CS 245

Notes

We have used statements of the form $\neg(A \vee B) \models\models \neg A \wedge \neg B$ as a shorthand for

$\neg(A \vee B) \models \neg A \wedge \neg B$

and

$\neg A \wedge \neg B \models \neg(A \vee B)$

It is sometimes useful to think of algebraic reasoning styles when developing reasoning tools for propositional logic.

Here we consider reasoning techniques designed to show equivalences between formulae.

It is typical in this style of reasoning to write, $A \equiv B$, formulae A and B are semantically equivalent.

Def. For $A, B \in Form(L^p)$,
$A \equiv B$ if for all valuations,
t, $A^t = B^t$.

It is not difficult to show

that $A \models B$ iff $A \equiv B$.

Algebraic reasoning styles have inspired so called 'laws' of propositional logic.

Commutativity: $(A \wedge B) \equiv (B \wedge A)$

$(A \vee B) \equiv (B \vee A)$

Associativity:

$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$$

$$A \vee (B \vee C) \equiv (A \vee B) \vee C$$

Idempotence

$$(A \vee A) \equiv A$$

$$(A \wedge A) \equiv A$$

Double Negation:

$$\neg(\neg A) \equiv A$$

Distribution:

$$(A \vee (B \wedge C)) \equiv (A \vee B) \wedge (A \vee C)$$

$$(A \wedge (B \vee C)) \equiv (A \wedge B) \vee (A \wedge C)$$

De Morgan's Laws:

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$

Simplification (Absorbtion):

$$(A \wedge 1) \equiv A$$

$$(A \vee 1) \equiv 1$$

$$(A \wedge 0) \equiv 0$$

$$(A \vee 0) \equiv A$$

Simplification:

$$A \vee (A \wedge B) \equiv A$$

$$A \wedge (A \vee B) \equiv A$$

Implication:

$$A \rightarrow B \equiv (\neg A) \vee B$$

Contra positive:

$$A \rightarrow B \equiv (\neg B) \rightarrow (\neg A)$$

Equivalence:

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$

(Law of the) Excluded Middle:

$$A \vee (\neg A) \equiv \underline{1}$$

Contradiction: $A \wedge (\neg A) \equiv 0$

From the def. of $\equiv$ we get the following useful lemma that shows that if $A, B \in \text{Form}(L^P)$ and $A \equiv B$ then replacing A with B does not alter the semantic meaning.

Lemma: Let $A, B \in Form(L^p)$ where $A \equiv B$. Then for $C \in Form(L^p)$

$$(A * C) \equiv (B * C).$$

Example: $p \equiv p$ (from def. of $\equiv$)

$p \equiv (p \vee p)$ (idempotence)

$p \equiv (\neg\neg p) \vee p)$ (since $p \equiv \neg\neg p$)

$p \equiv \neg p \rightarrow p$ (implication)

Example:

$$(p \wedge q) \vee (q \wedge r) \stackrel{?}{\equiv} q \wedge (p \vee r)$$

$$(p \wedge q) \vee (q \wedge r) \equiv (q \wedge p) \vee (q \wedge r)$$

commutativity

$$\equiv q \wedge (p \vee r)$$

distributivity

Consider the following code fragment

```
if ((input > 0) OR NOT output) {
  if (NOT (output AND (queuelength < 100))){
        P1
  } else if (output AND NOT (queuelength < 100)){
        P2
  }
  else {P3}
}else {P4}
```

When does each piece of code execute?

Let p: input > 0

q: queue length < 100

r: output

p	q	r	$p \vee r$	$\neg(q \wedge r)$	$(q \vee r) \wedge \neg q$	action
1	1	1	1	0	0	P_3
1	1	0	1	1	1	P_1
1	0	1	1	1	0	P_1
1	0	0	1	1	0	P_1
0	1	1	0	0	0	P_4
0	1	0	0	1	1	P_4
0	0	1	1	1	0	P_1
0	0	0	1	1	0	P_1

Conclusion: P_2 is never executed.

The conditions under which the code fragment P_2 is to be executed are never satisfied.

Can we identify `live code'?

That is, that there are conditions for which the code will be executed?

For instance, are there conditions under which P_3 will be executed?

P_3 will be executed if there is a satisfying assignment to

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

this is equivalent to

$$(p \vee r) \wedge (r \wedge q) \quad \wedge (\neg r \vee q)$$

This last formula is satisfied under the valuation:

$$p^t = 1, \quad r^t = 1, \quad q^t = 1$$

Recall that $A \to B$ and $\neg A \vee B$ are equivalent.

That is $A \to B \models \neg A \vee B$.

In this case $\to$ is definable in terms of $\vee$ and $\neg$.

We have defined one unary connective and four binary connectives.

There are 4 unary connectives.

There are 2 unary inputs each of which can be mapped to 2 values.

For binary connectives there are 2^2 possible input value each of which can be mapped to two different values.

So there are $2^{(2^2)}$ different binary operators.

In general there are $2^{(2^n)}$ different n-ary connectives.

Here are the four unary connectives

A	$f_1 A$	$f_2 A$	$f_3 A$	$f_4 A$
1	1	1	0	0
0	1	0	1	0

Consider the 16 different binary connectives $g_1, \ldots, g_{16}$.

For example:

A	B	g_1AB	g_2AB	g_3AB	g_8AB	g_4AB	g_5A
1	1	1	1	1	1	1	0
1	0	1	1	1	0	0	0
0	1	1	1	0	0	1	0
0	0	1	0	1	1	0	1

A	B	g_5 A B	g_{12} A B	g_{13} A B
1	1	0	1	0
1	0	1	0	1
0	1	1	0	0
0	0	1	0	0

A set of connectives is adequate if all n-ary, $n \geq 1$, connectives can be defined in terms of the set.

Fact: $\{\neg, \wedge, \vee\}$ is an adequate set of connectives.

Fact: $\{\neg, \wedge\}$, $\{\neg, \vee\}$ and $\{\neg, \rightarrow\}$ are adequate sets.