First-order logic

• We saw how propositional logic can create intelligent behavior

• But propositional logic is a poor representation for complex environments
  – Why?
• First-order logic is a more expressive and powerful representation

What don’t we like about propositional logic?

• Lacks expressive power to describe the environment concisely
  – Separate rules for every square/square relationship in Wumpus world

Natural Language

• English appears to be expressive
  – Squares adjacent to pits are breezy

• But natural language is a medium of communication, not a knowledge representation
  – Much of the information and logic conveyed by language is dependent on context
  – Information exchange is not well defined
  – Not compositional (combining sentences may mean something different)
  – It is ambiguous
But we borrow representational ideas from natural language

• Natural language syntax
  – Nouns and noun phrases refer to objects
    • People, houses, cars
  – Verbs and verb phrases refer to relationships btw objects
    • Red, round, nearby, eaten
  – Some relationships are clearly defined functions where there is only one output for a given input
    • Best friend, first thing, plus
• We build first order logic around objects and relations

Ontology

– a "specification of a conceptualization"
– A description of the objects and relationships that can exist
  • Propositional logic had only true/false relationships
  • First-order logic has many more relationships
– The ontological commitment of languages is different
  • How much can you infer from what you know?
    – Temporal logic defines additional ontological commitments because of timing constraints

Higher-order logic

• First-order logic is “first” because you relate objects (the first-order entities that actually exist in the world)
  – There are 10 chickens… chickens.number=10
  – There are 10 ducks… ducks.number=10
• You cannot build relationships between relations or functions
  – There are as many chickens as ducks… chickens.number = ducks.number
  – the number of objects belonging to a group must be a property of the group, and not the objects themselves
  – Cannot represent Leibniz’s law: If x and y share all properties, x is y

Another characterization of a logic

• Epistemological commitments
  – The possible states of knowledge permitted with respect to each fact
    – In first-order logic, each sentence is a statement that is
      • True, false, or unknown

Formal structure of first-order logic

• Models of first-order logic contain:
  – A set of objects (its domain)
    • Alice, Alice’s left arm, Bob, Bob’s hat
  – Relationships between objects
    • Represented as tuples
      – Sibling (Alice, Bob), Sibling (Bob, Alice)
      – On head (Bob, hat)
    • Person (Bob), Person (Alice)
  – Some relationships are functions if a given object is related to exactly one object in a certain way
    • Alice -> Alice’s left arm

First-order logic syntax

• Constant Symbols
  – A, B, Bob, Alice, Hat
• Predicate Symbols
  – is, onHead, hasColor, person
• Function Symbols
  – Mother, leftLeg

• Each predicate and function symbol has an arity
  – A constant the fixes the number of arguments
First-order logic syntax

- Names of things are arbitrary
  - Knowledge base adds meaning
- Number of possible domain elements is unbounded
  - Number of models is unbounded
  - Checking enumeration by entailment is impossible

Syntax

- Term
  - A logical expression that refers to an object
    - Constants
    1. We could assign names to all objects, like providing a name for every shoe in your closet
    2. Function symbols
    - Used in place of a constant symbol OnLeftFoot(John)

Atomic Sentences

- Formed by a predicate symbol followed by parenthesized list of terms
  - Sibling (Alice, Bob)
  - Married (Father(Alice), Mother(Bob))

- An atomic sentence is true in a given model, under a given interpretation, if the relation referred to by the predicate symbol holds among the objects referred to by the arguments

Complex sentences

- We can use logical connectives
  - ~Sibling(LeftLeg(Alice), Bob)
  - Sibling(Alice, Bob) \(^\land\) Sibling (Bob, Alice)

Quantifiers

- A way to express properties of entire collections of objects
  - Universal quantification (\(\forall\),)
    - The power of first-order logic
    - Forall, King(x) => Person(x)

Universal Quantification

- Forall x, P
  - P is true for every object x
  - Forall x, King(x) => Person(x)
    - Richard the Lionheart
    - King John
    - Richard's left leg
    - John's left leg
    - The crown
Universal Quantification

Richard the Lionheart is a king ⇒ Richard the Lionheart is a person.
King John is a king ⇒ King John is a person.
Richard’s left leg is a king ⇒ Richard’s left leg is a person.
John’s left leg is a king ⇒ John’s left leg is a person.
The crown is a king ⇒ the crown is a person.

• Note that all of these are true
  – Implication is true if premise is false
  – Using AND instead of implication is overly strong
• By asserting a universally quantified sentence, you assert a whole list of individual implications

Existential Quantification

• There exist
  – There exists an x such that Crown(x) ∧ OnHead(x, John)
  – It is true for at least one object
Richard the Lionheart is a crown ∧ Richard the Lionheart is on John’s head;
King John is a crown ∧ King John is on John’s head;
Richard’s left leg is a crown ∧ Richard’s left leg is on John’s head;
John’s left leg is a crown ∧ John’s left leg is on John’s head;
The crown is a crown ∧ the crown is on John’s head.
– AND, ∧, is the appropriate connective

Existential Quantification

• What if we used implication as the connective?
  \[ \exists x \, \text{Crown}(x) \Rightarrow \text{OnHead}(x, \text{John}) \]
Richard the Lionheart is a crown ⇒ Richard the Lionheart is on John’s head;
King John is a crown ⇒ King John is on John’s head;
Richard’s left leg is a crown ⇒ Richard’s left leg is on John’s head.
– Implication is true if both premise and conclusion are true or if premise is false
  • Richard the Lionheart is not a crown, first assertion is true, and existential is satisfied

Nested Quantifiers

• Building more complex sentences
  \[ \forall x \, \forall y \, \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y) \]
  \[ \forall x, y \, \text{Sibling}(x, y) \Rightarrow \text{Sibling}(y, x) \]
  • Every person is related to every other person

• Use \( \exists y \forall x \, \text{ Loves}(x, y) \) names and parentheses when appropriate

Combining

– Everyone who dislikes parsnips ==
  there does not exist someone who likes parsnips
\[ \forall x \, \neg \text{Likes}(x, \text{Parsnips}) \text{ is equivalent to } \neg \exists x \, \text{Likes}(x, \text{Parsnips}) \]
– Everyone likes ice cream ==
  there is no one who does not like ice cream
\[ \forall x \, \text{Likes}(x, \text{IceCream}) \text{ is equivalent to } \neg \exists x \, \neg \text{Likes}(x, \text{IceCream}) \]

Combining

• De Morgan’s rules apply
\[ \forall x \, \neg P \Rightarrow \neg \exists x \, P \]
\[ \neg \forall x \, P \Rightarrow \exists x \, \neg P \]
\[ \forall x \, P \Rightarrow \neg \exists x \, \neg P \]
\[ \exists x \, P \Rightarrow \neg \forall x \, \neg P \]
\[ (P \land Q) \Rightarrow \neg (P \lor Q) \]
\[ (P \lor Q) \Rightarrow \neg (P \land Q) \]
Equality

• Two terms refer to the same object
  – Father (John) = Henry
  – Richard has at least two brothers
    \[ \exists x, y \text{ Brother}(x, \text{Richard}) \land \text{Brother}(y, \text{Richard}) \land \neg (x = y) \]
    \[ \exists x, y \text{ Brother}(x, \text{Richard}) \land \text{Brother}(y, \text{Richard}) \]

An Example

• A Tell/Ask interface for a first-order knowledge base
  – Sentences are added with "Tell"
    • These are called assertions
      – Tell (KB, King(John))
      – Tell (KB, Forall x: King(x) => Person(x))
  – Queries are made with "Ask"
    • Ask (KB, King(John))
    • Ask (KB, Person(John))

An Example - Quantified queries

• Ask (KB, exist x: Person(x))
  – KB should return a list of variable/term pairs that satisfy the query

The Wumpus World

• More precise axioms than with propositional logic
  – Percept has five values
  – Time is important
    – A typical sentence
      • Percept ([Stench, Breeze, Glitter, None, None], 5)
  – The actions are terms
    • Turn(right), Turn(left), Forward, Shoot, Grab, Release
  – Computing best action with a query
    • Exist a: BestAction(a, 5)

The Wumpus World - After executing query, KB responds with variable/term list: (a/Grab)

Wumpus World - Defining the environment with [x,y] reference instead of alternative atomic name

  – Adjacency between two squares
    \[ \forall x, y, z, w \text{ Adjacent}(x, y, z, w) \Rightarrow \]
    \[ \{a \mid a \in \{x = 1, y, x = 1, x = y, y = 1, x = y - 1\}\} \]
  – Location of Wumpus is constant: Home (Wumpus)
  – Location of agent changes: At (Agent, [ ], t)

  \[ \forall x, t \text{ At}(Agent, x, t) \land \text{Breeze}(x) \Rightarrow \text{Breeze}(t) \]
Diagnostic Rules

• Rules leading from observed effects to hidden causes
  – Breezy implies pits
    \[ \forall x \text{ Breezy}(x) \Rightarrow \exists y \text{ Adjoins}(x,y) \land \text{Pit}(y) \]
  – Not breezy implies no pits
    \[ \forall x \neg \text{Breezy}(x) \Rightarrow \neg \exists y \text{ Adjoins}(x,y) \land \text{Pit}(y) \]

Causal Rules

• Some hidden property causes percepts to be generated
  – A pit causes adjacent squares to be breezy
    \[ \forall x \text{ Pit}(x) \Rightarrow \forall y \text{ Adjoins}(x,y) \Rightarrow \text{Breezy}(y) \]
  – If an square adjacent to a square is pitless, it will not be breezy
    \[ \forall x \forall y \left( \text{Adjoins}(x,y) \land \neg \text{Pit}(y) \right) \Rightarrow \neg \text{Breezy}(x) \]

Causal Rules

• The causal rules formulate a model
  – Knowledge of how the environment operates
  – Model can be very useful and important and replace straightforward diagnostic approaches

Conclusion

• If the axioms correctly and completely describe the way the world works and the way percepts are produced,
  • then any complete logical inference procedure will infer the strongest possible description of the world state given the available percepts
• The agent designer can focus on getting the knowledge right without worrying about the processes of deduction