

For Friday

- Read Chapter 8
- Program 1, Milestone 2 due

Program 1

- Any questions?

Propositional Logic

- Simple logic
- Deals only in facts
- Provides a stepping stone into first order logic

Syntax

- Logical Constants: **true** and **false**
- Propositional symbols $P, Q \dots$ are sentences
- If S is a sentence then (S) is a sentence.
- If S is a sentence then $\neg S$ is a sentence.
- If S_1 and S_2 are sentences, then so are:
 - $S_1 \wedge S_2$
 - $S_1 \vee S_2$
 - $S_1 \Rightarrow S_2$
 - $S_1 \Leftrightarrow S_2$

Semantics

- true and false mean truth or falsehood in the world
- P is true if its proposition is true of the world
- $\neg S$ is the negation of S
- The remainder follow standard truth tables
 - $S_1 \wedge S_2$: AND
 - $S_1 \vee S_2$: inclusive OR
 - $S_1 \Rightarrow S_2$: True unless S_1 is true and S_2 is false
 - $S_1 \Leftrightarrow S_2$: bi-conditional, or if and only if

Vocabulary

- An **interpretation** is an assignment of true or false to each atomic proposition
- A sentence true under any interpretation is **valid** (a **tautology** or **analytic sentence**)
- Validity can be checked by exhaustive checking of truth tables

Rules of Inference

- Alternative to truth-table checking
- A sequence of inference rule applications leading to a desired conclusion is a **logical proof**
- We can check inference rules using truth tables, and then use to create sound proofs
- We can treat finding a proof as a search problem

Propositional Inference Rules

- Modus Ponens or Implication Elimination
- And-Elimination
- Unit Resolution
- Resolution

Simpler Inference

- Horn clauses
- Forward-chaining
- Backward-chaining

Building an Agent with Propositional Logic

- Propositional logic has some nice properties
 - Easily understood
 - Easily computed
- Can we build a viable wumpus world agent with propositional logic???

The Problem

- Propositional Logic only deals with facts.
- We cannot easily represent general rules that apply to any square.
- We cannot express information about squares and relate (we can't easily keep track of which squares we have visited)

More Precisely

- In propositional logic, each possible atomic fact requires a separate unique propositional symbol.
- If there are n people and m locations, representing the fact that some person moved from one location to another requires nm^2 separate symbols.

First Order Logic

- Predicate logic includes a richer ontology:
 - objects (terms)
 - properties (unary predicates on terms)
 - relations (n-ary predicates on terms)
 - functions (mappings from terms to other terms)
- Allows more flexible and compact representation of knowledge
- $\text{Move}(x, y, z)$ for person x moved from location y to z .

Syntax for First-Order Logic

Sentence \rightarrow AtomicSentence | Sentence Connective Sentence

| Quantifier Variable Sentence | \neg Sentence | (Sentence)

AtomicSentence \rightarrow Predicate(Term, Term, ...) | Term=Term

Term \rightarrow Function(Term, Term, ...) | Constant | Variable

Connective \rightarrow \vee | \wedge | \Rightarrow | \Leftrightarrow

Quantifier \rightarrow \exists | \forall

Constant \rightarrow A | John | Car1

Variable \rightarrow x | y | z | ...

Predicate \rightarrow Brother | Owns | ...

Function \rightarrow father-of | plus | ...

Terms

- Objects are represented by terms:
 - Constants: Block1, John
 - Function symbols: father-of, successor, plus
- An n-ary function maps a tuple of n terms to another term: father-of(John), succesor(0), plus(plus(1,1),2)
- Terms are simply names for objects.
- Logical functions are not procedural as in programming languages. They do not need to be defined, and do not really return a value.
- Functions allow for the representation of an infinite number of terms.

Predicates

- Propositions are represented by a predicate applied to a tuple of terms. A predicate represents a property of or relation between terms that can be true or false:
 - Brother(John, Fred), Left-of(Square1, Square2)
 - GreaterThan(plus(1,1), plus(0,1))
- In a given interpretation, an n-ary predicate can be defined as a function from tuples of n terms to {True, False} or equivalently, a set of tuples that satisfy the predicate:
 - {<John, Fred>, <John, Tom>, <Bill, Roger>, ...}

Sentences in First-Order Logic

- An atomic sentence is simply a predicate applied to a set of terms.
 - $\text{Owns}(\text{John}, \text{Car1})$
 - $\text{Sold}(\text{John}, \text{Car1}, \text{Fred})$
- Semantics is True or False depending on the interpretation, i.e. is the predicate true of these arguments.
- The standard propositional connectives ($\vee \leftarrow \wedge \Rightarrow \Leftrightarrow$) can be used to construct complex sentences:
 - $\text{Owns}(\text{John}, \text{Car1}) \vee \text{Owns}(\text{Fred}, \text{Car1})$
 - $\text{Sold}(\text{John}, \text{Car1}, \text{Fred}) \Rightarrow \neg \text{Owns}(\text{John}, \text{Car1})$
- Semantics same as in propositional logic.