CS21 Decidability and Tractability

Lecture 8
January 22, 2024

1

NPDA, CFG equivalence

Theorem: a language L is recognized by a NPDA iff L is described by a CFG.

Must prove *two* directions:

(⇒) L is recognized by a NPDA implies L is described by a CFG.

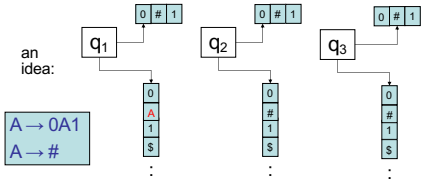
(⇐) L is described by a CFG implies L is recognized by a NPDA.

2

NPDA, CFG equivalence

Proof of (⇐): L is described by a CFG implies L is recognized by a NPDA.

an idea:



3

NPDA, CFG equivalence

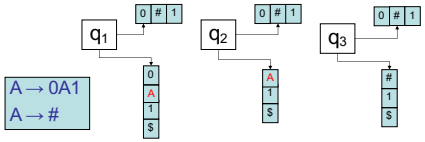
1. we'd like to non-deterministically guess the derivation, forming it on the stack
2. then scan the input, popping matching symbol off the stack at each step
3. accept if we get to the bottom of the stack at the end of the input.

what is wrong with this approach?

4

NPDA, CFG equivalence

– only have access to top of stack
– combine steps 1 and 2:
  • allow to match stack terminals with tape *during* the process of producing the derivation on the stack



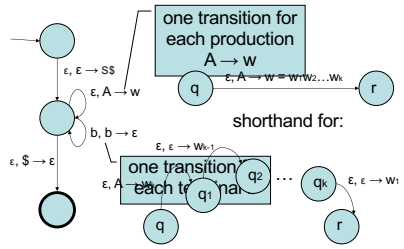
5

NPDA, CFG equivalence

- informal description of construction:
  – place $ and start symbol S on the stack
  – repeat:
    • if the top of the stack is a non-terminal A, pick a production with A on the lhs and substitute the rhs for A on the stack
    • if the top of the stack is a terminal b, read b from the tape, and pop b from the stack.
    • if the top of the stack is $, enter the accept state.

6

NPDA, CFG equivalence



7

NPDA, CFG equivalence

Proof of (⇒): L is recognized by a NPDA implies L is described by a CFG.

- harder direction
- first step: convert NPDA into "normal form":
  - single accept state
  - empties stack before accepting
  - each transition *either* pushes *or* pops a symbol

8

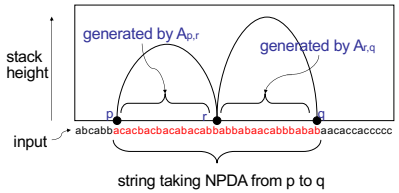
NPDA, CFG equivalence

- **main idea**: non-terminal $A_{p,q}$ generates exactly the strings that take the NPDA from state p (w/ empty stack) to state q (w/ empty stack)
- then $A_{start,\ accept}$ generates all of the strings in the language recognized by the NPDA.

9

NPDA, CFG equivalence

- Two possibilities to get from state p to q:



10

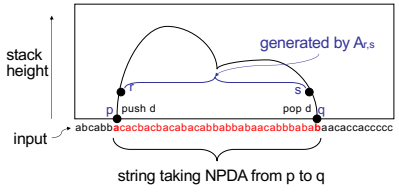
NPDA, CFG equivalence

- NPDA P = (Q, Σ, Γ, δ, start, {accept})
- CFG G:
  - non-terminals $V = \{A_{p,q} : p, q \in Q\}$
  - start variable $A_{start,\ accept}$
  - productions:
    for every p, r, q ∈ Q, add the rule
    $A_{p,q} \rightarrow A_{p,r}A_{r,q}$

11

NPDA, CFG equivalence

- Two possibilities to get from state p to q:



12

NPDA, CFG equivalence

- NPDA P = (Q, Σ, Γ, δ, s
- CFG G:
  - non-terminals V = {A
  - start variable $A_{start, acc}$
  - productions:

from state p, read a, push d, move to state r

from state s, read b, pop d, move to state q

for every p, r, s, q ∈ Q, d ∈ Γ and a, b ∈ (Σ ∪ {ε})
if (r, d) ∈ δ(p, a, ε), and
(q, ε) ∈ δ(s, b, d), add the rule
$A_{p,q} \rightarrow aA_{r,s}b$

13

NPDA, CFG equivalence

- NPDA P = (Q, Σ, Γ, δ, start, {accept})
- CFG G:
  - non-terminals V = $\{A_{p,q} : p, q \in Q\}$
  - start variable $A_{start, accept}$
  - productions:

for every p ∈ Q, add the rule
$A_{p,p} \rightarrow \varepsilon$

14

NPDA, CFG equivalence

- two claims to verify correctness:

1. if $A_{p,q}$ generates string x, then x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack)
2. if x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack), then $A_{p,q}$ generates string x

15

NPDA, CFG equivalence

1. if $A_{p,q}$ generates string x, then x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack)
   - induction on length of derivation of x.
   - base case: 1 step derivation. must have only terminals on rhs. In G, must be production of form $A_{p,p} \rightarrow \varepsilon$.

16

NPDA, CFG equivalence

1. if $A_{p,q}$ generates string x, then x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack)
   - assume true for derivations of length at most k, prove for length k+1.
   - verify case: $A_{p,q} \rightarrow A_{p,r}A_{r,q} \rightarrow^k x = yz$
   - verify case: $A_{p,q} \rightarrow aA_{r,s}b \rightarrow^k x = ayb$

17

NPDA, CFG equivalence

2. if x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack), then $A_{p,q}$ generates string x
   - induction on # of steps in P's computation
   - base case: 0 steps. starts and ends at same state p. only has time to read empty string ε.
   - G contains $A_{p,p} \rightarrow \varepsilon$.

18

NPDA, CFG equivalence

2. if x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack), then $A_{p,q}$ generates string x
   - induction step. assume true for computations of length at most k, prove for length k+1.
   - if stack becomes empty sometime in the middle of the computation (at state r)
     - y is read going from state p to r ($A_{p,r} \rightarrow^* y$)
     - z is read going from state r to q ($A_{r,q} \rightarrow^* z$)
     - conclude: $A_{p,q} \rightarrow A_{p,r}A_{r,q} \rightarrow^* yz = x$

19

NPDA, CFG equivalence

2. if x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack), then $A_{p,q}$ generates string x
   - if stack becomes empty only at beginning and end of computation.
     - first step: state p to r, read a, push d
     - go from state r to s, read string y ($A_{r,s} \rightarrow^* y$)
     - last step: state s to q, read b, pop d
     - conclude: $A_{p,q} \rightarrow aA_{r,s}b \rightarrow^* ayb = x$

20

NPDA, CFG equivalence

2. if x can take NPDA P from state p (w/ empty stack) to q (w/ empty stack), then $A_{p,q}$ generates string x
   - if stack becomes empty only at beginning and end of computation.
     - first step: state p to r, read a, push d
     - go from state r to s, read string y ($A_{r,s} \rightarrow^* y$)
     - last step: state s to q, read b, pop d
     - conclude: $A_{p,q} \rightarrow aA_{r,s}b \rightarrow^* ayb = x$

21

Chomsky Normal Form

- Useful to deal only with CFGs in a simple normal form
- Most common: Chomsky Normal Form (CNF)
- Definition: every production has form

  $A \rightarrow BC$ or $S \rightarrow \varepsilon$ or $A \rightarrow a$

  where A, B, C are any non-terminals (and B, C are not S) and a is any terminal.

22

Chomsky Normal Form

Theorem: Every CFL is generated by a CFG in Chomsky Normal Form.

Proof: exercise or in book…

23

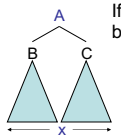
Deciding CFLs

- Useful to have an efficient algorithm to decide whether string x is in given CFL
  - e.g. programming language often described by CFG. Determine if string is valid program.
- If CFL recognized by deterministic PDA, just simulate the PDA.
  - but not all CFLs are (homework)…
- Can simulate NPDA, but this takes exponential time in the worst case.

24

Deciding CFLs

- Convert CFG into Chomsky Normal Form
- parse tree for string x generated by nonterminal A:



If $A \rightarrow^k x$ $(k > 1)$ then there must be a way to split x:

$x = yz$

- $A \rightarrow BC$ is a production and
- $B \rightarrow^i y$ and $C \Rightarrow^j z$ for $i, j < k$

Deciding CFLs

- An algorithm:

IsGenerated(x, A)

```
if |x| = 1, then return YES if A → x is a production,
  else return NO
for all n-1 ways of splitting x = yz
  for all ≤ m productions of form A → BC
    if IsGenerated(y, B) and IsGenerated(z, C),
          return YES
return NO
```

- worst case running time?