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.

- PC allows incomplete information (unlike most data structures and databases).

- PC is compositional and unambiguous:
  - Truth of $B_1 \land P_1$ depends on truth of $B_1$ and of $P_1$.

- 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 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 $B_{1,1} \land P_{1,2}$ 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 $B_{1,1} \land P_{1,2}$ 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:
  - Objects: E.g. people, houses, numbers, colors, hockey games, purchases, . . .
  - 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, . . .
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 \textit{facts}.
- First-order logic assumes the world contains:

  \textbf{Objects}: E.g. people, houses, numbers, colors, hockey games, purchases, \ldots
  
  - Think of nouns in a natural language

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

<table>
<thead>
<tr>
<th>Logic</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Propositional logic</strong></td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>facts + degree of truth</td>
<td>known fuzzy value</td>
</tr>
<tr>
<td>Relevance logic</td>
<td>facts</td>
<td>true/unknown/inconsistent</td>
</tr>
<tr>
<td>Modal logic (logic of beliefs)</td>
<td>facts, possible worlds</td>
<td>true/false/unknown + necessarily t/f/unkn</td>
</tr>
<tr>
<td>Description logic</td>
<td>concepts, roles, objects</td>
<td>true/false/unknown</td>
</tr>
</tbody>
</table>

...and lots of others!
Syntax of FOL: Basic Elements

- **Constants:**
  - Stand for objects
  - May be abstract – e.g. a marriage or a purchase
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), . . .
Syntax of FOL: Basic Elements

- **Constants:**
  - Stand for objects
  - May be abstract – e.g. a marriage or a purchase

- **Predicate symbols:**
  - Stand for properties, relations

- **Functions:**
  - Stand for functions
  - E.g. *Sqrt*, *LeftLegOf*(John), . . .
Syntax of FOL: Basic Elements

- Constants: \textit{Wumpus}, 2, \textit{SFU}, \ldots
- Predicates: \textit{Brother}, \textit{Plus}, \ldots
- Functions: \textit{Sqrt}, \textit{LeftLegOf}, \ldots
- Variables: \(x, y, \ldots\)
- Connectives: \(\land, \lor, \neg, \Rightarrow, \equiv\)
- 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), ...

- There are *assertions* concerning these objects.
  - *Assertions* are expressed by *formulas*.
  - E.g. *Student*(JohnQSmith), *favouriteCatOf*(John) = *Fluffy*, ...
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), ...
- There are *assertions* concerning these objects.
  - Assertions are expressed by *formulas*.
  - E.g. *Student*(JohnQSmith),
    *favouriteCatOf*(John) = *Fluffy*,
    \( \forall x. \text{BCUniv}(x) \Rightarrow (\neg \text{HasMedSchool}(x) \lor x = \text{UBC}) \)

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 *LeftLeg*ₐₜ*(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*.

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.
Atomic Sentences

- An *atomic sentences* is the simplest sentence that can be *true* or *false*.
- An atomic sentence is of the form $\text{predicate}(\text{term}_1, \ldots, \text{term}_n)$ or $\text{term}_1 = \text{term}_2$
- Example atomic sentences (and terms):
  - $\text{Likes}(\text{Arvind}, \text{ZeNian})$ could be true or false
  - $\text{BrotherOf}(\text{Mary}, \text{Sue})$ is false (for normal understanding of $\text{BrotherOf}$, $\text{Mary}$, $\text{Sue}$)
  - $\text{Married}(\text{FatherOf}(\text{Richard}), \text{MotherOf}(\text{John}))$ could be true or false.
- There may be more than one way to express something.
  Compare:
  - $\text{MotherOf}(\text{John}, \text{Sue})$ — predicate vs. $\text{Sue} = \text{MotherOf}(\text{John})$ — function.
Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:
  \( \neg S, (S_1 \land S_2), (S_1 \lor S_2), (S_1 \Rightarrow S_2), (S_1 \equiv S_2) \)
Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:
  \[ \neg S, (S_1 \land S_2), (S_1 \lor S_2), (S_1 \Rightarrow S_2), (S_1 \equiv S_2) \]
- Examples:
  - \( Red(car_{54}) \land \neg Red(car_{54}) \)
Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:
  \[ \neg S, (S_1 \land S_2), (S_1 \lor S_2), (S_1 \Rightarrow S_2), (S_1 \equiv S_2) \]

- Examples:
  - \( Red(car_{54}) \land \neg Red(car_{54}) \)
  - \( Sibling(Joe, Alice) \Rightarrow Sibling(Alice, Joe) \)
Complex Sentences

- Complex sentences are made from atomic sentences using the connectives of propositional logic:
  \( \neg S, (S_1 \land S_2), (S_1 \lor S_2), (S_1 \Rightarrow S_2), (S_1 \equiv S_2) \)
- Examples:
  - \( Red(car_{54}) \land \neg Red(car_{54}) \)
  - \( Sibling(Joe, Alice) \Rightarrow Sibling(Alice, Joe) \)
  - \( King(Richard) \lor King(John) \)
  - \( King(Richard) \Rightarrow \neg King(John) \)
  - \( Purchase(p) \land Buyer(p) = John \land ObjectType(p) = Bike \)
- 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

- \textit{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: $\forall \langle variables \rangle \langle sentence \rangle$

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

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

- 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: \( \forall \langle \text{variables} \rangle \langle \text{sentence} \rangle \)

- 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 \( P \)
  \[
  (At(Joe, SFU) \Rightarrow Smart(Joe)) \land \\
  (At(Alice, SFU) \Rightarrow Smart(Alice)) \land \\
  (At(SFU, SFU) \Rightarrow Smart(SFU)) \land \ldots
  \]
- 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: $\exists \langle variables \rangle \langle sentence \rangle$

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

Roughly speaking, equivalent to the disjunction of instantiations of
$\text{At}(\text{Joe}, \text{UVic}) \land \text{Smart}(\text{Joe}) \lor
\text{At}(\text{Alice}, \text{UVic}) \land \text{Smart}(\text{Alice}) \lor
\text{At}(\text{SFU}, \text{UVic}) \land \text{Smart}(\text{SFU}) \lor \ldots$

But again, we cannot have an infinite disjunction!
Existential Quantification

Form: \( \exists \langle \text{variables} \rangle \langle \text{sentence} \rangle \)

- Allows us to make a statement about an object without naming it.
- Someone at UVic is smart: \( \exists x (At(x, \text{UVic}) \land \text{Smart}(x)) \)
- There is a SFU student with a top GPA:
  \( \exists x (\text{Student}(x) \land \forall y (\text{Student}(y) \Rightarrow \text{GE}(\text{GPA}(x), \text{GPA}(y)))) \)
- *Roughly* speaking, equivalent to the *disjunction* of *instantiations* of \( P \)
  \[ (At(Joe, \text{UVic}) \land \text{Smart}(Joe)) \lor (At(Alice, \text{UVic}) \land \text{Smart}(Alice)) \lor (At(SFU, \text{UVic}) \land \text{Smart}(SFU)) \lor \ldots \]
- But again, we cannot have an infinite disjunction!
Another common mistake to avoid

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

$$\exists x \,(At(x, \textit{UVic}) \Rightarrow \textit{Smart}(x))$$

is true if (among other possibilities) there is someone who is not at UVic!

- On the other hand:

$$\exists x \,(At(x, \textit{UVic}) \land \textit{Smart}(x))$$

is true if there is someone who is at UVic and is smart.
Properties of Quantifiers

- \( \forall x \forall y \) is the same as \( \forall y \forall x \)
Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$
- $\exists x \exists y$ is the same as $\exists y \exists x$
Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$
- $\exists x \exists y$ is the same as $\exists y \exists x$
- $\exists x \forall y$ is *not* the same as $\forall y \exists x$:

"There is a person who likes everyone"

"Everyone is liked by at least one person"

Quantifier duality: each can be expressed using the other

$\forall x \text{Likes}(x, \text{IceCream}) \equiv \neg \exists x \neg \text{Likes}(x, \text{IceCream})$

$\exists x \text{Likes}(x, \text{Broccoli}) \equiv \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

Like De Morgan’s Rule
Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$
- $\exists x \exists y$ is the same as $\exists y \exists x$
- $\exists x \forall y$ is not the same as $\forall y \exists x$:
  - $\exists x \forall y \textit{Likes}(x, y)$
    “There is a person who likes everyone”
  - $\forall y \exists x \textit{Likes}(x, y)$
    “Everyone is liked by at least one person”
Properties of Quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$
- $\exists x \exists y$ is the same as $\exists y \exists x$
- $\exists x \forall y$ is *not* the same as $\forall y \exists x$:
  - $\exists x \forall y \text{ Likes}(x, y)$
    “There is a person who likes everyone”
  - $\forall y \exists x \text{ Likes}(x, y)$
    “Everyone is liked by at least one person”
- **Quantifier duality**: each can be expressed using the other
  - $\forall x \text{ Likes}(x, \text{ IceCream}) \equiv \neg \exists x \neg \text{ Likes}(x, \text{ IceCream})$
  - $\exists x \text{ Likes}(x, \text{ Broccoli}) \equiv \neg \forall x \neg \text{ Likes}(x, \text{ Broccoli})$

⚠️ Like De Morgan’s Rule
Expressing Sentences in FOL

• Brothers are siblings
Expressing Sentences in FOL

- Brothers are siblings
  \[ \forall x, y \ (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)). \]
Expressing Sentences in FOL

- Brothers are siblings
  \[ \forall x, y \ (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)) \].

- “Sibling” is symmetric

\[ \forall x, y \ (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)) \].
Expressing Sentences in FOL

- Brothers are siblings
  \[ \forall x, y \ (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)) \].

- “Sibling” is symmetric
  \[ \forall x, y \ (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)) \].
Expressing Sentences in FOL

• Brothers are siblings
  \( \forall x, y \ (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)) \).

• “Sibling” is symmetric
  \( \forall x, y \ (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)) \).

• One’s mother is one’s female parent
  \( \forall x, y \ (\text{Mother}(x, y) \equiv \text{Female}(x) \land \text{Parent}(x, y)) \).
Expressing Sentences in FOL

- Brothers are siblings
  \( \forall x, y \ (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)) \).

- “Sibling” is symmetric
  \( \forall x, y \ (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)) \).

- One’s mother is one’s female parent
  \( \forall x, y \ (\text{Mother}(x, y) \equiv (\text{Female}(x) \land \text{Parent}(x, y))) \).
Expressing Sentences in FOL

- Brothers are siblings
  \[ \forall x, y \ (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)). \]

- “Sibling” is symmetric
  \[ \forall x, y \ (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)). \]

- One’s mother is one’s female parent
  \[ \forall x, y \ (\text{Mother}(x, y) \equiv (\text{Female}(x) \land \text{Parent}(x, y))). \]

- A first cousin is a child of a parent’s sibling
Expressing Sentences in FOL

- Brothers are siblings
  \[ \forall x, y (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)). \]

- “Sibling” is symmetric
  \[ \forall x, y (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)). \]

- One’s mother is one’s female parent
  \[ \forall x, y (\text{Mother}(x, y) \equiv (\text{Female}(x) \land \text{Parent}(x, y))). \]

- A first cousin is a child of a parent’s sibling
  \[ \forall x, y (\text{FirstCousin}(x, y) \equiv \exists p, ps (\text{Parent}(p, x) \land \text{Sibling}(ps, p) \land \text{Parent}(ps, y))). \]
Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

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

\[
\forall x, y (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x))
\]

\[
\forall x, y (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y))
\]

\[
\forall x (\text{Dog}(x) \Rightarrow \text{Mammal}(x))
\]

\[
\text{Student}(\text{Anne})
\]
Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

- Compare: “sibling is symmetric” and “a brother is a sibling”.
  \[ \forall x, y (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)) \]
  \[ \forall x, y (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)) \]

- Compare: “a dog is a mammal” and “Anne is a student”.
  \[ \forall x (\text{Dog}(x) \Rightarrow \text{Mammal}(x)) \]
  \[ \text{Student}(\text{Anne}) \]
Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

- Compare: “sibling is symmetric” and “a brother is a sibling”.
  \[
  \forall x, y (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)) \\
  \forall x, y (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y))
  \]

- Compare: “a dog is a mammal” and “Anne is a student”.
  \[
  \forall x (\text{Dog}(x) \Rightarrow \text{Mammal}(x)) \\
  \text{Student}(\text{Anne})
  \]
Expressing Sentences in FOL

Natural language is highly ambiguous, and FOL removes ambiguity.

- Compare: “sibling is symmetric” and “a brother is a sibling”.
  \[ \forall x, y (\text{Sibling}(x, y) \equiv \text{Sibling}(y, x)) \]
  \[ \forall x, y (\text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y)) \]

- Compare: “a dog is a mammal” and “Anne is a student”.
  \[ \forall x (\text{Dog}(x) \Rightarrow \text{Mammal}(x)) \]
  \[ \text{Student}(\text{Anne}) \]
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 $\text{Sibling}$ in terms of $\text{Parent}$:

$$\forall x, y \ Sibling(x, y) \equiv [\neg (x = y) \land \exists m, f \ (\neg (m = f) \land \neg (m = f) \land Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y))]$$
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 $Parent$:
  \[
  \forall x, y \ Sibling(x, y) \equiv \neg(x = y) \land \\
  \exists m, f \ (\neg(m = f) \land \\
  Parent(m, x) \land Parent(f, x) \land \\
  Parent(m, y) \land Parent(f, y))
  \]

• Aside: Better is:
  \[
  \forall x, y \ Sibling(x, y) \equiv \neg(x = y) \land \exists m, f \ (Mother(m, x) \land \\
  Father(f, x) \land Mother(m, y) \land Father(f, y))
  \]

  + definitions of $Mother$ and $Father$.

※ As with programming, it is important how you express a domain.
Equality

Don’t confuse $\equiv$ and $\ =$. 

$\alpha \equiv \beta$ says that $\alpha$ and $\beta$ have the same truth value

$\equiv$ is a relation between formulas

E.g. $a \land b \equiv b \land a$.

$t_1 = t_2$ says that $t_1$ and $t_2$ refer to the same individual

$=$ is a relation between terms

E.g. CapitalOf(BC) = Victoria.
Equality

Don’t confuse $\equiv$ and $=$.

- $\alpha \equiv \beta$ says that $\alpha$ and $\beta$ have the same truth value
  - $\equiv$ is a relation between formulas
  - E.g. $a \land b \equiv b \land a$.

- $t_1 = t_2$ says that $t_1$ and $t_2$ refer to the same individual
  - $=$ is a relation between terms
  - E.g. $\text{CapitalOf}(\text{BC}) = \text{Victoria}$.
Don’t confuse $\equiv$ and $=.$

- $\alpha \equiv \beta$ says that $\alpha$ and $\beta$ have the same truth value
  - $\equiv$ is a relation between *formulas*
  - E.g. $a \land b \equiv b \land a$.

- $t_1 = t_2$ says that $t_1$ and $t_2$ refer to the same individual
  - $=$ is a relation between *terms*
  - E.g. $\text{CapitalOf}(BC) = \text{Victoria}$.
Interacting with FOL KBs

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

\[
\text{TELL}\left(\text{KB}, \forall x (\text{Grad}(x) \Rightarrow \text{Student}(x))\right)\\
\text{TELL}\left(\text{KB}, \text{Grad}(\text{Alice})\right)
\]

These sentences are assertions.

\[
\text{ASK}\left(\text{KB}, \exists x \text{ Student}(x)\right)
\]

These are queries or goals.

The KB should output \(x\) where \(\text{Student}(x)\) is true:
\[
\{x/\text{Alice}, \ldots\}
\]
Interacting with FOL KBs

- An agent needs to interact with its KB.
- Regarding a KB as an ADT, there are two primary operations, \textit{TELL} and \textit{ASK}.
- We want to \textit{TELL} things to the KB, e.g.
  \[
  \text{TELL}(KB, \forall x (Grad(x) \Rightarrow Student(x)))
  \]
  \[
  \text{TELL}(KB, Grad(Alice))
  \]
- These sentences are \textit{assertions}.
Interacting with FOL KBs

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

\[
\text{Tell}(KB, \text{Percept}([\text{Smell, Breeze, None, None, None, None}], 5))
\]
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:
  $$\text{Tell}(KB, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}, \text{None}, \text{None}], 5))$$

- Then:
  $$\text{Ask}(KB, \exists a \text{Action}(a, 5))$$
  - I.e., does $KB$ entail any particular actions at $t = 5$?
  - $Ask$ solves this and returns $\{a/\text{Shoot}\}$
Knowledge in the Wumpus World

- Need to specify axioms about the wumpus world; for example:
- "Perception to knowledge"
  \[ \forall b, g, t, m, c \ Percept([\text{Smell}, b, g, m, c], t) \Rightarrow \text{Smelt}(t) \]
  \[ \forall s, b, t, m, c \ Percept([s, b, \text{Glitter}, m, c], t) \Rightarrow \text{AtGold}(t) \]

  Aside: Must keep track of time, and so \text{Smelt}(t).
Knowledge in the Wumpus World

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

  "Perception to knowledge"
  \[ \forall b, g, t, m, c \text{ Percept}([\text{Smell}, b, g, m, c], t) \Rightarrow \text{Smelt}(t) \]
  \[ \forall s, b, t, m, c \text{ Percept}([s, b, \text{Glitter}, m, c], t) \Rightarrow \text{AtGold}(t) \]

  Aside: Must keep track of time, and so \text{Smelt}(t).

- Reflex action: \[ \forall t \text{ AtGold}(t) \Rightarrow \text{Action} (\text{Grab}, t) \]
Knowledge in the Wumpus World

• Need to specify axioms about the wumpus world; for example:

• “Perception to knowledge”
  \( \forall b, g, t, m, c \text{ Percept}([\text{Smell}, b, g, m, c], t) \Rightarrow \text{Smelt}(t) \)
  \( \forall s, b, t, m, c \text{ Percept}([s, b, \text{Glitter}, m, c], t) \Rightarrow \text{AtGold}(t) \)

  ❧ Aside: Must keep track of time, and so \( \text{Smelt}(t) \).

• Reflex action: \( \forall t \text{ AtGold}(t) \Rightarrow \text{Action(Grab, t)} \)

• Reflex action with internal state:
  Do we have the gold already?
  \( \forall t \text{ AtGold}(t) \land \neg \text{Holding(Gold, t)} \Rightarrow \text{Action(Grab, t)} \)
Knowledge in the Wumpus World

• Need to specify axioms about the wumpus world; for example:

  “Perception to knowledge”
  \[ \forall b, g, t, m, c \ Percept([Smell, b, g, m, c], t) \Rightarrow \text{Smelt}(t) \]
  \[ \forall s, b, t, m, c \ Percept([s, b, \text{Glitter}, m, c], t) \Rightarrow \text{AtGold}(t) \]

  Aside: Must keep track of time, and so \text{Smelt}(t).

• Reflex action: \( \forall t \ \text{AtGold}(t) \Rightarrow \text{Action}(\text{Grab}, t) \)

• Reflex action with internal state:
  Do we have the gold already?
  \[ \forall t \ \text{AtGold}(t) \land \neg \text{Holding}(\text{Gold}, t) \Rightarrow \text{Action}(\text{Grab}, t) \]

• Note that \( \text{Holding}(\text{Gold}, t) \) cannot be observed
  \[ \text{must keep track of change} \]
Knowledge in the Wumpus World

• Need to specify axioms about the wumpus world; for example:
  
  “Perception to knowledge”
  \[ \forall b, g, t, m, c \text{ Percept}([\text{Smell}, b, g, m, c], t) \Rightarrow \text{Smelt}(t) \]
  \[ \forall s, b, t, m, c \text{ Percept}([s, b, \text{Glitter}, m, c], t) \Rightarrow \text{AtGold}(t) \]

  
  Aside: Must keep track of time, and so \text{Smelt}(t).

• Reflex action: \[ \forall t \text{ AtGold}(t) \Rightarrow \text{Action(Grab, t)} \]

• Reflex action with internal state:
  Do we have the gold already?
  \[ \forall t \text{ AtGold}(t) \land \lnot \text{Holding(Gold, t)} \Rightarrow \text{Action(Grab, t)} \]

• Note that \text{Holding(Gold, t)} cannot be observed
  
  must keep track of change

• Q: If we know \text{Holding(Gold, t)} can we conclude \text{Holding(Gold, t + 1)}?
Knowledge in the Wumpus World

• Need to specify axioms about the wumpus world; for example:

“Perception to knowledge”

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

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

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

• Reflex action: \( \forall t \text{ AtGold}(t) \Rightarrow \text{Action(Grab, t)} \)

• Reflex action with internal state:
  Do we have the gold already?

\[
\forall t \text{ AtGold}(t) \land \neg \text{Holding(Gold, t)} \Rightarrow \text{Action(Grab, t)}
\]

• 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:
  \[ \forall x, t \ At(Agent, x, t) \land Smelt(t) \Rightarrow Smelly(x) \]
  \[ \forall x, t \ At(Agent, x, t) \land Breeze(t) \Rightarrow Breezy(x) \]

- Need to be careful that *all* information is represented.
  Consider “Squares are breezy near a pit”:
Representing Information

• Need to remember properties of locations:
  \( \forall x, t \; At(Agent, x, t) \land Smelt(t) \Rightarrow Smelly(x) \)
  \( \forall x, t \; At(Agent, x, t) \land Breeze(t) \Rightarrow Breezy(x) \)

• Need to be careful that all information is represented. Consider “Squares are breezy near a pit”:
  
  • Diagnostic rule – infer cause from effect
    \( \forall y \; Breezy(y) \Rightarrow \exists x Pit(x) \land Adjacent(x, y) \)
  
  • Causal rule – infer effect from cause
    \( \forall x, y \; Pit(x) \land Adjacent(x, y) \Rightarrow Breezy(y) \)
Representing Information

• Need to remember properties of locations:
  \[ \forall x, t \text{ At}(\text{Agent}, x, t) \land \text{Smelt}(t) \Rightarrow \text{Smelly}(x) \]
  \[ \forall x, t \text{ At}(\text{Agent}, x, t) \land \text{Breeze}(t) \Rightarrow \text{Breezy}(x) \]

• Need to be careful that all information is represented. Consider “Squares are breezy near a pit”:
  - Diagnostic rule – infer cause from effect
    \[ \forall y \text{ Breezy}(y) \Rightarrow \exists x \text{Pit}(x) \land \text{Adjacent}(x, y) \]
  - Causal rule – infer effect from cause
    \[ \forall x, y \text{ Pit}(x) \land \text{Adjacent}(x, y) \Rightarrow \text{Breezy}(y) \]

• 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:
  \[ \forall y \text{ Breezy}(y) \equiv [\exists x \text{Pit}(x) \land \text{Adjacent}(x, y)] \]
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.

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

Full Adder

1

2

3
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 \( (X_1) = \text{XOR} \)
  • Binary predicate:
    Type \( (X_1, \text{XOR}) \)
  • Unary predicate:
    \( \text{XOR}(X_1) \)
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: \( \text{Type}(X_1) = \text{XOR} \)
     - Binary predicate: \( \text{Type}(X_1, \text{XOR}) \)
     - Unary predicate: \( \text{XOR}(X_1) \)
The Electronic Circuits Domain

4. Encode general knowledge of the domain:
The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \, \text{Connected}(p_1, p_2) \Rightarrow \text{Signal}(p_1) = \text{Signal}(p_2)$
The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- \( \forall p_1, p_2 \, Connected(p_1, p_2) \Rightarrow Signal(p_1) = Signal(p_2) \)
- \( \forall p \, Signal(p) = 1 \lor Signal(p) = 0 \)
4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \text{ Connected}(p_1, p_2) \Rightarrow \text{Signal}(p_1) = \text{Signal}(p_2)$
- $\forall p \text{ Signal}(p) = 1 \lor \text{Signal}(p) = 0$
- $1 \neq 0$
The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \ \text{Connected}(p_1, p_2) \Rightarrow \text{Signal}(p_1) = \text{Signal}(p_2)$
- $\forall p \ \text{Signal}(p) = 1 \lor \text{Signal}(p) = 0$
- $1 \neq 0$
- $\forall p_1, p_2 \ \text{Connected}(p_1, p_2) \Rightarrow \text{Connected}(p_2, p_1)$
The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- $\forall p_1, p_2 \text{ Connected}(p_1, p_2) \Rightarrow \text{Signal}(p_1) = \text{Signal}(p_2)$
- $\forall p \text{ Signal}(p) = 1 \lor \text{Signal}(p) = 0$
- $1 \neq 0$
- $\forall p_1, p_2 \text{ Connected}(p_1, p_2) \Rightarrow \text{Connected}(p_2, p_1)$
- $\forall g \text{ Type}(g) = \text{OR} \Rightarrow$
  \[ \text{Signal}(\text{Out}(1, g)) = 1 \equiv \exists n \text{ Signal}(\text{In}(n, g)) = 1 \]
The Electronic Circuits Domain

4. Encode general knowledge of the domain:

- \( \forall p_1, p_2 \) \( \text{Connected}(p_1, p_2) \Rightarrow \text{Signal}(p_1) = \text{Signal}(p_2) \)
- \( \forall p \) \( \text{Signal}(p) = 1 \lor \text{Signal}(p) = 0 \)
- \( 1 \neq 0 \)
- \( \forall p_1, p_2 \) \( \text{Connected}(p_1, p_2) \Rightarrow \text{Connected}(p_2, p_1) \)
- \( \forall g \) \( \text{Type}(g) = \text{OR} \Rightarrow \)
  \( \text{Signal}(\text{Out}(1, g)) = 1 \equiv \exists n \text{Signal}(\text{In}(n, g)) = 1 \)
- \( \forall g \) \( \text{Type}(g) = \text{AND} \Rightarrow \)
  \( \text{Signal}(\text{Out}(1, g)) = 0 \equiv \exists n \text{Signal}(\text{In}(n, g)) = 0 \)
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\((p_1, p_2) \Rightarrow \text{Signal}(p_1) = \text{Signal}(p_2) \)
- \( \forall p \) Signal\((p) = 1 \lor \text{Signal}(p) = 0 \)
- \( 1 \neq 0 \)
- \( \forall p_1, p_2 \) Connected\((p_1, p_2) \Rightarrow \text{Connected}(p_2, p_1) \)
- \( \forall g \) Type\((g) = \text{OR} \Rightarrow \)
  \[ \text{Signal}(\text{Out}(1, g)) = 1 \equiv \exists n \text{Signal}(\text{In}(n, g)) = 1 \]
- \( \forall g \) Type\((g) = \text{AND} \Rightarrow \)
  \[ \text{Signal}(\text{Out}(1, g)) = 0 \equiv \exists n \text{Signal}(\text{In}(n, g)) = 0 \]
- \( \forall g \) Type\((g) = \text{XOR} \Rightarrow \)
  \[ \text{Signal}(\text{Out}(1, g)) = 1 \equiv \text{Signal}(\text{In}(1, g)) \neq \text{Signal}(\text{In}(2, g)) \]
- \( \forall g \) Type\((g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1, g)) \neq \text{Signal}(\text{In}(1, g)) \)
The Electronic Circuits Domain

5. Encode the specific problem instance:

\[\begin{align*}
\text{Type}(X_1) &= \text{XOR} & \text{Type}(X_2) &= \text{XOR} \\
\text{Type}(A_1) &= \text{AND} & \text{Type}(A_2) &= \text{AND} \\
\text{Type}(O_1) &= \text{OR} \\
\end{align*}\]

\[\begin{align*}
\text{Connected}(\text{Out}(1, X_1), \text{In}(1, X_2)) & & \text{Connected}(\text{In}(1, C_1), \text{In}(1, X_1)) \\
\text{Connected}(\text{Out}(1, X_1), \text{In}(2, A_2)) & & \text{Connected}(\text{In}(1, C_1), \text{In}(1, A_1)) \\
\text{Connected}(\text{Out}(1, A_2), \text{In}(1, O_1)) & & \text{Connected}(\text{In}(2, C_1), \text{In}(2, X_1)) \\
\text{Connected}(\text{Out}(1, A_1), \text{In}(2, O_1)) & & \text{Connected}(\text{In}(2, C_1), \text{In}(2, A_1)) \\
\text{Connected}(\text{Out}(1, X_2), \text{Out}(1, C_1)) & & \text{Connected}(\text{In}(3, C_1), \text{In}(2, X_2)) \\
\text{Connected}(\text{Out}(1, O_1), \text{Out}(2, C_1)) & & \text{Connected}(\text{In}(3, C_1), \text{In}(1, A_2)) \\
\end{align*}\]
6. Pose queries to the inference procedure

- E.g. what are the outputs, given a set of input signals?
- I.e.
  \[\exists o_1, o_2 \quad \left( \text{Signal}(\text{In}(1, C_1)) = 1 \land \text{Signal}(\text{In}(2, C_1)) = 0 \land \text{Signal}(\text{In}(3, C_1)) = 1 \right) \Rightarrow \left( \text{Signal}(\text{Out}(1, C_1)) = o_1 \land \text{Signal}(\text{Out}(2, C_1)) = o_2 \right)\]
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.
     \[ \exists o_1, o_2 \]
     \[ (\text{Signal}(\text{In}(1, C_1)) = 1 \land \text{Signal}(\text{In}(2, C_1)) = 0 \land \]
     \[ \text{Signal}(\text{In}(3, C_1)) = 1) \]
     \[ \Rightarrow \]
     \[ (\text{Signal}(\text{Out}(1, C_1)) = o_1 \land \text{Signal}(\text{Out}(2, C_1)) = o_2) \]

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