CSE 311 Section 1

Propositional Logic

Announcements & Reminders

- Sections are Graded
    - You will be graded on section participation, so please try to come ☺
    - If you cannot attend you will need to submit ALL the section problems to gradescope by 6:00 pm on the day after section
- HW1 due Friday **(1/16)** @ **11:00 PM** on Gradescope
    - Remember, you only have 3 late days to use throughout the quarter
    - You can use only 1 late days on any 1 assignment
- Check the course website for OH times!
- Concept Checks!
    - Absolute deadline on the day after the lecture is given @ **11:00 pm**

Problem 1

Problem 1a – Stop, Prop, and Roll

Steps:

1. Create propositional variables
2. Replace all propositions with created variables
3. Replace the operators

a) If I haven't had my coffee and the sun is not up, then I am angry. But if I have had my coffee or the sun is up, then I am happy.

Problem 1a – Stop, Prop, and Roll

Steps:

1. Create propositional variables
2. Replace all propositions with created variables
3. Replace the operators

a) If I haven't had my coffee and the sun is not up, then I am angry. But if I have had my coffee or the sun is up, then I am happy.

Step 1

p: I have had my coffee
q: The sun is up
r: I am angry
s: I am happy

Problem 1a – Stop, Prop, and Roll

Steps:

1. Create propositional variables
2. Replace all propositions with created variables
3. Replace the operators

a) If I haven't had my coffee and the sun is not up, then I am angry. But if I have had my coffee or the sun is up, then I am happy.

Step 1

p: I have had my coffee
q: The sun is up
r: I am angry
s: I am happy

Step 2

If not *p* and not *q*, **then** *r*. **If** *p* or *q*, **then** *s*.

Problem 1a – Stop, Prop, and Roll

Steps:

1. Create propositional variables
2. Replace all propositions with created variables
3. Replace the operators

a) If I haven't had my coffee and the sun is not up, then I am angry. But if I have had my coffee or the sun is up, then I am happy.

Step 1

p: I have had my coffee
q: The sun is up
r: I am angry
s: I am happy

Step 2

If not *p* and not *q*, **then** *r*. **If** *p* or *q*, **then** *s*.

Step 3

$((\neg p \wedge \neg q) \rightarrow r) \wedge ((p \vee q) \rightarrow s)$

Problem 1 – Stop, Prop, and Roll

1. Create propositional variables
2. Replace all propositions with created variables

2b. Convert the sentence to an “if then” statement

3. Replace the operators

b)

i) If the stack is empty, you can push but not pop.

ii) If the stack is full, you can pop but not push.

iii) If the stack is neither full nor empty, you can both pop and push.

c)

i) You can have your cake or you can eat your cake, but not both.

ii) If you have your cake and you drop your cake, then you are sad and you don’t have your cake. But if you eat your cake and are sad, then you don’t have your cake.

Work on parts b and c with the people around you, and then we’ll go over it together!

Problem 1 – Stop, Prop, and Roll

i) If the stack is empty, you can push but not pop.
ii) If the stack is full, you can pop but not push.
iii) If the stack is neither full nor empty, you can both pop and push.

Step 1
a: The stack is empty.
b: The stack is full.
c: You can push.
d: You can pop.

1. Create propositional variables
2. Replace all propositions with created variables

2b. Convert the sentence to an “if then” statement

3. Replace the operators

Problem 1 – Stop, Prop, and Roll

1. Create propositional variables
2. Replace all propositions with created variables

2b. Convert the sentence to an “if then” statement

3. Replace the operators

i) If the stack is empty, you can push but not pop.
ii) If the stack is full, you can pop but not push.
iii) If the stack is neither full nor empty, you can both pop and push.

Step 1

a: The stack is empty.
b: The stack is full.
c: You can push.
d: You can pop.

Step 2

i) If a, then c and not d
ii) If b, then d and not c
iii) If not a and not b, then c and d

Problem 1 – Stop, Prop, and Roll

1. Create propositional variables
2. Replace all propositions with created variables

2b. Convert the sentence to an “if then” statement

3. Replace the operators

i) If the stack is empty, you can push but not pop.
ii) If the stack is full, you can pop but not push.
iii) If the stack is neither full nor empty, you can both pop and push.

Step 1

a: The stack is empty.
b: The stack is full.
c: You can push.
d: You can pop.

Step 2

i) If a, then c and not d
ii) If b, then d and not c
iii) If not a and not b, then c and d

Step 3

i) $a \rightarrow (c \land \neg d)$
ii) $b \rightarrow (d \land \neg c)$
iii) $(\neg a \land \neg b) \rightarrow (c \land d)$ *OR* $\neg(a \lor b) \rightarrow (c \land d)$

Problem 1 – Stop, Prop, and Roll

i) You can have your cake or you can eat your cake, but not both.
ii) If you have your cake and you drop your cake, then you are sad and you don't have your cake. But if you eat your cake and are sad, then you don't have your cake.

Step 1
a: You can have your cake.
b: You can eat your cake.
c: You drop your cake.
d: You are sad.

1. Create propositional variables
2. Replace all propositions with created variables

2b. Convert the sentence to an "if then" statement

3. Replace the operators

Problem 1 – Stop, Prop, and Roll

1. Create propositional variables
2. Replace all propositions with created variables

2b. Convert the sentence to an “if then” statement

3. Replace the operators

i) You can have your cake or you can eat your cake, but not both.
ii) If you have your cake and you drop your cake, then you are sad and you don’t have your cake. But if you eat your cake and are sad, then you don’t have your cake.

Step 1
a: You can have your cake.
b: You can eat your cake.
c: You drop your cake.
d: You are sad.

Step 2
i) a or b, but not both
ii) If a and c, then d and not a. If b and d, then not a.

Problem 1 – Stop, Prop, and Roll

1. Create propositional variables
2. Replace all propositions with created variables

2b. Convert the sentence to an "if then" statement

3. Replace the operators

i) You can have your cake or you can eat your cake, but not both.
ii) If you have your cake and you drop your cake, then you are sad and you don't have your cake. But if you eat your cake and are sad, then you don't have your cake.

Step 1
a: You can have your cake.
b: You can eat your cake.
c: You drop your cake.
d: You are sad.

Step 2
i) a or b, but not both
ii) If a and c, then d and not a. If b and d, then not a.

Step 3
i) $a \oplus b$
ii) $((a \land c) \rightarrow (d \land \neg a)) \land ((b \land d) \rightarrow \neg a)$

Problem 2

Problem 2

```
public static boolean E(boolean a, boolean b, boolean c, boolean d) {
  if (!(a || b))
    return false;
  if (!(!a || !b))
    return false;
  if (!(a || c))
    return false;
  return (b || !d);
}
```

Or, equivalently:

```
return (a || b) && (!a || !b) && (a || c) && (b || !d);
```

Both calculate the CNF (AND of ORs) expression for

$(a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee c) \wedge (b \vee \neg d)$

Problem 2

a) Write a truth table for *E*. Include columns for *a, b, c, d*, all four disjunctions, and *E*.
E calculates $(a \lor b) \land (\neg a \lor \neg b) \land (a \lor c) \land (b \lor \neg d)$.

b) Write the **canonical** DNF expression for *E*.

c) Translate your DNF expression into a new Java implementation of *E*.

Work on problem 2 with the people around you, and then we'll go over it together!

Problem 2

Write a truth table for *E*. Include columns for *a, b, c, d*, all four disjunctions, and *E*. *E* calculates $(a \lor b) \land (\neg a \lor \neg b) \land (a \lor c) \land (b \lor \neg d)$.

a	b	$a \lor b$	$\neg a \lor \neg b$
F	F		
F	T		
T	F		
T	T		

a	c	$a \lor c$
F	F	
F	T	
T	F	
T	T	

b	d	$b \lor \neg d$
F	F	
F	T	
T	F	
T	T	

Problem 2

Write a truth table for *E*. Include columns for *a, b, c, d*, all four disjunctions, and *E*. *E* calculates $(a \lor b) \land (\neg a \lor \neg b) \land (a \lor c) \land (b \lor \neg d)$.

a	b	$a \lor b$	$\neg a \lor \neg b$
F	F	F	T
F	T	T	T
T	F	T	T
T	T	T	F

a	c	$a \lor c$
F	F	F
F	T	T
T	F	T
T	T	T

b	d	$b \lor \neg d$
F	F	T
F	T	F
T	F	T
T	T	T

Problem 2

a	*b*	*c*	*d*	*a* ∨ *b*	¬*a* ∨ ¬*b*	*a* ∨ *c*	*b* ∨ ¬*d*	*E*
F	F	F	F	F	T	F	T	F
F	F	F	T	F	T	F	F	F
F	F	T	F	F	T	T	T	F
F	F	T	T	F	T	T	F	F
F	T	F	F	T	T	F	T	F
F	T	F	T	T	T	F	T	F
F	T	T	F	T	T	T	T	T
F	T	T	T	T	T	T	T	T
T	F	F	F	T	T	T	T	T
T	F	F	T	T	T	T	F	F
T	F	T	F	T	T	T	T	T
T	F	T	T	T	T	T	F	F
T	T	F	F	T	F	T	T	F
T	T	F	T	T	F	T	T	F
T	T	T	F	T	F	T	T	F
T	T	T	T	T	F	T	T	F

Problem 2

a	b	c	d	$a \lor b$	$\neg a \lor \neg b$	$a \lor c$	$b \lor \neg d$	E
F	F	F	F	F	T	F	T	F
F	F	F	T	F	T	F	F	F
F	F	T	F	F	T	T	T	F
F	F	T	T	F	T	T	F	F
F	T	F	F	T	T	F	T	F
F	T	F	T	T	T	F	T	F
F	T	T	F	T	T	T	T	T
F	T	T	T	T	T	T	T	T
T	F	F	F	T	T	T	T	T
T	F	F	T	T	T	T	F	F
T	F	T	F	T	T	T	T	T
T	F	T	T	T	T	T	F	F
T	T	F	F	T	F	T	T	F
T	T	F	T	T	F	T	T	F
T	T	T	F	T	F	T	T	F
T	T	T	T	T	F	T	T	F

True Rows

Problem 2

b) Write the **canonical** DNF (OR of ANDs) expression for *E*.

a	***b***	***c***	***d***	***E***
F	T	T	F	T
F	T	T	T	T
T	F	F	F	T
T	F	T	F	T

Problem 2

b) Write the **canonical** DNF expression for *E*.

a	*b*	*c*	*d*	*E*
F	T	T	F	T
F	T	T	T	T
T	F	F	F	T
T	F	T	F	T

$(\neg a \land b \land c \land \neg d)$

$\lor$

$(\neg a \land b \land c \land d)$

$\lor$

$(a \land \neg b \land \neg c \land \neg d)$

$\lor$

$(a \land \neg b \land c \land \neg d)$

Problem 2

c) Translate your DNF expression into a new Java implementation of *E*.

$(\neg a \land b \land c \land \neg d) \lor (\neg a \land b \land c \land d) \lor (a \land \neg b \land \neg c \land \neg d) \lor (a \land \neg b \land c \land \neg d)$

Problem 2

c) Translate your DNF expression into a new Java implementation of *E*.

$(\neg a \land b \land c \land \neg d) \lor (\neg a \land b \land c \land d) \lor (a \land \neg b \land \neg c \land \neg d) \lor (a \land \neg b \land c \land \neg d)$

```
public static boolean E(boolean a, boolean b, boolean c, boolean d) {
  if (!a && b && c && !d)
    return true;
  if (!a && b && c && d)
    return true;
  if (a && !b && !c && !d)
    return true;
  return a && !b && c && !d;
}
```

Problem 3

Predicates & Quantifiers

- **Predicate**: A function that outputs booleans.
  - Red(x) outputs true if x is red
  - EqualTo(x, y) outputs true if x is equal to y
- **Domain of Discourse**: The types of things that can be inputs to a predicate.
  - Integers, real numbers, colors, mammals, students, etc
- **Quantifiers**
  - Universal Quantifier $\forall x\ P(x)$: For all x, P(x) is true. *(In latex, \forall)*
  - Existential Quantifier $\exists x\ P(x)$” There exists an x, such that P(x) is true. *(In latex, \exists)*
- **Domain Restrictions**
  - When restricting $\forall$, add the domain restriction as the hypothesis of an implication
  - When restricting $\exists$, AND the domain restriction with the statement

Problem 3 – Predicates

CS(x) returns true if and only if x majors in CS

CE(x) returns true if and only if x majors in CE

CSE(y) returns true if and only if y is a CSE class

MATH(y) returns true if and only if y is a MATH class

Wants(x,y) returns true if and only if x wants to take y

Likes(x,y) returns true if and only if x likes y

HasToTake(x,y) returns true if and only if x has to take y

a) $\neg \exists x\, (CS(x) \wedge CE(x))$
b) $\exists x\, (CS(x) \wedge \exists y(CSE(y) \wedge \neg HasToTake(x,y) \wedge Likes(x,y)))$
c) $\forall x\, (CE(x) \rightarrow \exists y\, (MATH(y) \wedge HasToTake(x,y)))$
d) $\exists x\, ((CS(x) \vee CE(x)) \wedge \forall y\, (CSE(y) \rightarrow Wants(x,y)))$

Problem 3 – Predicates

a) $\neg \exists x\, (CS(x) \land CE(x))$

b) $\exists x\, (CS(x) \land \exists y (CSE(y) \land \neg HasToTake(x,y) \land Likes(x,y)))$

c) $\forall x\, (CE(x) \rightarrow \exists y\, (MATH(y) \land HasToTake(x,y)))$

d) $\exists x\, ((CS(x) \lor CE(x)) \land \forall y\, (CSE(y) \rightarrow Wants(x,y)))$

Problem 3 – Predicates

a) $\neg \exists x\ (CS(x) \land CE(x))$

There is no student that majors in both CS and CE.

b) $\exists x\ (CS(x) \land \exists y(CSE(y) \land \neg HasToTake(x,y) \land Likes(x,y)))$

c) $\forall x\ (CE(x) \rightarrow \exists y\ (MATH(y) \land HasToTake(x,y)))$

d) $\exists x\ ((CS(x) \lor CE(x)) \land \forall y\ (CSE(y) \rightarrow Wants(x,y)))$

Problem 3 – Predicates

a) $\neg \exists x\, (CS(x) \land CE(x))$

There is no student that majors in both CS and CE.

b) $\exists x\, (CS(x) \land \exists y (CSE(y) \land \neg HasToTake(x,y) \land Likes(x,y)))$

There is a CS student who likes a CSE class they don't have to take.

c) $\forall x\, (CE(x) \rightarrow \exists y\, (MATH(y) \land HasToTake(x,y)))$

d) $\exists x\, ((CS(x) \lor CE(x)) \land \forall y\, (CSE(y) \rightarrow Wants(x,y)))$

Problem 3 – Predicates

a) $\neg \exists x\, (CS(x) \land CE(x))$

There is no student that majors in both CS and CE.

b) $\exists x\, (CS(x) \land \exists y(CSE(y) \land \neg HasToTake(x,y) \land Likes(x,y)))$

There is a CS student who likes a CSE class they don't have to take.

c) $\forall x\, (CE(x) \rightarrow \exists y\, (MATH(y) \land HasToTake(x,y)))$

All CE students have a MATH class they have to take.

d) $\exists x\, ((CS(x) \lor CE(x)) \land \forall y\, (CSE(y) \rightarrow Wants(x,y)))$

Problem 3 – Predicates

a) $\neg \exists x\, (CS(x) \land CE(x))$

There is no student that majors in both CS and CE.

b) $\exists x\, (CS(x) \land \exists y (CSE(y) \land \neg HasToTake(x,y) \land Likes(x,y)))$

There is a CS student who likes a CSE class they don't have to take.

c) $\forall x\, (CE(x) \rightarrow \exists y\, (MATH(y) \land HasToTake(x,y)))$

All CE students have a MATH class they have to take.

d) $\exists x\, ((CS(x) \lor CE(x)) \land \forall y\, (CSE(y) \rightarrow Wants(x,y)))$

There is a student who majors in CS or CE and wants to take all CSE classes.

That’s All, Folks!

Thanks for coming to section this week!
Any questions?