parse trees. If we used a DPDA, it would have to decide at each step which action to take based on the current input and stack. If there is no clear path to take, the DPDA will fail. However, a NPDA could explore all the different parse trees as NPDAs have the ability to "guess" the right transitions. The ability to guess is the NPDA's ability to explore possible computation paths simultaneously. NPDAs can actually branch into multiple paths, effectively "guessing" the correct path to take towards acceptance. If a NPDA finds any path that leads to an accepting

state, the NPDA accepts the input string. Thus, NPDAs are a very powerful tool to use for creating PDAs for CFGs.

Now that I have a better understanding of what exactly nondeterministic PDAs are and their ability to "guess," I wanted to expand my research into NFAs and NTMs. NFAs, or nondeterministic finite automata, is a finite automaton where at each step the machine can:

1. Branch into multiple different states for the same input symbol
2. Make transitions without "consuming" output (use epsilons transitions)

Similar to NPDAs, NFAs explore all paths simultaneously. If any path leads to an accepting state, the NFA accepts the input string. A nondeterministic turing machine is similar to NPDAs and NFAs as it also branches into all possible paths simultaneously, accepting any string that provides at least one path to an accepting state.

While NPDAs are strictly more powerful than DPDAs, NFAs and NTMs are both equal in power to their counterparts (DFAs and DTMs). Any language recognized by a NFA can be recognized by an equivalent DFA. Any language recognized by a NTM can also be recognized by a DTM. However, in the case of NFA vs. DFA, NFAs can be more compact than an equivalent DFA. In the case of NTM vs DTM, NTMs can solve problems much faster and efficiently than DTMs. This is especially evident in the case of NP problems. NP problems, or nondeterministic polynomial time problems are "the set of decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time by a deterministic Turing machine, or alternatively the set of problems that can be solved in polynomial time by a nondeterministic Turing machine" (Mao). This definition of the NP complexity class alone illustrates the difference in power between NTMs and DTMs. Additionally, for my curiosity / to better understand, I read up on the P vs NP problem. This

problem asks whether problems verified in polynomial time by DTMs can also be solved by DTMs in polynomial time. If yes, then NP = P. If not, then they do not equal each other. However, the P vs. NP problem has yet to be solved.

Nondeterminism has many practical applications across a wide variety of subjects. The ability to "guess" the correct transitions by simultaneously calculating different paths have helped model parallel computation and parallel processing. Very importantly, nondeterminism has helped inspire search techniques, such as the Minimax Algorithm and A*. Naturally, nondeterminism is also essential to cryptography, optimization, and generative AI. Nondeterminism allows gen-AI LLMs "to generate diverse and creative outputs" (Monnette). Quantum computers even use quantum mechanics to simulate nondeterministic behavior by processing multiple possibilities simultaneously.

Nondeterminism is an increasingly important concept that has inspired a myriad of real-world solutions in fields such as computation & processing, cryptography, and AI. By better understanding nondeterminism and how it works at a lower level in NPDAs, NFAs, and NTMs, I have a newfound appreciation and understanding for this computing breakthrough.

Sources:

Hardesty, Larry. "Explained: P vs. Np." *MIT News | Massachusetts Institute of Technology*, 29 Oct. 2009, news.mit.edu/2009/explainer-pnp.

Mao, Lei. "P vs Np, NP Complete, Np Hard." *Lei Mao's Log Book*, Lei Mao's Log Book, 20 Nov. 2023, leimao.github.io/blog/P-VS-NP/#:~:text=In%20contrast%2C%20a%20NTM%20can%20

find%20such%20a%20route%20as%20follows:&text=NP%20is%20the%20set%20of%20decision%20problems,polynomial%20time%20by%20a%20nondeterministic%20Turing%20machine.

Monnette, Jeffrey. “Gen Ai Is Non-Deterministic: Why It Matters and How It Changes the Way We Work with Software.” *LinkedIn*, 30 Sept. 2024, www.linkedin.com/pulse/gen-ai-non-deterministic-why-matters-how-changes-way-we-monnette-vrldc/.

Moore, Cristopher. *Automata, Languages, and Grammars - the Nature Of …*, 24 Jan. 2015, nature-of-computation.org/supplements/automata-notes.pdf.

“Non-Deterministic Algorithm.” *Engati*, www.engati.com/glossary/non-deterministic-algorithm#:~:text=Non%2Ddeterministic%20algorithms%20are%20essential,for%20finding%20near%2Doptimal%20solutions. Accessed 16 Dec. 2024.

“NP-Complete Isn’t (Always) Hard.” *Hillel Wayne*, 20 Feb. 2023, www.hillelwayne.com/post/np-hard/#:~:text=NP%20is%20the%20set%20of,different%20choice%20in%20each%20timeline.

Sipser, Michael. *Introduction to the Theory of Computation*. Cengage Learning, Inc, 2021.