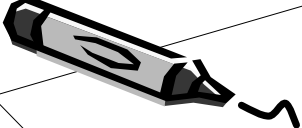


Propositional Logic

CIS*3700 (Winter 2007)




Syntax



- Syntax specifies what sentences are well-formed

$\langle \text{Sentence} \rangle \rightarrow \langle \text{AtomicSentence} \rangle \mid \langle \text{ComplexSentence} \rangle$

$\langle \text{AtomicSentence} \rangle \rightarrow \text{TRUE} \mid \text{FALSE} \mid \langle \text{Symbol} \rangle$



$\langle \text{Symbol} \rangle \rightarrow P \mid Q \mid R \mid \dots$

$\langle \text{ComplexSentence} \rangle \rightarrow \neg \langle \text{Sentence} \rangle$
 $\mid (\langle \text{Sentence} \rangle \wedge \langle \text{Sentence} \rangle)$
 $\mid (\langle \text{Sentence} \rangle \vee \langle \text{Sentence} \rangle)$
 $\mid (\langle \text{Sentence} \rangle \Rightarrow \langle \text{Sentence} \rangle)$
 $\mid (\langle \text{Sentence} \rangle \Leftrightarrow \langle \text{Sentence} \rangle)$

Simplified Syntax



- Order of precedence: from highest to lowest
 - \neg (negation)
 - \wedge (conjunction)
 - \vee (disjunction)
 - \Rightarrow (implication)
 - \Leftrightarrow (biconditional)
- Example:
 - $\neg P \vee Q \wedge R \Rightarrow S$ is equivalent to $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

Semantics



- Semantics defines the truth value of a sentence with regard to a model.
- Truth table for the five logical operations:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>



Logical Entailment

- A model is a "possible" world in which a sentence is true.
- A sentence α entails a sentence β iff in every model where α is true, β is also true. This is denoted as $\alpha \models \beta$. We also say that β is a logical consequence of α .
- When $\alpha \models \beta$, is it also true that $\beta \models \alpha$?

Inference

- KB can be viewed as a set of conjunctive sentences
- Inference derives a new sentence from the existing sentences in KB, denoted as $\text{KB} \vdash \alpha$.
- Entailment vs. Inference: entailment is like a needle in the haystack; inference is like finding the needle in the haystack.

Evaluation of Inference Algorithm

- Given an inference algorithm:
 - For some sentence α , it holds that $KB \vdash \alpha$, and
 - For some sentence β , it holds that $KB \not\vdash \beta$
- An inference algorithm is sound if it derives only entailed sentences
- An inference algorithm is complete if it can derive all entailed sentences.

Inference by Model Checking

```
function TT-ENTAILS?(KB,  $\alpha$ ) returns true or false
  symbols  $\leftarrow$  a list of the proposition symbols in KB and  $\alpha$ 
  return TT-CHECK-ALL(KB,  $\alpha$ , symbols, [])

function TT-CHECK-ALL(KB,  $\alpha$ , symbols, model) returns true or false
  if EMPTY?(symbols) then
    if PL-TRUE?(KB, model) then return PL-TRUE?( $\alpha$ , model)
    else return true
  else do
    P  $\leftarrow$  FIRST(symbols); rest  $\leftarrow$  REST(symbols)
    return TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, true, model)) and
      TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, false, model))
```

Inference by Model Checking

- Idea: enumerate all models and check whether KB entails α .
- Sound?
- Complete?
- Complexity?
- Need algorithms that are most efficient on average cases.

Logical Equivalence

- Sentences α and β are logically equivalent if they are true in the same set of models, denoted as $\alpha = \beta$.
- Does $\neg(\alpha \wedge \beta) = (\neg\alpha \vee \neg\beta)$ hold?
- Theorem: $\alpha = \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$.
- Commonly used logical equivalences: how can they be used?

Common Logical Equivalence

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of \wedge
 $(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of \vee
 $((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of \wedge
 $((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of \vee
 $\neg(\neg\alpha) \equiv \alpha$ double-negation elimination
 $(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition
 $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination
 $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination
 $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan
 $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan
 $(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of \wedge over \vee
 $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of \vee over \wedge

Validity

- A sentence is valid if it is true in all models
 - A valid sentence is called a tautology
 - E.g., $P \vee \neg P$ is valid
 - All valid sentences are equivalent to TRUE
- Deduction theorem: For any sentences α and β , $\alpha \models \beta$ iff the sentence $\alpha \Rightarrow \beta$ is valid.
- How can we test/verify the validity of a sentence?

Proof of Deduction Theorem

- Suppose $\alpha \models \beta$, we show $\alpha \Rightarrow \beta$ is valid.
 - Let m be any model. If α is true in m , then by definition, β is true in m . Hence, $\alpha \Rightarrow \beta$ is also true in m .
 - If α is false in m , then $\alpha \Rightarrow \beta$ is true in m .
 - Thus, we have shown that $\alpha \Rightarrow \beta$ is true in any model. That is, $\alpha \Rightarrow \beta$ is valid.
- Suppose $\alpha \Rightarrow \beta$ is valid:
 - For any model m , if α is true in m , β must be true in m . Therefore, $\alpha \models \beta$.



Satisfiability

- A sentence is satisfiable/consistent if it is true in at least one model:
 - m is a model of α , or m satisfies α .
- A sentence is unsatisfiable/inconsistent if there exists no model that satisfies it.
- Contradiction theorem: $\alpha \models \beta$ iff the sentence $(\alpha \wedge \neg\beta)$ is unsatisfiable. Also called proof by contradiction or refutation.



Proof of Contradiction Theorem

- To show that $(\alpha \wedge \neg\beta)$ is unsatisfiable, we do the following transformation:

$$\begin{aligned}\neg(\alpha \Rightarrow \beta) &= \neg(\neg\alpha \vee \beta) \\ &= (\neg(\neg\alpha) \wedge \neg\beta) \\ &= (\alpha \wedge \neg\beta)\end{aligned}$$



Inference Rules

- Inference rules help derive a new sentence:
 - Modus ponens:
$$\frac{a \Rightarrow b, a}{b}$$
 - And elimination:
$$\frac{a \wedge b}{a}$$
- Are they sound?
- What do we gain by having these rules?



Proof

- Proof is a sequence of applications of inference rules that derives a sentence a from a KB.
- Proof is another way of implementing logical inferences:
Inference algorithm = inference rules + search
- Will finding a proof be more efficient than model checking in many cases?



Resolution

- A powerful inference rule that yields a sound and complete inference algorithm when coupled with a complete search algorithm:

$$\frac{a \vee b, \neg b \vee g}{a \vee g}$$

where $(\alpha \vee \gamma)$ is called a resolvent.



Special Cases of Resolution

- Modus Ponens: from $\{\alpha, \neg\alpha \vee \beta\}$, the resolvent is β .
- Contradiction: from $\{\alpha, \neg\alpha\}$, the resolvent is $\{\}$.
- Chaining: from $\{\neg\alpha \vee \beta, \neg\beta \vee \gamma\}$, the resolvent is $(\neg\alpha \vee \gamma)$.



Conjunctive Normal Form

- CNF: a conjunction of disjunctions of literals.
- Example:

$$\begin{aligned}
 & (\neg P \vee (Q \wedge \neg R)) \vee S \\
 = & ((\neg P \vee Q) \wedge (\neg P \vee \neg R)) \vee S \\
 = & S \vee ((\neg P \vee Q) \wedge (\neg P \vee \neg R)) \\
 = & (S \vee (\neg P \vee Q)) \wedge (S \vee (\neg P \vee \neg R)) \\
 = & (S \vee \neg P \vee Q) \wedge (S \vee \neg P \vee \neg R)
 \end{aligned}$$



Breaking into Clauses

- A conjunctive normal form allows us to break a sentence into a set of clauses, which are disjunctions of literals.
- Given $(S \vee \neg P \vee Q) \wedge (S \vee \neg P \vee \neg R)$, we can get two clauses:

$$(S \vee \neg P \vee Q) \text{ and } (S \vee \neg P \vee \neg R)$$



Example

- Consider the following set of clauses:

- (1) $P \vee Q$
- (2) $\neg P \vee Q$
- (3) $P \vee \neg Q$
- (4) $\neg P \vee \neg Q$

- Proof:

- | | |
|--------------|------------------|
| (5) Q | from (1) and (2) |
| (6) $\neg Q$ | from (3) and (4) |
| (7) $\{\}$ | from (5) and (6) |

Thus, the given set of clauses is unsatisfiable.



Resolution Algorithm

- Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

```

function PL-RESOLUTION( $KB, \alpha$ ) returns true or false
  clauses  $\leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg\alpha$ 
  new  $\leftarrow \{\}$ 
  loop do
    for each  $C_i, C_j$  in clauses do
      resolvents  $\leftarrow$  PL-RESOLVE( $C_i, C_j$ )
      if resolvents contains the empty clause then return true
      new  $\leftarrow$  new  $\cup$  resolvents
    if new  $\subseteq$  clauses then return false
  clauses  $\leftarrow$  clauses  $\cup$  new
  
```



Deleting Strategy

- Unlimited applications of resolution may generate irrelevant and redundant clauses:

- S0:
- (1) $P \vee Q$
 - (2) $\neg P \vee Q$
 - (3) $P \vee \neg Q$
 - (4) $\neg P \vee \neg Q$

- S1:
- | | |
|----------------------|------------------|
| (5) Q | from (1) and (2) |
| (6) P | from (1) and (3) |
| (7) $Q \vee \neg Q$ | from (1) and (4) |
| (8) $P \vee \neg P$ | from (1) and (4) |
| (9) $Q \vee \neg Q$ | from (2) and (3) |
| (10) $P \vee \neg P$ | from (2) and (3) |
| (11) $\neg P$ | from (2) and (4) |
| (12) $\neg Q$ | from (3) and (4) |



Deleting Strategy

S2: (13) Q	from (1) and (2)
(14) P	from (1) and (3)
(15) $Q \vee \neg Q$	from (1) and (4)
(16) $P \vee \neg P$	from (1) and (4)
(17) $Q \vee \neg Q$	from (2) and (3)
(18) $P \vee \neg P$	from (2) and (3)
(19) $\neg P$	from (2) and (4)
(20) $\neg Q$	from (3) and (4)
...	
(35) $\neg P \vee \neg Q$	from (4) and (9)
(36) $\neg P \vee \neg Q$	from (4) and (10)
(37) Q	from (5) and (7)
(38) Q	from (5) and (9)
(39) {}	from (5) and (12)



Deleting Strategy

- Deleting strategy: remove tautologies and redundant clauses whenever possible.

S0: (1) $P \vee Q$	
(2) $\neg P \vee Q$	
(3) $P \vee \neg Q$	
(4) $\neg P \vee \neg Q$	
S1: (5) Q	from (1) and (2)
(6) P	from (1) and (3)
(7) $\neg P$	from (2) and (4)
(8) $\neg Q$	from (3) and (4)
S2: (9) {}	from (5) and (8)



Properties of Resolution Algorithm

- It's easy to show that resolution algorithm is sound, but is it also complete?
- Resolution closure: the set of all clauses derived from the applications of resolution. Is it finite?
- Ground resolution theorem: if a set of clauses is unsatisfiable, then their closure contains an empty clause.



Wumpus KB

- Agent's KB for the Wumpus world on slide 13 of the previous topic:

$\neg S_{1,1}$	$\neg B_{1,1}$
$\neg S_{2,1}$	$B_{2,1}$
$S_{1,2}$	$\neg B_{1,2}$
$R_1: \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$	
$R_2: \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$	
$R_3: \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$	
$R_4: S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$	



Finding Wumpus

- Building a truth table for $KB \models W_{1,3}$ needs 4096 entries.
- Using inference rules:
 - Modus ponens for $\neg S_{1,1}$ and R_1 : $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
 - And-elimination to (1) above: $\neg W_{1,1}, \neg W_{1,2}, \neg W_{2,1}$
 - Modus ponens for $\neg S_{2,1}$ and R_2 and then and-elimination: $\neg W_{2,2}, \neg W_{3,1}$
 - Modus ponens for $S_{1,2}$ and R_4 : $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
 - Resolution to (4) and (2a): $W_{1,3} \vee W_{1,2} \vee W_{2,2}$
 - Resolution to (5) and (3a): $W_{1,3} \vee W_{1,2}$
 - Resolution to (6) and (2b): $W_{1,3}$



Problems with Propositional Logic

- Unable to answer "what action should I take?", but able to answer questions such as "should I go forward?", "should I turn right?", etc.
- A large number of propositions are needed to describe the actions. For example, $16 \times 4 = 64$ propositions for the moves in Wumpus world.
- The world changes over time: 6400 propositions required to specify 100 time steps in Wumpus world.

