Overview of First-Order Logic

Chapter 8

Outline

- Why FOL?
- Syntax of FOL
- Expressing Sentences in FOL
- Wumpus world in FOL
- Knowledge Engineering

Pros and Cons of Propositional Logic (PC)

Pros:

- PC is *declarative*: formulas correspond to assertions.

Pros and Cons of Propositional Logic (PC)

Pros:

- PC is *declarative*: formulas correspond to assertions.
- PC allows incomplete information
  (unlike most data structures and databases)

Pros and Cons of Propositional Logic (PC)

Pros:

- PC is *declarative*: formulas correspond to assertions.
- PC allows incomplete information
  (unlike most data structures and databases)
- PC is *compositional* and *unambiguous*:
  - truth of $B_{1,1} \wedge P_{1,2}$ depends on truth of $B_{1,1}$ and of $P_{1,2}$

Pros and Cons of Propositional Logic (PC)

Pros:

- PC is *declarative*: formulas correspond to assertions.
- PC allows incomplete information
  (unlike most data structures and databases)
- PC is *compositional* and *unambiguous*:
  - truth of $B_{1,1} \wedge P_{1,2}$ depends on truth of $B_{1,1}$ and of $P_{1,2}$
- Meaning in PC is *context-independent*
  - Unlike natural language: Compare "Bring me the iron".
    - "iron" could be an instrument for removing creases from clothes, a golf club, a piece of metal, ...
    - "me" depends on who is doing the talking.

Pros and Cons of PC

Cons:

- PC has limited expressive power
    - E.g., cannot say "pits cause breezes in adjacent squares" except by writing one sentence for each square

First-order logic

- Propositional logic assumes the world is described by *facts*.

First-order logic

- Propositional logic assumes the world is described by *facts*.
- First-order logic assumes the world contains:

First-order logic

- Propositional logic assumes the world is described by *facts*.
- First-order logic assumes the world contains:

  Objects: E.g. people, houses, numbers, colors, hockey games, purchases, . . .

  - Think of nouns in a natural language

First-order logic

- Propositional logic assumes the world is described by *facts*.
- First-order logic assumes the world contains:

Objects: E.g. people, houses, numbers, colors, hockey games, purchases, . . .

  - Think of nouns in a natural language

Relations: E.g. red, round, honest, prime, . . . , brother of, bigger than, likes, occurred after, owns, comes between, . . .

First-order logic

- Propositional logic assumes the world is described by *facts*.
- First-order logic assumes the world contains:

Objects: E.g. people, houses, numbers, colors, hockey games, purchases, . . .

- Think of nouns in a natural language

Relations: E.g. red, round, honest, prime, . . . , brother of, bigger than, likes, occurred after, owns, comes between, . . .

Functions: E.g. father of, best friend, plus, . . .

Aside: Logics in General

There are lots of logics:

Logic	*Ontological Commitment*	*Epistemological Commitment*
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
		true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known fuzzy value
Modal logic (logic of beliefs)	facts, possible worlds	true/false/unknown + necessarily t/f/unkn
Description logic	concepts, roles, objects	true/false/unknown

Syntax of FOL: Basic Elements

- Constants:
  - Stand for objects
  - May be abstract – e.g. a marriage or a purchase
  - E.g. *Wumpus*, 2, *SFU*, . . .

Syntax of FOL: Basic Elements

- Constants:
  - Stand for objects
  - May be abstract – e.g. a marriage or a purchase
  - E.g. Wumpus, 2, SFU, ...
- Predicate symbols:
  - Stand for properties, relations
  - E.g. $\mathit{Block}(A)$, $\mathit{Brother}(\mathit{Richard}, \mathit{John})$, $\mathit{Plus}(2, 3, 5)$, ...

Syntax of FOL: Basic Elements

- Constants:
  - Stand for objects
  - May be abstract – e.g. a marriage or a purchase
  - E.g. $Wumpus$, 2, SFU, ...
- Predicate symbols:
  - Stand for properties, relations
  - E.g. $Block(A)$, $Brother(Richard, John)$, $Plus(2, 3, 5)$, ...
- Functions:
  - Stand for functions
  - E.g. $Sqrt$, $LeftLegOf(John)$, ...

Syntax of FOL: Basic Elements

- Constants: *Wumpus*, 2, *SFU*, ...
- Predicates: *Brother*, *Plus*, ...
- Functions: *Sqrt*, *LeftLegOf*, ...
- Variables: x, y, ...
- Connectives: $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\equiv$.
- Equality: $=$
- Quantifiers: $\forall$, $\exists$

And, strictly speaking, there is punctuation: "(", ")", ",".

Terms and Atomic Sentences

Basic idea with FOL:

- There are *objects* or *things* in the domain being described.
  - *Terms* in the language denote objects.
  - E.g. $JohnQSmith$, 12, $CMPT310$, $favouriteCatOf(John)$, ...

Terms and Atomic Sentences

Basic idea with FOL:

- There are *objects* or *things* in the domain being described.
  - *Terms* in the language denote objects.
  - E.g. $JohnQSmith$, 12, $CMPT310$, $favouriteCatOf(John)$, ...
- One makes *assertions* concerning these objects.
  - *Formulas* in the language express assertions.
  - E.g. $Student(JohnQSmith)$,
    $favouriteCatOf(John) = Fluffy$,
    $\forall x.\ BCUniv(x) \Rightarrow (\neg HasMedSchool(x) \vee x = UBC)$

And that's it!

Terms

- *Term* = logical expression that refers to an object.

Terms

- *Term* = logical expression that refers to an object.
- A term can be:
  - a constant, such as *Chris*, car_{54}, ...
  - a function application such as *LeftLegOf*(*Richard*), *Sqrt*(2), *Sqrt*(*Sqrt*(2)), ...
- A term can contain variables
  - When we get to formulas, we'll want variables to be *quantified*
- A term with no variables is called *ground*

Atomic Sentences

- An *atomic sentences* is the simplest sentence that can be *true* or *false*.

Atomic Sentences

- An *atomic sentences* is the simplest sentence that can be *true* or *false*.
- An atomic sentence is of the form $predicate(term_1, \ldots, term_n)$ or $term_1 = term_2$
- Example atomic sentences (and terms):
  - $Likes(Arvind, ZeNian)$ could be true or false
  - $BrotherOf(Mary, Sue)$ is false (for normal understanding of $BrotherOf$, $Mary$, Sue)
  - $Married(FatherOf(Richard), MotherOf(John))$ could be true or false.
- There may be more than one way to express something. Compare:

  $MotherOf(John, Sue)$ – predicate vs.

  $Sue = MotherOf(John)$ – function.

Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:

  $$\neg S,\ (S_1 \wedge S_2),\ (S_1 \vee S_2),\ (S_1 \Rightarrow S_2),\ (S_1 \equiv S_2)$$

Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:

  $$\neg S, \ (S_1 \wedge S_2), \ (S_1 \vee S_2), \ (S_1 \Rightarrow S_2), \ (S_1 \equiv S_2)$$

- Examples:
  - $Red(car_{54}) \wedge \neg Red(car_{54})$

Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:
  $$\neg S,\ (S_1 \wedge S_2),\ (S_1 \vee S_2),\ (S_1 \Rightarrow S_2),\ (S_1 \equiv S_2)$$
- Examples:
  - $Red(car_{54}) \wedge \neg Red(car_{54})$
  - $Sibling(Joe, Alice) \Rightarrow Sibling(Alice, Joe)$

Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:

  $$\neg S,\ (S_1 \wedge S_2),\ (S_1 \vee S_2),\ (S_1 \Rightarrow S_2),\ (S_1 \equiv S_2)$$

- Examples:
  - $Red(car_{54}) \wedge \neg Red(car_{54})$
  - $Sibling(Joe, Alice) \Rightarrow Sibling(Alice, Joe)$
  - $King(Richard) \vee King(John)$
  - $King(Richard) \Rightarrow \neg King(John)$
  - $Purchase(p) \wedge$
    $Buyer(p) = John \wedge$
    $ObjectType(p) = Bike$
- Semantics is the same as in propositional logic

Variables

- *Student*(*John*) is true or false and says something about a specific individual, John.
- We can be much more flexible if we allow variables which can range over element of the domain.

Variables

- $Student(John)$ is true or false and says something about a specific individual, John.
- We can be much more flexible if we allow variables which can range over element of the domain.
- Now allow sentences of the form:

$$(\forall x S),\ (\exists x S)$$

  - $(\forall x S)$ is true if no matter what x refers to, S is true.
  - $(\exists x S)$ is true if there is some element of the domain for which S is true.

Universal Quantification

Form: $\forall \langle \mathit{variables} \rangle \langle \mathit{sentence} \rangle$

- Allows us to make statements about all objects that have certain properties.
- Everyone at SFU is smart: $\forall x\ At(x, SFU) \Rightarrow Smart(x)$

Universal Quantification

Form: $\forall \langle \mathit{variables} \rangle \langle \mathit{sentence} \rangle$

- Allows us to make statements about all objects that have certain properties.
- Everyone at SFU is smart: $\forall x \; At(x, SFU) \Rightarrow Smart(x)$
- Every number has a successor:
  $\forall x \; NNum(x) \Rightarrow NNum(Succ(x))$

Universal Quantification

Form: $\forall \langle variables \rangle \langle sentence \rangle$

- Allows us to make statements about all objects that have certain properties.
- Everyone at SFU is smart: $\forall x\ At(x, SFU) \Rightarrow Smart(x)$
- Every number has a successor: $\forall x\ NNum(x) \Rightarrow NNum(Succ(x))$
- *Roughly* speaking, equivalent to the *conjunction* of *instantiations* of P

$$\begin{array}{ll} (At(Joe, SFU) \Rightarrow Smart(Joe)) & \wedge \\ (At(Alice, SFU) \Rightarrow Smart(Alice)) & \wedge \\ (At(SFU, SFU) \Rightarrow Smart(SFU)) & \wedge \ldots \end{array}$$

- Aside: Formulas are *finite* in length, so universal quantification in general can't be expressed as a big conjunction.

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x(At(x, SFU) \wedge Smart(x))$$

means

"Everyone is at SFU and everyone is smart"

and not

"Everyone at SFU is smart".

Existential Quantification

Form: $\exists \langle \mathit{variables} \rangle \langle \mathit{sentence} \rangle$

- Allows us to make a statement about an object without naming it.
- Someone at UVic is smart: $\exists x (At(x, UVic) \land Smart(x))$

Existential Quantification

Form: $\exists\langle \mathit{variables} \rangle \langle \mathit{sentence} \rangle$

- Allows us to make a statement about an object without naming it.
- Someone at UVic is smart: $\exists x(At(x, UVic) \wedge Smart(x))$
- There is a SFU student with a top GPA:

Existential Quantification

Form: $\exists \langle \mathit{variables} \rangle \langle \mathit{sentence} \rangle$

- Allows us to make a statement about an object without naming it.
- Someone at UVic is smart: $\exists x(At(x, UVic) \wedge Smart(x))$
- There is a SFU student with a top GPA:
  $\exists x(Student(x) \wedge \forall y(Student(y) \Rightarrow GE(GPA(x), GPA(y))))$

Existential Quantification

Form: $\exists\langle variables\rangle\langle sentence\rangle$

- Allows us to make a statement about an object without naming it.
- Someone at UVic is smart: $\exists x(At(x, UVic) \wedge Smart(x))$
- There is a SFU student with a top GPA:
  $\exists x(Student(x) \wedge \forall y(Student(y) \Rightarrow GE(GPA(x), GPA(y))))$
- *Roughly* speaking, equivalent to the *disjunction* of *instantiations* of P

$$\begin{array}{ll} (At(Joe, UVic) \wedge Smart(Joe)) & \vee \\ (At(Alice, UVic) \wedge Smart(Alice)) & \vee \\ (At(SFU, UVic) \wedge Smart(SFU)) & \vee \ldots \end{array}$$

- But again, we cannot have an infinite disjuntion!

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: Using $\Rightarrow$ as the main connective with $\exists$:

  $$\exists x(At(x, UVic) \Rightarrow Smart(x))$$

  is true if (among other possibilities) there is someone who is not at UVic!
- On the other hand:

  $$\exists x(At(x, UVic) \wedge Smart(x))$$

  is true if there is someone who is at UVic and is smart.

Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$ (why?)

Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$ (why?)
- $\exists x \exists y$ is the same as $\exists y \exists x$ (why?)

Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$ (why?)
- $\exists x \exists y$ is the same as $\exists y \exists x$ (why?)
- $\exists x \forall y$ is *not* the same as $\forall y \exists x$:

Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$ (why?)
- $\exists x \exists y$ is the same as $\exists y \exists x$ (why?)
- $\exists x \forall y$ is *not* the same as $\forall y \exists x$:
  - $\exists x \forall y \; Loves(x, y)$
    "There is a person who loves everyone"
  - $\forall y \exists x \; Loves(x, y)$
    "Everyone is loved by at least one person"

Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$ (why?)
- $\exists x \exists y$ is the same as $\exists y \exists x$ (why?)
- $\exists x \forall y$ is *not* the same as $\forall y \exists x$:
  - $\exists x \forall y \; Loves(x, y)$
    "There is a person who loves everyone"
  - $\forall y \exists x \; Loves(x, y)$
    "Everyone is loved by at least one person"
- *Quantifier duality*: each can be expressed using the other

$$\forall x \; Likes(x, IceCream) \;\equiv\; \neg \exists x \; \neg Likes(x, IceCream)$$
$$\exists x \; Likes(x, Broccoli) \;\equiv\; \neg \forall x \; \neg Likes(x, Broccoli)$$

☞ Like De Morgan's Rule

Expressing Sentences in FOL

- Brothers are siblings

Expressing Sentences in FOL

- Brothers are siblings
  $\forall x, y\ (Brother(x, y) \Rightarrow Sibling(x, y))$.

Expressing Sentences in FOL

- Brothers are siblings
  $\forall x, y \; (Brother(x, y) \Rightarrow Sibling(x, y))$.
- "Sibling" is symmetric

Expressing Sentences in FOL

- Brothers are siblings
  $\forall x, y \; (Brother(x, y) \Rightarrow Sibling(x, y))$.
- "Sibling" is symmetric
  $\forall x, y \; (Sibling(x, y) \equiv Sibling(y, x))$.

Expressing Sentences in FOL

- Brothers are siblings
  $\forall x, y \; (Brother(x, y) \Rightarrow Sibling(x, y)).$
- "Sibling" is symmetric
  $\forall x, y \; (Sibling(x, y) \equiv Sibling(y, x)).$
- One's mother is one's female parent

Expressing Sentences in FOL

- Brothers are siblings
  $\forall x, y\ (Brother(x, y) \Rightarrow Sibling(x, y)).$
- "Sibling" is symmetric
  $\forall x, y\ (Sibling(x, y) \equiv Sibling(y, x)).$
- One's mother is one's female parent
  $\forall x, y\ (Mother(x, y) \equiv (Female(x) \land Parent(x, y))).$

Expressing Sentences in FOL

- Brothers are siblings
  $\forall x, y\ (Brother(x, y) \Rightarrow Sibling(x, y))$.
- "Sibling" is symmetric
  $\forall x, y\ (Sibling(x, y) \equiv Sibling(y, x))$.
- One's mother is one's female parent
  $\forall x, y\ (Mother(x, y) \equiv (Female(x) \wedge Parent(x, y)))$.
- A first cousin is a child of a parent's sibling

Expressing Sentences in FOL

- Brothers are siblings
  $\forall x, y \ (Brother(x, y) \Rightarrow Sibling(x, y))$.
- "Sibling" is symmetric
  $\forall x, y \ (Sibling(x, y) \equiv Sibling(y, x))$.
- One's mother is one's female parent
  $\forall x, y \ (Mother(x, y) \equiv (Female(x) \wedge Parent(x, y)))$.
- A first cousin is a child of a parent's sibling
  $\forall x, y \ (FirstCousin(x, y) \equiv$
  $\exists p, ps(Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)))$

Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

- Compare: “sibling is symmetric” and “a brother is a sibling”.

Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

- Compare: "sibling is symmetric" and "a brother is a sibling".
  $\forall x, y(Sibling(x, y) \equiv Sibling(y, x))$.
  $\forall x, y(Brother(x, y) \Rightarrow Sibling(x, y))$.

Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

- Compare: "sibling is symmetric" and "a brother is a sibling".
  $\forall x, y(Sibling(x, y) \equiv Sibling(y, x))$.
  $\forall x, y(Brother(x, y) \Rightarrow Sibling(x, y))$.
- Compare: "a dog is a mammal" and "Anne is a student".

Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

- Compare: "sibling is symmetric" and "a brother is a sibling".
  $\forall x, y(Sibling(x, y) \equiv Sibling(y, x))$.
  $\forall x, y(Brother(x, y) \Rightarrow Sibling(x, y))$.
- Compare: "a dog is a mammal" and "Anne is a student".
  $\forall x(Dog(x) \Rightarrow Mammal(x))$.
  $Student(Anne)$.

Equality

- $t_1 = t_2$ is true iff t_1 and t_2 refer to the same object

Equality

- $t_1 \; = \; t_2$ is true iff t_1 and t_2 refer to the same object
- E.g., definition of *Sibling* in terms of *Parent*:

$$\begin{aligned}\forall x, y\ Sibling(x,y)\ \equiv\ [&\neg(x = y) \land \\ &\exists m, f\ (\neg(m = f) \land \\ &\quad Parent(m,x) \land Parent(f,x) \land \\ &\quad Parent(m,y) \land Parent(f,y))]\end{aligned}$$

Equality

- $t_1 = t_2$ is true iff t_1 and t_2 refer to the same object
- E.g., definition of *Sibling* in terms of *Parent*:

$$\begin{aligned} \forall x, y\ Sibling(x, y) \equiv\ & [\neg(x = y) \land \\ & \exists m, f\ (\neg(m = f) \land \\ & \quad Parent(m, x) \land Parent(f, x) \land \\ & \quad Parent(m, y) \land Parent(f, y))] \end{aligned}$$

- Aside: Better is:

$$\begin{aligned} \forall x, y\ Sibling(x, y) \equiv [\neg(x = y) \land \exists m, f\ (Mother(m, x) \land \\ Father(f, x) \land Mother(m, y) \land Father(f, y))] \end{aligned}$$

$+$ definitions of *Mother* and *Father*.

☞ As with programming, it is important *how* you express a domain.

Equality

Don't confuse $\equiv$ and $=$.

Equality

Don't confuse $\equiv$ and $=$.

- $\alpha \equiv \beta$ says that α and β share the same truth value
    - $\equiv$ is a relation between *formulas*
    - E.g. $a \wedge b \ \equiv \ b \wedge a$.

Equality

Don't confuse $\equiv$ and $=$.

- $\alpha \equiv \beta$ says that α and β share the same truth value
  - $\equiv$ is a relation between *formulas*
  - E.g. $a \wedge b \;\equiv\; b \wedge a$.
- $t_1 = t_2$ says that t_1 and t_2 refer to the same individual
  - $=$ is a relation between *terms*
  - E.g. $CapitalOf(BC) = Victoria$.

Interacting with FOL KBs

- An agent needs to interact with its KB.
- Regarding a KB as an ADT, there are two primary operations, *TELL* and *ASK*.

Interacting with FOL KBs

- An agent needs to interact with its KB.
- Regarding a KB as an ADT, there are two primary operations, *TELL* and *ASK*.
- We want to *TELL* things to the KB, e.g.

  $TELL(KB, \forall x(Grad(x) \Rightarrow Student(x)))$

  $TELL(KB, Grad(Alice))$

  - These sentences are *assertions*

Interacting with FOL KBs

- An agent needs to interact with its KB.
- Regarding a KB as an ADT, there are two primary operations, *TELL* and *ASK*.
- We want to *TELL* things to the KB, e.g.
  $TELL(KB, \forall x(Grad(x) \Rightarrow Student(x)))$
  $TELL(KB, Grad(Alice))$
  - These sentences are *assertions*
- We also want to *ASK* things of a *KB*,
  $ASK(KB, \exists x\ Student(x))$
  - These are *queries* or *goals*
  - The KB should output x where $Student(x)$ is true:
    $\{x/Alice, \ldots\}$

Interacting with FOL KBs: The Wumpus World

- Suppose a wumpus-world agent is using a FOL KB and perceives a smell and a breeze (but no glitter) at $t = 5$:

Interacting with FOL KBs: The Wumpus World

- Suppose a wumpus-world agent is using a FOL KB and perceives a smell and a breeze (but no glitter) at $t = 5$:
- Express by the percept sentence:
  $Tell(KB, Percept([Smell, Breeze, None, None, None], 5))$

Interacting with FOL KBs: The Wumpus World

- Suppose a wumpus-world agent is using a FOL KB and perceives a smell and a breeze (but no glitter) at $t = 5$:
- Express by the percept sentence:
  $Tell(KB, Percept([Smell, Breeze, None, None, None], 5))$
- Then:
  $Ask(KB, \exists a Action(a, 5))$
  - I.e., does KB entail any particular actions at $t = 5$?
  - Ask solves this and returns $\{a/Shoot\}$

Knowledge in the Wumpus World

- Need to specify axioms about the wumpus world; for example:
- *"Perception"*

  $\forall b, g, t, m, c \; Percept([Smell, b, g, m, c], t) \Rightarrow Smelt(t)$

  $\forall s, b, t, m, c \; Percept([s, b, Glitter, m, c], t) \Rightarrow AtGold(t)$

  ☞ Aside: Must keep track of time, and so $Smelt(t)$.

Knowledge in the Wumpus World

- Need to specify axioms about the wumpus world; for example:
- *"Perception"*

  $\forall b, g, t, m, c \; Percept([Smell, b, g, m, c], t) \Rightarrow Smelt(t)$

  $\forall s, b, t, m, c \; Percept([s, b, Glitter, m, c], t) \Rightarrow AtGold(t)$

  ☞ Aside: Must keep track of time, and so $Smelt(t)$.

- *Reflex*: $\forall t \; AtGold(t) \Rightarrow Action(Grab, t)$

Knowledge in the Wumpus World

- Need to specify axioms about the wumpus world; for example:
- *"Perception"*

  $\forall b, g, t, m, c \; Percept([Smell, b, g, m, c], t) \Rightarrow Smelt(t)$

  $\forall s, b, t, m, c \; Percept([s, b, Glitter, m, c], t) \Rightarrow AtGold(t)$

  ☞ Aside: Must keep track of time, and so $Smelt(t)$.
- *Reflex*: $\forall t \; AtGold(t) \Rightarrow Action(Grab, t)$
- *Reflex with internal state*: Do we have the gold already?

  $\forall t \; AtGold(t) \land \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

Knowledge in the Wumpus World

- Need to specify axioms about the wumpus world; for example:
- *"Perception"*

  $\forall b, g, t, m, c\ Percept([Smell, b, g, m, c], t) \Rightarrow Smelt(t)$

  $\forall s, b, t, m, c\ Percept([s, b, Glitter, m, c], t) \Rightarrow AtGold(t)$

  ☞ Aside: Must keep track of time, and so $Smelt(t)$.
- *Reflex*: $\forall t\ AtGold(t) \Rightarrow Action(Grab, t)$
- *Reflex with internal state*: Do we have the gold already?

  $\forall t\ AtGold(t) \land \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$
- Note that $Holding(Gold, t)$ cannot be observed

  ☞ must keep track of change

Knowledge in the Wumpus World

- Need to specify axioms about the wumpus world; for example:
- *"Perception"*

  $\forall b, g, t, m, c \ Percept([Smell, b, g, m, c], t) \Rightarrow Smelt(t)$
  $\forall s, b, t, m, c \ Percept([s, b, Glitter, m, c], t) \Rightarrow AtGold(t)$

  ☞ Aside: Must keep track of time, and so $Smelt(t)$.

- *Reflex*: $\forall t \ AtGold(t) \Rightarrow Action(Grab, t)$
- *Reflex with internal state*: Do we have the gold already?

  $\forall t \ AtGold(t) \land \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

- Note that $Holding(Gold, t)$ cannot be observed

  ☞ must keep track of change

- Q: If we know $Holding(Gold, t)$ can we conclude $Holding(Gold, t + 1)$?

Knowledge in the Wumpus World

- Need to specify axioms about the wumpus world; for example:
- *"Perception"*

  $\forall b, g, t, m, c\ Percept([Smell, b, g, m, c], t) \Rightarrow Smelt(t)$

  $\forall s, b, t, m, c\ Percept([s, b, Glitter, m, c], t) \Rightarrow AtGold(t)$

  ☞ Aside: Must keep track of time, and so $Smelt(t)$.

- *Reflex*: $\forall t\ AtGold(t) \Rightarrow Action(Grab, t)$
- *Reflex with internal state*: Do we have the gold already?

  $\forall t\ AtGold(t) \wedge \neg Holding(Gold, t) \Rightarrow Action(Grab, t)$

- Note that $Holding(Gold, t)$ cannot be observed

  ☞ must keep track of change

- Q: If we know $Holding(Gold, t)$ can we conclude $Holding(Gold, t+1)$?
  - Ans: No

Representing Information

- Need to remember properties of locations:
  $\forall x, t \; At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$
  $\forall x, t \; At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$
- Need to be careful that *all* information is represented.
  Consider "Squares are breezy near a pit":

Representing Information

- Need to remember properties of locations:
  $\forall x, t\ At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$
  $\forall x, t\ At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$
- Need to be careful that *all* information is represented. Consider "Squares are breezy near a pit":
  - *Diagnostic* rule – infer cause from effect
    $\forall y\ Breezy(y) \Rightarrow \exists x Pit(x) \wedge Adjacent(x, y)$
  - *Causal* rule – infer effect from cause
    $\forall x, y\ Pit(x) \wedge Adjacent(x, y) \Rightarrow Breezy(y)$

Representing Information

- Need to remember properties of locations:
  $\forall x, t\ At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$
  $\forall x, t\ At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$
- Need to be careful that *all* information is represented.
  Consider "Squares are breezy near a pit":
  - *Diagnostic* rule – infer cause from effect
    $\forall y\ Breezy(y) \Rightarrow \exists x Pit(x) \wedge Adjacent(x, y)$
  - *Causal* rule – infer effect from cause
    $\forall x, y\ Pit(x) \wedge Adjacent(x, y) \Rightarrow Breezy(y)$
- Neither of these is complete – e.g., the causal rule doesn't say whether squares far away from pits can be breezy
- *Definition* for the *Breezy* predicate:
  $\forall y\ Breezy(y) \equiv [\exists x\ Pit(x) \wedge Adjacent(x, y)]$

Knowledge Engineering in FOL

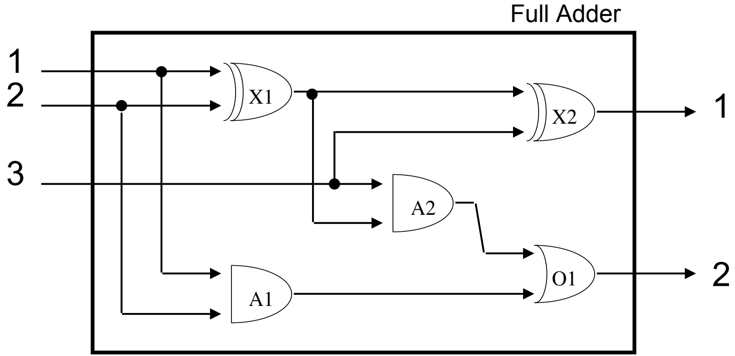
1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base.

Knowledge Engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base.

Aside: This is pretty much the same as designing a database schema + instance.

The Electronic Circuits Domain



The Electronic Circuits Domain

1. Identify the task

The Electronic Circuits Domain

1. Identify the task
    - Does the circuit actually add properly? (circuit verification)

The Electronic Circuits Domain

1. Identify the task
    - Does the circuit actually add properly? (circuit verification)
2. Assemble the relevant knowledge

The Electronic Circuits Domain

1. Identify the task
   - Does the circuit actually add properly? (circuit verification)
2. Assemble the relevant knowledge
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - Irrelevant: size, shape, color, cost of gates

The Electronic Circuits Domain

1. Identify the task
   - Does the circuit actually add properly? (circuit verification)
2. Assemble the relevant knowledge
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - Irrelevant: size, shape, color, cost of gates
3. Decide on a vocabulary

The Electronic Circuits Domain

1. Identify the task
   - Does the circuit actually add properly? (circuit verification)
2. Assemble the relevant knowledge
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - Irrelevant: size, shape, color, cost of gates
3. Decide on a vocabulary
   - Different possibilities:
     - Function: $Type(X_1) = XOR$
     - Binary predicate: $Type(X_1, XOR)$
     - Unary predicate: $XOR(X_1)$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2\ Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$
- $\forall p \; Signal(p) = 1 \vee Signal(p) = 0$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$
- $\forall p \; Signal(p) = 1 \lor Signal(p) = 0$
- $1 \neq 0$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$
- $\forall p \; Signal(p) = 1 \lor Signal(p) = 0$
- $1 \neq 0$
- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Connected(p_2, p_1)$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \ Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$
- $\forall p \ Signal(p) = 1 \lor Signal(p) = 0$
- $1 \neq 0$
- $\forall p_1, p_2 \ Connected(p_1, p_2) \Rightarrow Connected(p_2, p_1)$
- $\forall g \ \ Type(g) = OR \Rightarrow$
  $Signal(Out(1, g)) = 1 \ \equiv \ \exists n \ Signal(In(n, g)) = 1$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$
- $\forall p \; Signal(p) = 1 \vee Signal(p) = 0$
- $1 \neq 0$
- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Connected(p_2, p_1)$
- $\forall g \;\; Type(g) = OR \Rightarrow$
  $Signal(Out(1, g)) = 1 \;\equiv\; \exists n \; Signal(In(n, g)) = 1$
- $\forall g \;\; Type(g) = AND \Rightarrow$
  $Signal(Out(1, g)) = 0 \;\equiv\; \exists n \; Signal(In(n, g)) = 0$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \ Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$
- $\forall p \ Signal(p) = 1 \lor Signal(p) = 0$
- $1 \neq 0$
- $\forall p_1, p_2 \ Connected(p_1, p_2) \Rightarrow Connected(p_2, p_1)$
- $\forall g \ \ Type(g) = OR \Rightarrow$
  $Signal(Out(1, g)) = 1 \ \equiv \ \exists n \ Signal(In(n, g)) = 1$
- $\forall g \ \ Type(g) = AND \Rightarrow$
  $Signal(Out(1, g)) = 0 \ \equiv \ \exists n \ Signal(In(n, g)) = 0$
- $\forall g \ \ Type(g) = XOR \Rightarrow$
  $Signal(Out(1, g)) = 1 \ \equiv \ Signal(In(1, g)) \neq Signal(In(2, g))$

The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2)$
- $\forall p \; Signal(p) = 1 \vee Signal(p) = 0$
- $1 \neq 0$
- $\forall p_1, p_2 \; Connected(p_1, p_2) \Rightarrow Connected(p_2, p_1)$
- $\forall g \;\; Type(g) = OR \Rightarrow$
  $Signal(Out(1, g)) = 1 \;\equiv\; \exists n \; Signal(In(n, g)) = 1$
- $\forall g \;\; Type(g) = AND \Rightarrow$
  $Signal(Out(1, g)) = 0 \;\equiv\; \exists n \; Signal(In(n, g)) = 0$
- $\forall g \;\; Type(g) = XOR \Rightarrow$
  $Signal(Out(1, g)) = 1 \;\equiv\; Signal(In(1, g)) \neq Signal(In(2, g))$
- $\forall g \;\; Type(g) = NOT \Rightarrow Signal(Out(1, g)) \neq Signal(In(1, g))$

The Electronic Circuits Domain

5. Encode the specific problem instance:

$$
\begin{array}{ll}
Type(X_1) = XOR & Type(X_2) = XOR \\
Type(A_1) = AND & Type(A_2) = AND \\
Type(O_1) = OR &
\end{array}
$$

$$
\begin{array}{ll}
Connected(Out(1,X_1), In(1,X_2)) & Connected(In(1,C_1), In(1,X_1)) \\
Connected(Out(1,X_1), In(2,A_2)) & Connected(In(1,C_1), In(1,A_1)) \\
Connected(Out(1,A_2), In(1,O_1)) & Connected(In(2,C_1), In(2,X_1)) \\
Connected(Out(1,A_1), In(2,O_1)) & Connected(In(2,C_1), In(2,A_1)) \\
Connected(Out(1,X_2), Out(1,C_1)) & Connected(In(3,C_1), In(2,X_2)) \\
Connected(Out(1,O_1), Out(2,C_1)) & Connected(In(3,C_1), In(1,A_2))
\end{array}
$$

The Electronic Circuits Domain

6. Pose queries to the inference procedure
   - E.g. what are the outputs, given a set of input signals?
   - I.e.

$$\exists o_1, o_2$$
$$(Signal(In(1, C_1)) = 1 \wedge Signal(In(2, C_1)) = 0 \wedge$$
$$Signal(In(3, C_1)) = 1)$$
$$\Rightarrow$$
$$(Signal(Out(1, C_1)) = o_1 \wedge Signal(Out(2, C_1)) = o_2)$$

The Electronic Circuits Domain

6. Pose queries to the inference procedure
   - E.g. what are the outputs, given a set of input signals?
   - I.e.

     $\exists o_1, o_2$
     $(Signal(In(1, C_1)) = 1 \wedge Signal(In(2, C_1)) = 0 \wedge$
     $Signal(In(3, C_1)) = 1)$
     $\Rightarrow$
     $(Signal(Out(1, C_1)) = o_1 \wedge Signal(Out(2, C_1)) = o_2)$
7. Debug the knowledge base
   - E.g. may have omitted assertions like $0 \neq 1$.

Summary

- First-order logic:
  - Much more expressive than propositional logic
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers
- FOL is harder to reason with
  - Undecidable in general