

Introducing relations

Lecture 9 First Order Logic

We have seen that Propositional Logic is in general too tedious to use, and it still can't express many things we want to express.

In this chapter, we will modify the KRL a little bit so that it can express many more things in a more succinct way.

Reference:

- Textbook chapter 7, section 9.1–9.2

AI(0270)

AI(0270)-9.1

Predicates

We add “**predicate**” into our set of **possible sentences**:

$$\text{Predicate}(Obj_1, Obj_2, \dots, Obj_n)$$

where the Obj_i are “objects”.

Being sentences, they have a truth value of either true or false.

For example, suppose we use object symbols like *Room-1-1*, etc. To express that room 1, 2 is the North neighbour of room 1, 1, we can write

$$\text{NorthNeighbour}(\text{Room-1-2}, \text{Room-1-1})$$

You can write it the reverse way: it is you who interpret it. But be consistent.

The **interpretation** of this sentence is **up to the world**: the language doesn't say anything about it, except that it must always be true or false.

Each predicate needs a **fixed number of objects**.

It is called a **property** if it needs 1, and a **relation** if it needs more. If it needs 0, it is a constant predicate—which is used to replace propositions.

AI(0270)-9.2

AI(0270)-9.3

Functions

Why **propositional** logic is **not expressive enough**?

- Our language consists **only of propositional symbol**...
So what? Every sentence is either true or false!
- and these propositional symbols are all **completely unrelated**.
This is a big problem: if we want to say “for any room, if the room has no wumpus and the room has no pit, then the room is okay”, then the sentences “the room has wumpus”, “the room has pit” and “the room is okay” must relate in some way for each room.
- So the first thing to do is to add a new type of sentences: **relation among objects**. The interpretation is **not hardcoded into the language**.
It is crucial that the interpretation is not hardcoded in the language. We want a more systematic way to add things that we can interpret ourselves, rather than adding a new unrelated symbol every time.
- So **objects** makes **atomic sentences relate with others**.

- Some relations have the property that **for any given object, exactly one object is related to it**. E.g., Father, Mother, Negation, etc.
- For these objects, it makes sense to **indirectly** name the object. E.g., the father of Pat, the Negation of 10, etc.
- We write it by using a “**function**”: *MotherOf(Pat)*, *NegationOf(10)*, etc.
- Note that **there is no “procedure”** telling us who is the Mother of Pat. We just use it so that we can reason **without knowing what object** it refers to.
- Since we have **no automatic way** to know what is the “function value”, we need a way to **specify** that ourselves.
E.g., we might know that Pat's mother is June, and we need to be able to tell this to our reasoning system.
- We do it with a special relation called **equality**.

The special relation: equality

- There is a special binary relation called “=” (**equality**).
- It's **interpretation is fixed** by the language: it is true if the two objects are the **same object**.
- Different names do not imply inequality!**
- For convenience, we write it in a special syntax: e.g.,

$$\begin{aligned} \text{Room-1-1} &= \text{Room-1-1} \\ \text{MotherOf}(\text{Pat}) &= \text{June} \end{aligned}$$

- Clearly, **sometimes we need to tell the reasoning system before the system knows about equality**: e.g., the latter relation must be told, or the reasoning system will not know who is the mother of Pat.
- But once equality of two objects is established, **everything that apply to the one object will also apply to the other object**.

AI(0270)-9.4

Universal Quantifier

- Why we would like to **replace propositional symbols** like *OK-1-1* by **predicates** like *OK(Room-1-1)*?
- Because we can then easily express things like “for anything for which the Wumpus and Pit properties both hold, the OK property holds”.
- But for this to be possible, we need two things: a **sentence** which express “**for anything**”, and a **term** which express the “**anything**”.
- How to write it? We should be familiar with this already:
$$\forall x \text{ Room}(x) \Rightarrow \text{Wumpus}(x) \vee \text{Pit}(x) \vee \text{OK}(x)$$
- But there is one big restrictions: a variable like x can only represent an **object**, not a **predicate**.
- We call this language **First Order Logic** because of this restriction.
Object is consider to be the “bottom” building blocks, then the truth deriving from objects (predicates), then the truth of things derived from predicates, ...

AI(0270)-9.5

Meaning of universal quantifier

You can understand the **universal quantifier** like a **HUGE conjunction**. E.g., the above universal quantifier means:

$$\begin{aligned}
&(Room(Room-1-1) \Rightarrow \\
&\quad Wumpus(Room-1-1) \vee Pit(Room-1-1) \vee OK(Room-1-1)) \\
&\wedge (Room(Room-1-2) \Rightarrow \\
&\quad Wumpus(Room-1-2) \vee Pit(Room-1-2) \vee OK(Room-1-2)) \\
&\wedge (Room(Agent) \Rightarrow Wumpus(Agent) \vee Pit(Agent) \vee OK(Agent)) \\
&\wedge (Room(Isaac) \Rightarrow Wumpus(Isaac) \vee Pit(Isaac) \vee OK(Isaac)) \\
&\wedge \dots
\end{aligned}$$

Note that **implication is a very natural connective** to use to construct a universally quantified sentence.

If we use the **"and"** instead, we will get a statement **much too strong**.

Because we have a conjunct " $Room(x) \dots$ ", so it says everything is a room (as well as other things)—probably not what we really mean.

AI(0270)-9.6

Existential Quantifier

We also have the **existential quantifier**, meaning **"for some"**. E.g., there is some room which contains a wumpus:

$$\exists x Room(x) \wedge Wumpus(x)$$

This can be interpreted like a **big disjunction**:

$$\begin{aligned}
&(Room(Room-1-1) \wedge Wumpus(Room-1-1)) \\
&\vee (Room(Room-1-2) \wedge Wumpus(Room-1-2)) \\
&\vee (Room(Agent) \wedge Wumpus(Agent)) \\
&\vee (Room(Isaac) \wedge Wumpus(Isaac)) \vee \dots
\end{aligned}$$

Note that **conjunction is a very natural connective** to use to construct an existentially quantified sentence.

If **implication or disjunction** is used, the statement is **too weak**.

Because we have a disjunct " $Room(x) \dots$ ", making the statement true as long as there is any room (and in some other situations)—probably not what we mean.

AI(0270)-9.7

Syntax of FOL

Now we have everything in place. Let's summarize:

- A **sentence** can be either **simple** or **complex**.
- **Simple sentences** are always **predicates**, or the predefined **equality** relation.
- **Complex sentences** can be formed by using our logic **connectives** (and, or, not, implies, equivalent to) on **sentences**.
- **Complex sentences** can also be formed by introducing a variable and **quantify a sentence with the variable** universally or existentially.
- A **predicate** is composed of a predicate symbol and some **terms**.
- A **term** can be an **object symbol** or a **variable symbol** introduced previously. It can also be a **function** operating on some **term**.
So we can have $FatherOf(MotherOf(June))$, $MotherOf(x)$, etc.

AI(0270)-9.8

Particulars on the syntax

- The precedence: same as propositional logic, except that **equality** has very high precedence, while **quantifiers** have very low precedence.
- What happens if the same variable is used in **more than one quantifiers**? E.g., what is meant by the following?

$$\forall x P(x) \Rightarrow \exists x Q(x)$$

- **Our convention**: the variable will be binded to the **closest enclosing quantifier**. So the above means

$$\forall x (P(x) \Rightarrow (\exists y Q(y)))$$

- But since we **can always rename variables**, we will avoid having to use this rule.
In other words, we will try not to reuse variables in the same sentence.

AI(0270)-9.9

Higher order logic?

There is one question: **why stop at first-order logic?**

- **Higher-order logic** allows **variables to represent** more complicated things, like **predicates**, another **sentence**, etc.
- And there are things that can't be represented without such operators.
E.g., "Everything that relates to me will relates to you in the same way":
 $\forall x x(Me) \Rightarrow x(You)$
- But to be able to reason, we must restrict ourselves to something less complex.
- Currently, **very few is known** about how to reason in higher-order logic.

AI(0270)-9.10

Example: representing the wumpus world—Part 1

Now let's see some example about how to use our new language.

- Room-1-1, ..., Room-4-4 are the only rooms.
 $Room(x) \Rightarrow x = Room-1-1 \vee \dots \vee x = Room-4-4$
- Room-1-1, ..., Room-4-4 are all distinct rooms.
 $Room-1-1 \neq Room-1-2, \dots, Room-4-3 \neq Room-4-4$
- Room-1-2 is the north-neighbour of Room-1-1,
 $NorthNeighbour(Room-1-2, Room-1-1), \dots$
- x is north-neighbour of y if and only if y is south-neighbour of x .
 $\forall x \forall y NorthNeighbour(x, y) \Leftrightarrow SouthNeighbour(y, x)$
- There is exactly one room containing a wumpus.
 $\exists x Room(x) \wedge Wumpus(x)$
 $\forall x \forall y (Wumpus(x) \wedge Wumpus(y)) \Rightarrow x = y$

AI(0270)-9.11

Some more rules in the wumpus world

- For each room, there is at most one North neighbouring room.
 $\forall x \forall y \forall z (NorthNeighbour(y, x) \wedge NorthNeighbour(z, x)) \Rightarrow y = z$
- Neighbouring rooms is one of north, south, east or west neighbours.
 $\forall x \forall y Neighbour(x, y) \Leftrightarrow (NorthNeighbour(x, y) \vee SouthNeighbour(x, y) \vee EastNeighbour(x, y) \vee WestNeighbour(x, y))$
- If a room has a pit, then all neighbouring rooms is breezy.
 $\forall x \forall y Pit(x) \wedge Neighbour(x, y) \Rightarrow Breezy(y)$
 What this implies on non-breezy rooms?
- If a room is breezy, then some neighbouring room has a pit.
 $\forall x Breezy(x) \Rightarrow \exists y Neighbour(x, y) \wedge Pit(y)$

All these are true in any cave, and should be considered background knowledge. We say that these are **axioms** of the Wumpus world.

AI(0270)-9.12

What happens when quantifiers nests?

- When the **same quantifiers nest**, the **meaning does not change** if we change the order. E.g., the following sentences are the same.
 $\forall x \forall y Student(x) \wedge Exam(y) \Rightarrow \neg Like(x, y)$
 $\forall y \forall x Student(x) \wedge Exam(y) \Rightarrow \neg Like(x, y)$
- Most of the time we just write it like:
 $\forall x, y Student(x) \wedge Exam(y) \Rightarrow \neg Like(x, y)$
- This is true **also for existential quantifiers**.
- But what if **different types of quantifiers nest**? Let's see:
 $\exists x \forall y Student(x) \wedge (Exam(y) \Rightarrow Like(x, y))$
 $\forall y \exists x Student(x) \wedge (Exam(y) \Rightarrow Like(x, y))$
- They **mean completely different** things! (Some student likes all exams; each exam has some student who like it.)
 Which is a stronger statement?

AI(0270)-9.13

The relationship between quantifiers

De Morgan's rule says that $(\neg x \vee \neg y)$ and $\neg(x \wedge y)$ are the same, while $(\neg x \wedge \neg y)$ and $\neg(x \vee y)$ are the same.

Because \forall is a big conjunction, and \exists is a big disjunction, it is natural that **de Morgan's rule also holds**. The following are equivalent:

$$\begin{array}{ll} \exists x \neg P & \text{and} & \neg \forall x P \\ \forall x \neg P & \text{and} & \neg \exists x P \end{array}$$

The most important use of this rule is to **move negation into the "body" of quantifications**. The following sentences are all equivalent:

$$\begin{array}{l} \neg \forall x P(x) \Rightarrow \exists y Q(x, y) \\ \exists x \neg(P(x) \Rightarrow \exists y Q(x, y)) \\ \exists x \neg(\neg P(x) \vee \exists y Q(x, y)) \\ \exists x P(x) \wedge \neg \exists y Q(x, y) \\ \exists x P(x) \wedge \forall y \neg Q(x, y) \\ \exists x \forall y P(x) \wedge \neg Q(x, y) \end{array}$$

AI(0270)-9.14

Special notations for arithmetics

- We allow the notations like $x + 1$ to represent the sum of x and 1 .
- This should be considered as **syntactic sugar**: we allow a function like $+(x, 1)$ to be written like $x + 1$ instead for easy reading.
- But that **doesn't mean our reasoning program automatically knows** how to deal with **arithmetics**. To it, "1", "2", etc., are just objects like "Room-1-1", "Agent" or "Isaac"; and "+" is just a functional symbol like "MotherOf".
- Unless you tell it, the reasoning system won't know even $1 + 1 = 2$.
- General arithmetics is **out of the capability of predicate calculus**.
- **With capability of arithmetics, complete reasoning is impossible**.
 This is proved by Gödel, in the Gödel's incompleteness theorem. Essentially, it says that for any system that is expressive enough to express logic and Mathematical Induction, and for any finite set of consistent axioms, there are true sentences that cannot be proved using just those axioms.

AI(0270)-9.15

Special syntax for sets and lists

Again for convenience, we use the following syntax for sets:

- \emptyset : Object: Empty set
- $\{x\}$: Object: Set containing just x
- $\{x, y\}$: Object: Set containing just x and y
- $\{x, y\}s$: Function: Set containing x, y and elements in set s
- $r \cup s$: Function: Union of set r and set s
- $r \cap s$: Function: Intersection of set r and set s
- $x \in s$: Predicate: x is an element of set s
- $r \subseteq s$: Predicate: Set r is a (perhaps improper) subset of set s

We also use the following for lists:

- $[]$: Object: Empty list
- $[x]$: Object: List containing just x
- $[x, y]$: Object: List containing just x and y , in that order
- $[x, y|l]$: Function: Set containing x, y and elements in list l , in that order

AI(0270)-9.16

Queries with existential quantifier

Suppose we ask a question that is **existentially quantified**, e.g.,

$$\exists x Child(x, Alice)$$

If the answer is affirmative, we want a **proof** telling us **some objects** x that satisfies the statement, by giving us a **set of substitutions**, e.g.,

$$\{ \{x/Bob\}, \{x/Cindy\} \}$$

Set of set, since there can be multiple substitutions, each for multiple variables.

So in effect the query is a question like "do you know any child of Alice", and the answer is something like "Yes, they are Bob and Cindy."

What if the reasoning program doesn't know the values? E.g., what will happen if we tell KB that Room-1-2 is breezy, and then ask "do you know any neighbouring room of Room-1-2 that contains a pit?"

Answer: it returns a binding containing a brand-new constant symbol.

AI(0270)-9.17

Situation calculus: Representing changes

- The world becomes more complicated if things can change. E.g., if we want to ask the KB about questions like **where is the agent now?**
- Predicate logic is **monotonic**, i.e., true things will never become false, and false things will never become true.
- Knowledge doesn't really "change". Instead, **new knowledge comes**.
- Example: we can say that the agent is at room 1, 1 at time 0. I.e.,
 $At(Agent, Room-1-1, S_0)$
- Whenever the agent **decides** to make a move, it **adds the decision to KB**. E.g., if at time 0, the agent decides to move north:
 $Result(Move(North), S_0) = S_1$
- S_1 is called a **situation object**, which can be viewed as a **fixed snapshot** of the world which never change.

AI(0270)-9.18

Situation Calculus: Cont'd

- Now the agent **must be able to tell** what is in the new situation S_1 using **extra axioms** of the Wumpus world. E.g.:

If the agent moves north in a situation, it is at the north neighbouring room in the next situation.

$$\forall r, r', s, s' (At(Agent, r, s) \wedge Result(Move(North), s) = s' \wedge NorthNeighbour(r', r)) \Rightarrow At(Agent, r', s')$$

- We **need axioms even for things that doesn't change**. E.g., "if the agent is moving, the "holding gold" status is not changed".
 $\forall x, s, s' Result(Move(x), s) = s' \Rightarrow (HoldingGold(s) \Leftrightarrow HoldingGold(s'))$
- Other aspects of the world, like measurements, events, concurrency, etc., can be captured in similar ways. See Ch.8 for details.

AI(0270)-9.19

Inference: truth-table no longer works!

So much about how to write in FOL. Now comes the difficult part: **how to do inference in predicate logic?**

- **The truth-table approach won't work**: we can't determine truth of a universally or existentially quantified sentence using truth-table.
- **Rule-based inference becomes our hope for an inference procedure**: by repeatedly applying inference rules, we want to be able to tell whether a KB entails a statement.
- So **is there a set of inference rules** that are good?
- And, if so, **how to use those rules?**
- It turns out that **there is a set of inference rule** which is **sound** and **complete**, although it is not very efficient.

Again proved by Gödel, in the Gödel's **completeness** theorem. He didn't tell what is the inference procedure, but we will show it: "Resolution".

AI(0270)-9.20

New inference rules: some new terms

- The inference rules of propositional logic can still be used. But what **inference rules** can be used for **quantifiers**?
- It is to be expected that our rules will involve **manipulating the variables** of the sentence. But we have no syntax to do so yet!
- Or rather, we have, but didn't formalize it. Recall that we can have **substitution** like
 $\{x/Cindy, y/Bob\}$
- This can be seen as renaming x to Cindy, y to Bob. We allow renaming from variables to variables as well.
- We use the notation $SUBST$ to represent a substitution. E.g.:

$$SUBST(\{x/Cindy\}, Child(x, Alice))$$

means

$$Child(Cindy, Alice)$$

AI(0270)-9.21

Universal Elimination and Existential Introduction

The trivial inference rules first:

- **Universal Elimination**: for any variable v , sentence α , and variable-free term g , we have

$$\frac{\forall v \alpha}{SUBST(\{v/g\}, \alpha)}$$

I.e., if we know "for all v (some sentence)", then (some sentence) is true if we replace v by any variable-free term g .

We call a variable-free term a "**ground term**".

- **Existential Introduction**: for any sentence α , variable v that does not appear in α , and ground term g , we have

$$\frac{\alpha}{\exists v SUBST(\{g/v\}, \alpha)}$$

I.e., if we know (some sentence) is true, then we can introduce a new variable in place of a term, and use an existential quantifier to quantify it.

AI(0270)-9.22

Existential Elimination

Now the (somewhat) non-trivial rule:

- **Existential elimination**: for any sentence α , variable v , and constant symbol k not appearing anywhere else in KB:

$$\frac{\exists v \alpha}{SUBST(\{v/k\}, \alpha)}$$

That is, if there exists a sentence that is true for some v , then we can invent a new term k to replace v .

This might seem strange, but it is in fact trivial as well: **the new symbol looks like a variable** anyway. We have **no knowledge**, and thus **no commitment**, about it—until somebody tell us $k = t$ for some concrete term t .

You probably would nearly yell out "cheat!!!". It is not really a way to infer new knowledge. Instead, it shows the equivalence between a new symbol and a variable used in an existential quantifier.

AI(0270)-9.23

Example inference in Predicate Calculus

Suppose we have the following:

It is a crime for an American to sell weapons to hostile nations of US. The country Nono is hostile to US, and has some missiles. All its missiles come from Colonel West, who is an American.

What we want: **West is a criminal**

What we should do?

- First, **convert** the **knowledge**, and all background knowledge, in First Order Logic.
- Then, **convert** the **question** to First Order Logic.
- Now, try to use inference rules to **prove** it.

AI(0270)-9.24

The conversion

Knowledge

$$1. \forall x, y, z (American(x) \wedge Weapon(y) \wedge Nation(z) \wedge Hostile(z) \wedge Sells(x, z, y)) \Rightarrow Criminal(x)$$

$$2. Hostile(Nono)$$

$$3. Nation(Nono)$$

$$4. \exists x Missile(x) \wedge Own(Nono, x)$$

$$5. \forall x Missile(x) \wedge Own(Nono, x) \Rightarrow Sells(West, Nono, x)$$

$$6. American(West)$$

$$7. Missile(x) \Rightarrow Weapon(x)$$

This is background knowledge. We must be careful not to assume that the reasoning system knows any "common-sense".

Query

$$8. Criminal(West)$$

AI(0270)-9.25

Proof

Proof process

Existential elimination to (4):

$$9. Missile(M1) \wedge Own(Nono, M1)$$

And-elimination of (9):

$$10. Missile(M1)$$

$$11. Own(Nono, M1)$$

Universal elimination to (7)

$$12. Missile(M1) \Rightarrow Weapon(M1)$$

Modus Ponens on (12) using (10)

$$13. Weapon(M1)$$

Universal Elimination on (5) using $\{x/M1\}$

$$14. Missile(M1) \wedge Own(Nono, M1) \Rightarrow Sells(West, Nono, M1)$$

AI(0270)-9.26

Proof (cont'd)

Modus Ponens on (14), using (9):

$$15. Sells(West, Nono, M1)$$

Universal Elimination on (1) three times:

$$16. (American(West) \wedge Weapon(M1) \wedge Nation(Nono) \wedge Hostile(Nono) \wedge Sells(West, Nono, M1)) \Rightarrow Criminal(West)$$

And-Introduction to (6), (10), (3), (2), (15):

$$17. American(West) \wedge Weapon(M1) \wedge Nation(Nono) \wedge Hostile(Nono) \wedge Sells(West, Nono, M1)$$

Modus Ponens on (16) using (17):

$$18. Criminal(West)$$

Goal found.

AI(0270)-9.27

Problems in the inference procedure

How fast is it?

- The proof is 12 steps long, so search depth is 12.
- What is the branching factor? Well... it becomes larger and larger as the proof goes...
Because there are more and more possibilities to do things like And-introduction and quantifier eliminations.
- And is easily something like 10000.
Why this large? Consider how many different and-introduction you can do!

It should be clear that the process is **not efficient** enough for automatic reasoning. Problem:

- Applications of rules are complete **guess works**. We have taken no step in making sure that only "useful things" come out.

Is there a **better way**?!

AI(0270)-9.28