

Logic

Propositional Logic	2
Syntactic Elements	2
Syntax	3
Semantics	4
Deduction	5
Entailment	6
The Wumpus World Example	7
Inference Rules	8
Inference Procedures	9
Resolution Inference Procedure	10
Useful Equivalences	11
First-Order Logic	12
Syntactic Elements	12
Syntax	13
Semantics	14
Inference Rules	15
Example: Setup	16
Example, Proof, Part 1	17
Example, Proof, Part 2	18
Unification	19
Unify-Lists Procedure	20
Unify-Terms Procedure	21
Resolve Procedure Preliminaries	22
Resolve Procedure	23

Propositional Logic

Syntactic Elements

- An *atomic sentence* consists of a single *propositional symbol*, representing a proposition that can be true or false.
- A *literal* is a propositional symbol or its negation.
- *Complex sentences* are constructed from simpler sentences using *logical connectives*:
 \neg (not), \wedge (and), \vee (or), \rightarrow (implies) [I prefer \rightarrow to \Rightarrow], and \leftrightarrow (iff).

Syntax

A *proposition* or *propositional sentence* can be formed as follows:

- Every propositional symbol is a sentence.
- If A is a sentence, then $\neg A$ is a sentence.
- If A_1 and A_2 are sentences, then so are:
 - $A_1 \wedge A_2$, (and, conjunction)
 - $A_1 \vee A_2$, (or, disjunction)
 - $A_1 \rightarrow A_2$, and (implies, implication)
 - $A_1 \leftrightarrow A_2$. (iff, biconditional)

Semantics

- A *world* or *domain* is the situation (set of facts) we want to represent.
- Different *possible worlds* correspond to different situations.
- An *interpretation* maps each propositional symbol to the world.
- A sentence is *true* if its interpretation in the world is true.
- A *knowledge base* is a set of sentences.
- A *model* of a world assigns true and false to each sentence.

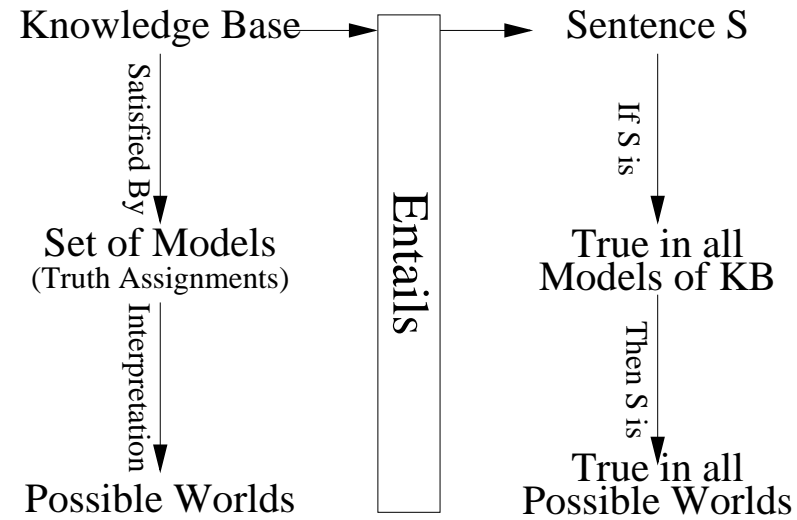
Deduction

- A sentence S is *satisfiable* within a KB if S is true in some model of the KB, i.e., some truth assignment makes S and the KB true.
- A sentence S is *entailed* by a KB if S is true in all models of the KB (denoted as $KB \models S$), i.e., every truth assignment that makes KB true also makes S true.
- *Logical inference* or *deduction* is concerned with producing entailed sentences from KBs.

CS 3793 Artificial Intelligence

Logic - 5

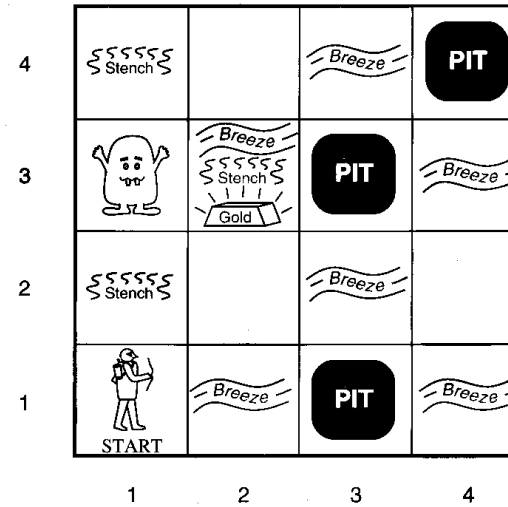
Entailment



CS 3793 Artificial Intelligence

Logic - 6

The Wumpus World Example



CS 3793 Artificial Intelligence

Logic - 7

Inference Rules

- **Modus Ponens** From $A \rightarrow B$ and A Infer B
- **Unit Resolution** From $A \vee B$ and $\neg B$ Infer A
- **Resolution** From $A \vee B$ and $\neg B \vee C$ Infer $A \vee C$

CS 3793 Artificial Intelligence

Logic - 8

Inference Procedures

- An *inference procedure* uses inference rules to produce proofs. Denoted as $KB \vdash S$.
- An inference procedure is *sound* if it can only prove entailed sentences, i.e., every sentence it proves is entailed.
- An inference procedure is *complete* if it can prove any entailed sentence, i.e., every entailed sentence can be proved.

CS 3793 Artificial Intelligence

Logic – 9

Resolution Inference Procedure

- Resolution applies to *conjunctive normal form*.
 - A KB is a conjunction of sentences.
 - Each sentence is a disjunction of literals.
- Resolution is *refutation complete*. If a KB is in CNF, and if the KB is inconsistent, then resolution will infer inconsistent literals.
- To deduce a sentence S , first temporarily add $\neg S$ to the KB. Then, if resolution infers inconsistent literals, then $\neg S$ is not satisfiable within the KB, which means that S is entailed.

CS 3793 Artificial Intelligence

Logic – 10

Useful Equivalences for Converting to CNF

Several logical equivalences are handy for converting sentences to CNF

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

CS 3793 Artificial Intelligence

Logic – 11

First-Order Logic

12

Syntactic Elements

- A symbol in first-order logic can be a predicate, a function, a constant term, or a variable term.
- First-order logic uses the connectives \neg (not), \wedge (and), \vee (or), \rightarrow (implies), and \leftrightarrow (iff).
- First-order logic also uses the *quantifiers* \forall (for all) and \exists (there exists).

CS 3793 Artificial Intelligence

Logic – 12

Syntax

- A *term* is a constant or variable, or a function applied to a sequence of terms.
- A *ground term* is a term with no variables.
- An *atomic sentence* or *atom* is a predicate applied to a sequence of terms.
- A *ground atom* is an atom with no variables.
- Connectives can be used to construct more complex sentences in the usual way.
- If A is a sentence and x is a variable, then:
 - $\forall x A$ is a sentence (universal quantifier).
 - $\exists x A$ is a sentence (existential quantifier).
- A *well-formed formula* is a sentence in which all the variables are quantified.

CS 3793 Artificial Intelligence

Logic – 13

Semantics

- The connectives \wedge , \vee , \neg , \rightarrow , and \leftrightarrow are evaluated in the usual way.
- $\forall x A$ is true if every substitution for x makes A true.
- $\exists x A$ is true if at least one substitution for x makes A true.
- An *interpretation* consists of objects in the world and mappings for terms and predicates.
- Each ground term is mapped to an object.
- Each predicate is mapped to a relation (think relational database).
- A ground atom is *true* if the predicate's relation holds between the terms' objects.

CS 3793 Artificial Intelligence

Logic – 14

Inference Rules

- Universal Elimination $\frac{\text{From } \forall x A(x)}{\text{Infer } A(t) \text{ where } t \text{ is any term}}$
- Existential Elimination $\frac{\text{From } \exists x A(x)}{\text{Infer } A(c) \text{ where } c \text{ is a new constant}}$
- Resolution (disjunctive) $\frac{\text{From } \forall x, y B(y) \vee A(x) \text{ and } \forall x, z \neg A(x) \vee C(z)}{\text{Infer } \forall y, z B(y) \vee C(z)}$
- Resolution (implicative) $\frac{\text{From } \forall x, y B(y) \rightarrow A(x) \text{ and } \forall x, z A(x) \rightarrow C(z)}{\text{Infer } \forall y, z B(y) \rightarrow C(z)}$

CS 3793 Artificial Intelligence

Logic – 15

Example: Setup

- Knowledge Base

```
-parent(x,y) | -ancestor(y,z) | ancestor(x,z)
-parent(x,y) | ancestor(x,y)
-mother(x,y) | parent(x,y)
-father(x,y) | parent(x,y)
mother(Liz,Charley)
father(Charley,Billy)
```

- To prove ancestor(Liz,Billy)
- Refute -ancestor(Liz,Billy)

CS 3793 Artificial Intelligence

Logic – 16

Example, Proof, Part 1

```
-parent(x,y) | -ancestor(y,z) | ancestor(x,z)
-ancestor(Liz,Billy)
-----
-parent(Liz,y) | -ancestor(y,Billy)

-mother(x,y) | parent(x,y)
-parent(Liz,y) | -ancestor(y,Billy)
-----
-mother(Liz,y) | -ancestor(y,Billy)

mother(Liz,Charley)
-mother(Liz,y) | -ancestor(y,Billy)
-----
-ancestor(Charley,Billy)
```

CS 3793 Artificial Intelligence

Logic – 17

Example, Proof, Part 2

```
-parent(x,y) | ancestor(x,y)
-ancestor(Charley,Billy)
-----
-parent(Charley,Billy)

-father(x,y) | parent(x,y)
-parent(Charley,Billy)
-----
-father(Charley,Billy)

father(Charley,Billy)
-father(Charley,Billy)
----- contradiction
```

CS 3793 Artificial Intelligence

Logic – 18

Unification

- To use the resolution inference rule, we need to be able to match atoms in sentences. This is called *unification*.
- If successful, unification returns a *substitution*.
- A substitution specifies values for variables that would make the two atoms identical.

CS 3793 Artificial Intelligence

Logic – 19

Unify-Lists Procedure

```
function UNIFY-LISTS( $A, B, \theta$ )
  if  $A$  and  $B$  have different predicates/functions
    or have different numbers of arguments
  then return failure
  for  $i \leftarrow 1$  to number of arguments do
     $termA \leftarrow$   $i$ th argument of  $A$ 
     $termB \leftarrow$   $i$ th argument of  $B$ 
     $\theta \leftarrow$  UNIFY-TERMS( $termA, termB, \theta$ )
    if  $\theta =$  failure then return failure
  end for
  return  $\theta$ 
```

CS 3793 Artificial Intelligence

Logic – 20

Unify-Terms Procedure

```
function UNIFY-TERMS( $A, B, \theta$ )
  while  $A$  is a variable and  $\theta[A]$  exists
    do  $A \leftarrow \theta[A]$ 
  while  $B$  is a variable and  $\theta[B]$  exists
    do  $B \leftarrow \theta[B]$ 
  if  $A = B$  then do nothing
  else if  $A$  is a variable then  $\theta[A] \leftarrow B$ 
  else if  $B$  is a variable then  $\theta[B] \leftarrow A$ 
  else if  $A$  or  $B$  is a constant then  $\theta \leftarrow$  failure
  else  $\theta \leftarrow$  UNIFY-LISTS( $A, B, \theta$ )
  return  $\theta$ 
```

CS 3793 Artificial Intelligence

Logic – 21

Resolve Procedure Preliminaries

- The RESOLVE procedure assumes that sentences A and B are disjunctions represented as sets of literals. A literal is an atom or its negation.
- RESOLVE assumes that sentences A and B have no variables with the same names.
- RESOLVE returns all the sentences it infers.

CS 3793 Artificial Intelligence

Logic – 22

Resolve Procedure

```
function RESOLVE( $A, B$ )
  for each  $litA$  in  $A$  do
    for each  $litB$  in  $B$  do
      if one of  $litA$  and  $litB$  is negated, not both then
         $\theta \leftarrow$  UNIFY-LISTS( $litA, litB, \text{empty substitution}$ )
        if  $\theta \neq$  failure and has no recursive substs. then
           $A' \leftarrow$  apply substitution  $\theta$  to  $A$ 
           $litA' \leftarrow$  apply substitution  $\theta$  to  $litA$ 
           $B' \leftarrow$  apply substitution  $\theta$  to  $B$ 
           $litB' \leftarrow$  apply substitution  $\theta$  to  $litB$ 
           $C \leftarrow (A' - \{litA'\}) \cup (B' - \{litB'\})$ 
          add  $C$  to formulas inferred
    return formulas inferred
```

CS 3793 Artificial Intelligence

Logic – 23