Warm Up Problem

Let t be a truth valuation, p and q propositional variables with $t(p) = \mathtt{T}$, and let α be a well-formed formula. Which of the following are written using the correct notation? What does the equality-type character mean on each line?

1. $\alpha = \mathtt{T}$
2. $\alpha^t = \mathtt{T}$
3. $\alpha^t \equiv \mathtt{T}$
4. $(p \wedge q)^t = (\mathtt{T} \wedge q)$
5. $(p \wedge q)^t \equiv (\mathtt{T} \wedge q^t)$
6. $(p \wedge q) = (q \wedge p)$
7. $(p \wedge q) \equiv (q \wedge p)$

Propositional Logic: Semantics

Carmen Bruni
With thanks to A. Gao for these slides!

Based on slides by Jonathan Buss, Lila Kari, Anna Lubiw and Steve Wolfman with thanks to B. Bonakdarpour, A. Gao, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek

Lecture 5

Semantic entailment

- Define semantic entailment.
- Explain subtleties of semantic entailment.
- Determine whether a semantic entailment holds by using truth tables, valuation trees, and/or logical identities.
- Prove semantic entailment using truth tables and/or valuation trees.

Adequate sets of Connectives and Conjunctive Normal Form

- Define an adequate set of connectives and CNF
- Prove that a set is adequate
- Simplify formulas to CNF form

A piece of pseudo code

```
if ( (input > 0) or (not output) ) {
    if ( not (output and (queuelength < 100) ) ) {
        P_1
    } else if ( output and (not (queuelength < 100)) ) {
        P_2
    } else { P_3 }
} else { P_4 }
```

When does each piece of code get executed?

Let i: `input > 0`,
u: `output`,
q: `queuelength < 100`.

A Code Example, cont'd

i	u	q	$(i \vee (\neg u))$	$(\neg(u \wedge q))$	$(u \wedge (\neg q))$	
T	T	T	T			
T	T	F	T			
T	F	T	T			
T	F	F	T			
F	T	T	F			P_4
F	T	F	F			P_4
F	F	T	T			
F	F	F	T			

A Code Example, cont'd

i	u	q	$(i \vee (\neg u))$	$(\neg(u \wedge q))$	$(u \wedge (\neg q))$	
T	T	T	T	F	F	P_3
T	T	F	T	T		P_1
T	F	T	T	T		P_1
T	F	F	T	T		P_1
F	T	T	F			P_4
F	T	F	F			P_4
F	F	T	T	T		P_1
F	F	F	T	T		P_1

Finding Live Code

Prove that P_3 is live code. That is, the conditions leading to P_3 is satisfiable.

Theorem:
$\Big((i \vee (\neg u)) \wedge \Big(\big(\neg(\neg(u \wedge q))\big) \wedge \big(\neg(u \wedge (\neg q))\big)\Big)\Big) \equiv ((i \wedge u) \wedge q)$

Proof: In class

Two pieces of code: Are they equivalent?

Prove that the two code fragments are equivalent.

Listing 1: Your code

```
if (i || !u) {
  if (!(u && q)) {
    P1
  } else if (u && !q) {
    P2
  } else { P3 }
} else { P4 }
```

Listing 2: Your friend's code

```
if ((i && u) && q) {
  P3
} else if (!i && u) {
  P4
} else {
  P1
}
```

Simplifying Code

To prove that the two fragments are equivalent, show that each block of code P_1, P_2, P_3, and P_4 is executed under equivalent conditions.

Block	Fragment 1	Fragment 2
P_1	$(i \vee (\neg u)) \wedge (\neg(u \wedge q))$	$(\neg(i \wedge u \wedge q)) \wedge (\neg((\neg i) \wedge u))$
P_2	$(i \vee (\neg u)) \wedge (\neg(\neg(u \wedge q))) \wedge (u \wedge (\neg q))$	F
P_3	$(i \vee (\neg u)) \wedge (\neg(\neg(u \wedge q))) \wedge (\neg(u \wedge (\neg q)))$	$(i \wedge u \wedge q)$
P_4	$(\neg(i \vee (\neg u)))$	$(\neg(i \wedge u \wedge q)) \wedge ((\neg i) \wedge u)$

Satisfiability of Sets of Formulas

The notion of satisfiability extends to sets of formulas.

Let Σ denote a set of well-formed formulas and t a valuation. Define

$$\Sigma^t = \begin{cases} \mathtt{T} & \text{if for each } \beta \in \Sigma,\ \beta^t = \mathtt{T} \\ \mathtt{F} & \text{otherwise} \end{cases}$$

that is, $\Sigma^t = \mathtt{T}$ if and only if for all $\beta \in \Sigma$, $\beta^t = \mathtt{T}$.

When $\Sigma^t = \mathtt{T}$, we say that t *satisfies* Σ.

A set Σ is *satisfiable* iff there is some valuation t such that $\Sigma^t = \mathtt{T}$.

Example. The set $\{((p \to q) \vee r),\ (p \vee (q \vee s))\}$ is satisfiable.

Exercise

Let V be the set of all truth valuations. Write down using only symbols what it means for a set of formulas Σ to be satisfiable. Do the same for what it means for $\Sigma^t = \mathtt{T}$ for some arbitrary $t \in V$.

Exercise

Let V be the set of all truth valuations. Write down using only symbols what it means for a set of formulas Σ to be satisfiable. Do the same for what it means for $\Sigma^t = \mathtt{T}$ for some arbitrary $t \in V$.

1. $\exists t \in V,\ \Sigma^t = \mathtt{T}$
2. $\Sigma^t = \mathtt{T} \leftrightarrow (\forall \beta \in \Sigma,\ \beta^t = \mathtt{T})$

Proving arguments valid

Recall that logic is the science of reasoning.

One important goal of logic is to infer that a conclusion is true based on a set of premises.

A logical argument:

Premise 1

Premise 2

...

Premise n

Conclusion

A common problem is to prove that an argument is valid, that is the set of premises semantically entails the conclusion.

Logical Consequence, a.k.a. Entailment

Let Σ be a set of formulas, and let α be a formula. We say that

- α is a *logical consequence* of Σ, or
- Σ *(semantically) entails* α, or
- $\Sigma \models \alpha$,

if and only if

$$\text{for all truth valuations } t, \text{ if } \Sigma^t = \mathtt{T} \text{ then also } \alpha^t = \mathtt{T}.$$

We write $\Sigma \not\models \alpha$ for "not $\Sigma \models \alpha$". That is,

$$\text{there exists a truth valuation } t \text{ such that } \Sigma^t = \mathtt{T} \text{ and } \alpha^t = \mathtt{F}.$$

For the previous slide, $\Sigma = \{p_1, p_2, ..., p_n\}$ could be a set of premises and let α could be the conclusion that we want to derive.

Exercise

Write down using only symbols what $\Sigma \models \alpha$ means where Σ is a set of well-formed formulas and α is well-formed.

Exercise

Write down using only symbols what $\Sigma \vDash \alpha$ means where Σ is a set of well-formed formulas and α is well-formed.

$$\Sigma \vDash \alpha \leftrightarrow (\forall t \in V, (\Sigma^t \rightarrow \alpha^t = T))$$

Proving or disproving entailment

Proving that Σ entails α, denoted $\Sigma \models \alpha$:

- Using a truth table: Consider all rows of the truth table in which all of the formulas in Σ are true. Verify that α is true in all of these rows.
- Direct proof: For every truth valuation under which all of the premises are true, show that the conclusion is also true under this valuation.
- Proof by contradiction: Assume that the entailment does not hold, which means that there is a truth valuation under which all of the premises are true and the conclusion is false. Derive a contradiction.

Proving that Σ does not entail α, denoted $\Sigma \not\models \alpha$:

- Find one truth valuation t under which all of the premises in Σ are true and the conclusion α is false.

Equivalence and Entailment

Equivalence can be expressed using the notion of entailment.

Lemma. $\alpha \equiv \beta$ if and only if both $\{\alpha\} \models \beta$ and $\{\beta\} \models \alpha$.

Use of Connectives:

Adequate sets and normal forms

Definability of Connectives

Formulas $(\alpha \to \beta)$ and $((\neg\alpha) \vee \beta)$ are equivalent.

Thus $\to$ is said to be *definable* in terms of $\neg$ and $\vee$.

We never need to use $\to$; we can always write an equivalent formula without it.

There are actually sixteen possible binary connectives. (Why?)

Of these, two are essentially nullary (they ignore the values they connect).

Four others are essentially unary (they ignore one value but not the other).

Adequate Sets

A set of connectives is said to be *adequate* iff any n-ary ($n \geq 1$) connective is definable in terms of the ones in the set.

or equivalently

A set of connectives is said to be *adequate* iff every well-formed formula is logically equivalent to a well-formed formula using only connectives from the set.

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

Lemma. Each of the sets $\{\wedge, \neg\}$, $\{\vee, \neg\}$, and $\{\rightarrow, \neg\}$ is adequate.

Theorem. The set $\{\wedge, \vee\}$ is *not* an adequate set of connectives.

Adequate Sets

A set of connectives is said to be *adequate* iff every well-formed formula is logically equivalent to a well-formed formula using only connectives from the set.

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

Proof:

Adequate Sets

A set of connectives is said to be *adequate* iff every well-formed formula is logically equivalent to a well-formed formula using only connectives from the set.

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

Proof: By structural induction. Let $P(\gamma)$ be the statement that γ is logically equivalent to a well-formed formula that uses only connectives from the set $S = \{\wedge, \vee, \neg\}$. We prove this by structural induction.

Adequate Sets

A set of connectives is said to be *adequate* iff every well-formed formula is logically equivalent to a well-formed formula using only connectives from the set.

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

Proof: By structural induction. Let $P(\gamma)$ be the statement that γ is logically equivalent to a well-formed formula that uses only connectives from the set $S = \{\wedge, \vee, \neg\}$. We prove this by structural induction.

BC: If γ is a propositional formula, the statement holds since there are no connectives.

IH: Assume $P(\alpha)$ and $P(\beta)$ holds for some α and β.

Adequate Sets

A set of connectives is said to be *adequate* iff every well-formed formula is logically equivalent to a well-formed formula using only connectives from the set.

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

Proof (Continued):

IC: If $\gamma = (\neg\alpha)$, then since α is logically equivalent to a formula using only connectives from S, we have that γ must also be since it is the negation of this formula.

If $\gamma = (\alpha \star \beta)$, then if $\star \in S$, we are done by the logic above. If $\star = \rightarrow$, then $\gamma \equiv ((\neg\alpha) \vee \beta)$ by the "Implication" rule and we're done. Lastly, if $\star = \leftrightarrow$, then we have that $\gamma \equiv ((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha))$ and using the implication rule, we would also be done.

Adequate Sets

A set of connectives is said to be *adequate* iff every well-formed formula is logically equivalent to a well-formed formula using only connectives from the set.

Lemma. Each of the sets $\{\wedge, \neg\}$, $\{\vee, \neg\}$, and $\{\rightarrow, \neg\}$ is adequate.

Proof Sketch: The first two follow from DeMorgan's Laws. The third can follow from the first two. They can all be proven by structural induction or reduced to another known adequate set of connectives.

Proof of last Theorem

Theorem. The set $\{\wedge, \vee\}$ is *not* an adequate set of connectives.

This follows from two results:

1. Any well-formed formula composed using only $\wedge$ and $\vee$ when evaluated under a valuation making all propositional variables true gives you true. (Use Structural Induction!)
2. The formula $(\neg p)$ cannot be expressed in this set. (Hint: Otherwise, such a well-formed formula should only contain this atom p and the connectives in $\{\wedge, \vee\}$ which will give you a contradiction by the previous part).

Some Questions about Adequate Sets

Question: Is there a connective $*$ such that the singleton set $\{*\}$ is adequate?

Question: Are there connectives c_1, c_2 and c_3 such that $\{c_1, c_2, c_3\}$ is adequate, but none of $\{c_1, c_2\}$, $\{c_1, c_3\}$, or $\{c_2, c_3\}$ is adequate? (Such a set is called a *minimal* adequate set.)

Question: Find all minimal adequate sets containing only binary, unary and nullary connectives.

Conjunctive Normal Form

Conjunctive normal form (CNF):

- A *literal* is a (propositional) variable or the negation of a variable.
- A *clause* is a disjunction of literals. (literals connected by $\vee$)
- A formula is in *conjunctive normal form* if it is a conjunction of clauses.

In other words, a formula is in CNF if and only if

- its only connectives are $\neg$, $\vee$ and/or $\wedge$,
- $\neg$ applies only to variables, and
- $\vee$ applies only to subformulas with no occurrence of $\wedge$.

For CNF Formulas, it is usually easier to read by dropping brackets and only using them around clauses and the whole formula (possible by associativity since we're only dealing with $\wedge$ and $\vee$ binary connectives and only care up to logical equivalence):

CNF examples

Some CNF formulas:

- $\big((p \vee (\neg q)) \wedge \big(r \wedge ((\neg r) \vee (p \vee q))\big)\big)$
- $\big((\neg r) \vee (p \vee q)\big)$ (Only one clause.)
- $\big((\neg r) \wedge (p \wedge q)\big)$ (Three singleton clauses.)

With simpler bracketing:

- $\big((p \vee (\neg q)) \wedge r \wedge ((\neg r) \vee p \vee q)\big)$.
- $\big((\neg r) \vee p \vee q\big)$. (Only one clause.)
- $\big((\neg r) \wedge p \wedge q\big)$. (Three singleton clauses.)

CNF examples

Some formulas not in CNF:

- $\big(\neg\big((\neg p) \wedge q\big)\big)$. ($\neg$ applied to $\wedge$.)
- $\big(p \vee (r \wedge q)\big)$. ($\vee$ applied to $\wedge$.)
- $\big(\neg(\neg q)\big)$. ($\neg$ applied to $\neg$.)
- $(p \to q)$. (Uses $\to$.)

Converting to CNF

1. Eliminate implication and equivalence.
   Replace $(\alpha \to \beta)$ by $((\neg\alpha) \vee \beta)$
   Replace $(\alpha \leftrightarrow \beta)$ by $((\neg\alpha) \vee \beta) \wedge (\alpha \vee (\neg\beta))$.
   (*Now only* $\wedge$, $\vee$ *and* $\neg$ *appear as connectives.*)
2. Apply De Morgan's and double-negation laws as often as possible.
   Replace $(\neg(\alpha \vee \beta))$ by $((\neg\alpha) \wedge (\neg\beta))$.
   Replace $(\neg(\alpha \wedge \beta))$ by $((\neg\alpha) \vee (\neg\beta))$.
   Replace $(\neg(\neg\alpha))$ by α.
   (*Now negation only occurs in literals.*)
3. Transform into a conjunction of clauses using distributivity.
   Replace $(\alpha \vee (\beta \wedge \gamma))$ by $((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$.
   (*One could stop here, but....*)
4. Simplify using idempotence, contradiction, excluded middle and Simplification I & II.

Working with CNF Formulas

In a clause, there is no need to have any variable twice. [Why?]
Also, the order of the literals in the clause does not matter.

Thus we can think of a clause as simply a set of literals.

Similarly, in a CNF formula, no clause [set] need appear more than once, and the order of clauses does not matter.

Thus we can think of a CNF formula as simply a set of clauses.

For example, the formula $\Big((p \vee q) \wedge \big((q \vee (\neg r)) \wedge s\big)\Big)$
can be described by the set of clauses $\{p, q\}$, $\{q, (\neg r)\}$ and $\{s\}$.

We shall use these observations in our next topic, "Resolution".