CSCB36

REVIEW SEMINAR!

CFG Review

$$S \to ABC$$
$$A \to 11A, \epsilon$$
$$B \to 0B0, 0$$
$$C \to C10, C01, 1$$

A CFG G: 4-tuple (V, Σ, P, S), with

- V: A finite set of variables
- Σ: A finite set of terminals
- P: A finite set of productions (A -> b, A∈V, b∈(V∪Σ)*)
- S ∈ V: The start variable

- L(G): the language of all strings derived from S
- A language L' is said to be context-free (CFL) if L(G) = L' for some CFG G

CFG Review

A CFG $G = (V, \Sigma, P, S)$ is *right linear* if every production in P has one of the following forms:

1. $A \to \epsilon$
2. $A \to xB$

where $A, B \in V$ and $x \in \Sigma^*$

G is *strict right-linear* if every production in P has one of following forms:

1. $A \to \epsilon$
2. $A \to xB$

where $A, B \in V$, $x \in \Sigma^*$, and $|x| \leq 1$.

$A \to 0 \mid 0B$

$A \to 0C$

$C \to 1D$

$D \to 0B$

CFG Review

- CFG are more powerful than Regular Expressions
- In fact, regular expressions are a subset of languages generated by CFG
  - THE BIG RESULT PLUS!
  - The following are equivalent
    - L = L(R) for some regular expression R
    - L = L(G) for some right-linear CFG G
    - L = L(G) for some strict right-linear CFG G

CFG Question

5. **[8 marks]** Let $\Sigma = \{0, 1\}$. Let $L_5 = \{x \in \Sigma^* : \#_{00}(x) = \#_{110}(x)\}$.

Here is Nick's design, with 7 variables, for a CFG that generates L_5.

S generates L_5.
A_{00} generates $\{x \in L_5 : x$ starts with 0, x ends with 0$\}$.
A_{01} generates $\{x \in L_5 : x$ starts with 0, x ends with 1$\}$.
A_{10} generates $\{x \in L_5 : x$ starts with 1 but not 10, x ends with 0$\}$.
A_{11} generates $\{x \in L_5 : x$ starts with 1 but not 10, x ends with 1$\}$.
$A_{10:0}$ generates $\{x \in L_5 : x$ starts with 10, x ends with 0$\}$.
$A_{10:1}$ generates $\{x \in L_5 : x$ starts with 10, x ends with 1$\}$.

Please help Nick get started on his CFG by using the left-to-right (L2R) method to "code" the productions for the variables listed below.

$S \to$

$A_{00} \to$

$A_{10} \to$

$A_{101} \to$

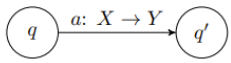
CFG Question – SOLUTION

- $S \rightarrow A_{00} \mid A_{01} \mid A_{10} \mid A_{11} \mid A_{10:0} \mid A_{10:1} \mid \varepsilon$
- $A_{00} \rightarrow 0A_{01}A_{10:0} \mid 0A_{10} \mid 0A_{10:0} \mid 0$
- $A_{10} \rightarrow 1A_{10} \mid 1A_{10:0}A_{00}$
- $A_{10:1} \rightarrow 1A_{01}$

PDA Definition

A *pushdown automaton* (PDA) is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where

- Q is a set of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $\delta\colon Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to \mathcal{P}\Big(Q \times (\Gamma \cup \{\epsilon\})\Big)$ is the transition function; each label is of the form



  where $a \in \Sigma \cup \{\epsilon\}$, $X, Y \in \Gamma \cup \{\epsilon\}$, and $q, q' \in Q$. There are four cases with X and Y:

  1. $X = \epsilon$ and $Y \in \Gamma$: push Y onto the stack
  2. $X \in \Gamma$ and $Y = \epsilon$: pop X from the stack
  3. $X, Y \in \Gamma$: replace X with Y at the top of the stack
  4. $X = Y = \epsilon$: perform no action on the stack
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accepting states

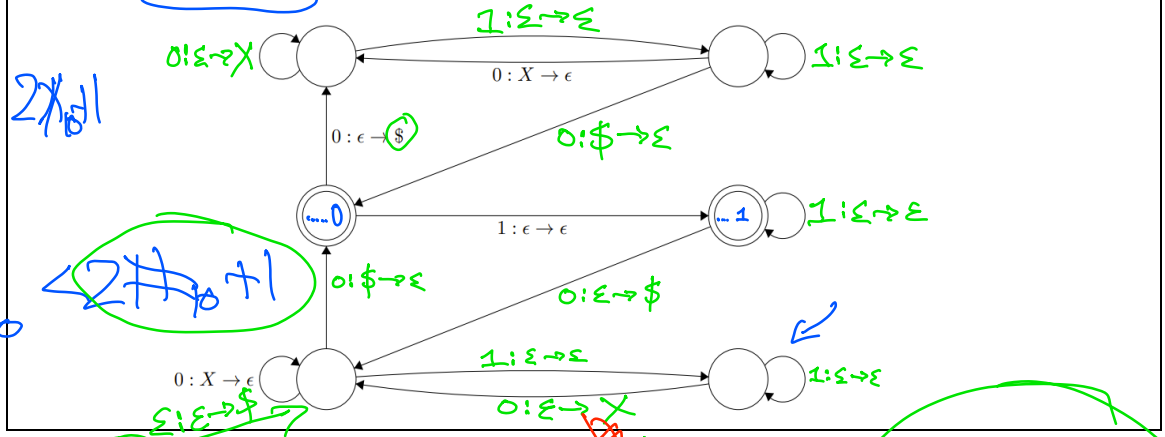
PDA Review

- A PDA accepts $x \in \Sigma^*$ if by starting with an empty stack, there is a sequence of transitions that:
  - Reads all of x
  - Ends in an accepting state
  - Leaves the stack empty
- CFG and PDA are equally as powerful
  - A language is accepted by some CFG if and only if it is accepted by some PDA

PDA Question

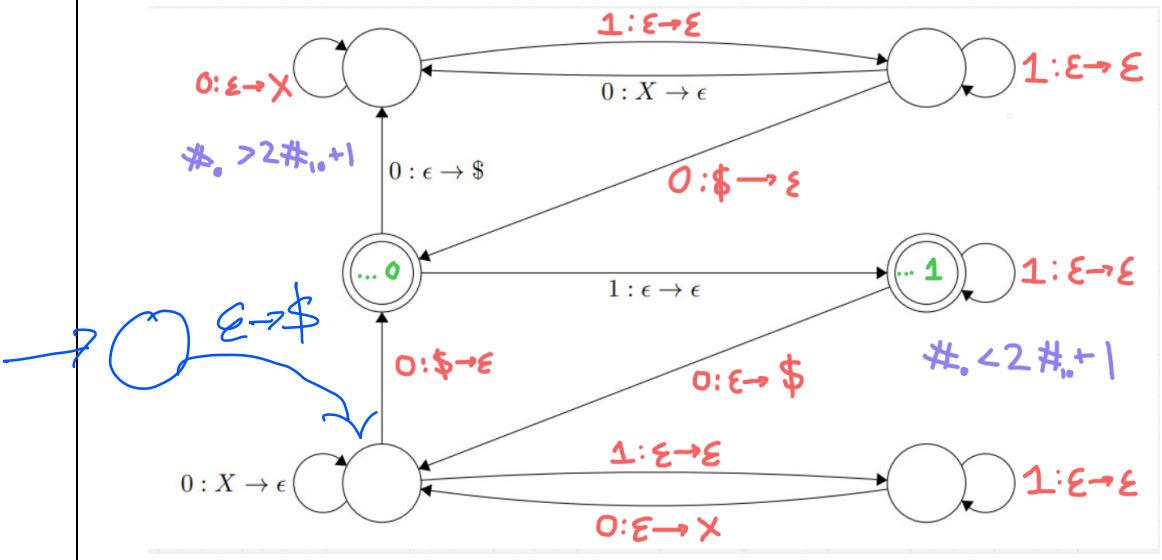
6. **[7 marks]** Let $\Sigma = \{0, 1\}$. Let $L_6 = \{x \in \Sigma^* : \#_0(x) = 2\#_{10}(x) + 1\}$.
For your information, if $x = 10$, then $\#_0(x) = \#_{10}(x) = 1$. Also, $1000, 0100, 0010 \in L_6$.

Nick created a DPDA that accepts L_6. Before anyone saw it, the **Automaton Mangler** struck! Destroyed beyond recognition were the initial state plus one transition from it, and all except 4 transition labels. Here is what remains of Nick's DPDA.



9

PDA Question – SOLUTION



CNF/DNF Review

- Literal: a single propositional variable or its negation

  x, y, ¬x, ¬z

- Minterm: a literal, or a conjunction (∧) of minterms

  x, (a ∧ ¬b ∧ c), f(x) ∧ g(x, y)

- DNF: a disjunction (∨) of minterms

  (a ∧ ¬b ∧ c) ∨ (a ∧ b) ∨ (¬c)

- Maxterm: a literal, or a disjunction (∨) of maxterms

  x, (a ∨ ¬b ∨ c), f(x) ∨ g(x, y)

- CNF: a conjunction (∧) of minterms

  (a ∨ ¬b ∨ c) ∧ (a ∨ b) ∧ (¬c)

CNF/DNF Review

- Every propositional formula can be converted into a CNF and a DNF.

x	y	(x -> y) ∧ y
0	0	1 **0**
0	1	1 **1**
1	0	0 **0**
1	1	1 **1**

Equivalent to:
"Either x is false and y is true, or x and y are both true"
(¬x ∧ y) ∨ (x ∧ y)
DNF

Equivalent to:
"It is NOT the case that (x and y are both false, or x is true and y is false)"
¬((¬x ∧ ¬y) ∨ (x ∧¬ y))
¬(¬x ∧ ¬y) ∧ ¬(x ∧¬y)
(x ∨ y) ∧ (¬x ∨ y)
CNF

$(x \rightarrow y) \wedge y$

$(\neg x \vee y) \wedge y \rightarrow$ CNF

PNF Review

Definition 6.13 *A first-order formula is in* ***Prenex Normal Form (PNF)*** *if and only if it is of the form*

$$\mathbf{Q}_1 x_1 \mathbf{Q}_2 x_2 \ldots \mathbf{Q}_k x_k \, E$$

where $k \geq 0$, *each* $\mathbf{Q}_i$ *is a quantifier* ($\forall$ *or* $\exists$) *and* E *is a* quantifier-free *first-order formula.*

Informally: a formula is in PNF if all the quantifiers are in the front, followed by only one formula.

∀x∀y (P(x) -> Q(y)) is in PNF.

∀x, P(x) -> ∀y, Q(y) is not in PNF.

Converting to PNF is simple:

1. Look up your equivalence laws.
2. Apply your equivalence laws until the equation looks like a PNF.

Duality of quantifiers:

I. $\neg \mathbf{Q}x\ F$ LEQV $\overline{\mathbf{Q}}x\ \neg F$

Factoring quantifiers:

IIa. $E \wedge \mathbf{Q}x\ F$ LEQV $\mathbf{Q}x\ (E \wedge F)$, if x is not free in E

IIb. $E \vee \mathbf{Q}x\ F$ LEQV $\mathbf{Q}x\ (E \vee F)$, if x is not free in E

IIc. $\mathbf{Q}x\ E \wedge F$ LEQV $\mathbf{Q}x\ (E \wedge F)$, if x is not free in F

IId. $\mathbf{Q}x\ E \vee F$ LEQV $\mathbf{Q}x\ (E \vee F)$, if x is not free in F

IIe. $\mathbf{Q}x\ E \to F$ LEQV $\overline{\mathbf{Q}}x\ (E \to F)$, if x is not free in F

III. $E \to \mathbf{Q}x\ F$ LEQV $\mathbf{Q}x\ (E \to F)$, if x is not free in E

Renaming of quantified variables:

III. $\mathbf{Q}x\ F$ LEQV $\mathbf{Q}y\ F^x_y$, if y does not occur in F

PWF

$\exists x, (P(x) \rightarrow Q(y))$

Free variables

$(\exists x, \underbrace{P(x)}_{①}) \rightarrow \underbrace{Q(y)}_{②}$

$\Updownarrow$

✗ NOT

$\forall x, (P(x) \rightarrow Q(y))$

✓ PWF

PNF/CNF/DNF practice

Consider the following statement: "If someone has a birthday today, then someone got a gift from all of their friends".

Let $B(x)$ represent "x has a birthday", $F(x, y)$ mean "x and y are friends" with $F(x, y) \leftrightarrow F(y, x)$, and $G(x, y)$ represent "x gave y a gift".

1. Represent the above statement as a propositional formula.
2. Rewrite the formula to be in PNF. (Make sure to cite your steps!)
3. Rewrite the unquantified part of the PNF formula as a DNF and a CNF.
4. Create an interpretation with domain $\{a, b, c\}$ that satisfies the formula.

$\exists x \forall y, \boxed{h(x) \rightarrow j(y)}$



PNF/CNF/DNF solution (1)

Consider the following statement: "If someone has a birthday today, then someone got a gift from all of their friends".

1. $(\exists x, B(x)) \rightarrow (\exists x \forall y, F(x, y) \rightarrow G(y, x))$

Some notes on this formula:

(a) Careful: the x of $B(x)$ is NOT the same x as in $F(x, y) \rightarrow G(y, x)$!

(b) You could have said $F(x, y)$ or $F(y, x)$; both are valid.

(c) Do note that it's $G(y, x)$ and not $G(x, y)$. y is giving x a gift, not (necessarily) vice versa.

PNF/CNF/DNF solution (2)

2.

$$
\begin{aligned}
& (\exists x, B(x)) \to (\exists x \forall y, F(x,y) \to G(y,x)) && \\
\text{leqv } & (\exists z, B(z)) \to (\exists x \forall y, F(x,y) \to G(y,x)) && \text{III} \\
\text{leqv } & \forall z, (B(z) \to (\exists x \forall y, F(x,y) \to G(y,x))) && \text{IIe} \\
\text{leqv } & \forall z \exists x \forall y, (B(z) \to (F(x,y) \to G(y,x))) && \text{IIf (twice)}
\end{aligned}
$$

which is in PNF, as required.

PNF/CNF/DNF solution (3)

Step 1. Make a truth table

3. For brevity, let b, f, g represent $B(z), F(x, y), G(y, x)$ respectively.

b	f	g	$b \rightarrow$	$(f \rightarrow g)$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

PNF/CNF/DNF solution (3)

b	f	g	$b \rightarrow$	$(f \rightarrow g)$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

DNF formula: we can just read this one off the truth table.

$$(\neg b \wedge \neg f \wedge \neg g) \vee$$
$$(\neg b \wedge \neg f \wedge g) \vee$$
$$(\neg b \wedge f \wedge \neg g) \vee$$
$$(\neg b \wedge f \wedge g) \vee$$
$$(b \wedge \neg f \wedge \neg g) \vee$$
$$(b \wedge \neg f \wedge g) \vee$$
$$(b \wedge f \wedge g)$$

CNF formula

$$\neg(b \wedge f \wedge \neg g)$$
$$\text{leqv}(\neg b \vee \neg f \vee g)$$

PNF/CNF/DNF solution (4)

$\{a, b, c\}$

$(\exists x, B(x)) \rightarrow (\exists x \forall y, F(x, y) \rightarrow G(y, x))$

T F T

4. There are many interpretations, but one would be:

$B(a)$	$F(a, b)$	$G(a, b)$
$\neg B(b)$	$F(b, c)$	$\neg G(b, a)$
$\neg B(c)$	$F(a, c)$	$G(b, c)$
		$G(c, b)$
		$G(a, c)$
		$G(c, a)$

- it's a's birthday
- b got a gift from all their friends

Then notice:

(a) Everyone is friends.

(b) Everyone got each other a gift, except b who didn't get a a gift.

(c) It's a's birthday.

T ⇒ T ⇐⇒ T

Sets of Connectives Review

- **Completeness for a set of connectives**

  A set C of connectives is said to be *complete* iff every boolean function can be represented by a propositional formula that uses only connectives in C. From the course notes, we have $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ as examples of complete sets.

  Note:

  Any formula that uses *no connectives at all* also uses only connectives in *any* set of connectives.
  E.g., the formula x uses only connectives in $\{\wedge, \vee\}$.

  Abbreviation:

  We use *uoc* as an abbreviation for *uses only connectives in.*
  E.g., "F uoc C" means "F uses only connectives in C".

Proving Sets of Connectives are Logically Complete Review

Here are the steps to formally prove that a set C is complete:

1. Use structural induction to define the set G that uoc {¬,∧} or {¬,∨} (the choice is yours; either is acceptable).

2. Use structural induction to prove that for every formula F ∈ G, there exists a formula F ' such that F ' uoc C and F ' leqv F.

3. Our result follows from the fact that {¬,∧} (or {¬,∨} if you chose it) is complete.

Proving Sets of Connectives are Logically NOT Complete Review

Here are the steps to formally prove that a set C is NOT complete:

1. Use structural induction to define the set H of formulas that uoc C.
2. Define a predicate P(F) that holds for every F ∈ H, but not in general.
3. Use structural induction to prove that P(F) holds for every formula F ∈ H.
4. Give a specific formula F and show that P(F) does not hold. Then our result follows as argued above.

Proving/Disproving Sets of Connectives are Logically Complete Practice

(a) Determine whether the set of connectives

$$C = \{\rightarrow, \neg\}$$

is complete.

(b) The unary connective 0 is defined by

$$\forall \tau,\ \tau^*(0P) = 0.$$

Determine whether the set of connectives

$$C = \{0, \wedge\}$$

is complete.