Chapter 8: First-Order Logic

8.1 Representation Revisited

1. Representation Languages : Programming languages like Java or Python are commonly used to represent facts using data structures. For example, an array can represent the contents of a wumpus world.

2. Limitation of Programming Languages : While programming languages are effective for representing facts, they lack a general mechanism to derive new facts from existing ones. Each update to data structures typically requires domain-specific procedures.

3. Declarative vs. Procedural: Declarative languages like propositional logic separate knowledge and inference, making inference domain-independent. In contrast, procedural approaches in programming languages tie knowledge and inference closely together.

4. Expressiveness: While propositional logic can handle partial information using disjunction and negation, it lacks expressiveness for concisely describing environments with many objects.

5. Compositionality: Propositional logic exhibits compositionality, meaning the meaning of a sentence depends on the meanings of its parts. This property ensures consistency and coherence in representing knowledge.

6. Limitations of Propositional Logic: Although useful, propositional logic struggles to concisely describe complex environments, requiring separate rules for each element, unlike natural languages or structured representation languages like first-order logic.

• Skip 8.1.1, 8.1.2

8.2, 8.3 Syntax and Semantics, Using FOL

Models for First- Order Logic

Models are structures describing possible worlds:

- Link Vocabs of sentences to elements of a possible world
    - Provides for finding truth of any sentence
- Propositional logic link propositional symbols to predefined truth values.

FIRST – ORDER LOGIC Model have:

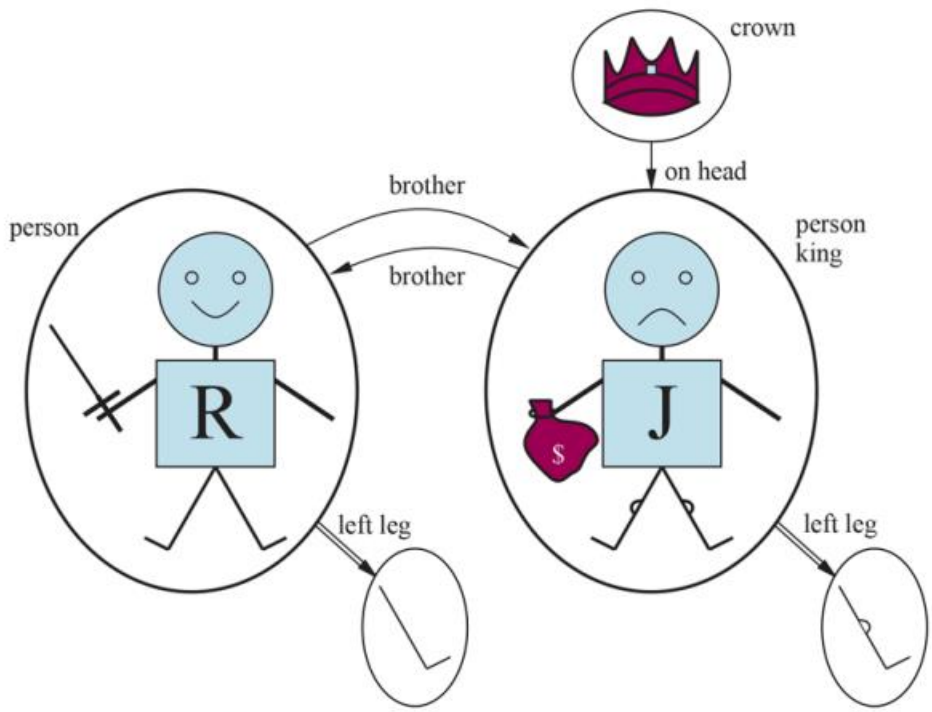
- Objects
    - the domain of the model are its set of objects
        - Must be non empty

- The set of objects are aka **domain elements.**

Consider:

- Richard the LoinHeart, king of England

  1189 → 1199
- His younger brother, evil King john who ruled 1199 → 1215
- The left legs of Richard and john ?
- A crown



A model containing five objects, two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).

- Objects can be related in many ways

- A relation is a set of tuples.
- John and Richard are brothers:

  <Richard, John>,<John, Richard>
- A crown on johns head

  <Crown, John>

Both are binary relations.

UNARY RELATIONS:

Person true for john and Richard

King true for john only(Once Richard is dead)

Some Relations are functions

Each person has one left leg, so the model includes a unary "left leg", so the model includes a unary ":Left leg" function with maps:

<Richard> → Richard's Left leg

<John > → John's left leg

First order logic model requires total functions.

- A value exist for every input tuple
- Weird : the crown must have a left leg and so all the left legs.

SYMBOLS AND INTERPRETATIOS

- Syntactic elements represent objects, relations, functions.

- Three Kinds: Constant Symbols for objects,
  Predicate Symbols for relations
  Function Symbol for functions
- Each predicate and function symbol has an arity, fixing the number of arguments.

Models must provide information to determine truth.

- Needs interpretation. Specify objects, relations and functions etc.

A possible Interpretation of king example :

- Richard refers to Richard the lionheart, john refers to evil King john
- Brother refers to the brotherhood relations.etc

Many interpretations

- 5 constant objects so 25 possible interpretations

TERMS:

A logical expression referring to an object.

- Constant symbols are these.
- don't always need a constant symbol to name every object
  Example: Leftleg(John),
  "Mother(John)", which refers to John's mother.
- Uses a function symbol followed by arguments

ATOMIC SENTENCES:

Basic statements of facts and relationships

Example: "Richard is the brother of King John" is an atomic sentence represented as Brother(Richard, John).

COMPLEX SENTENCES:

Logical connectives can help build more complicated stuff.

¬Brother(LeftLeg(Richard),John)

Brother(Richard,John) ∧ Brother(John,Richard)

King(Richard) ∨ King(Richard)

¬King(Richard) ⇒ King(John).

QUANTIFIERS

Quantifiers: Quantifiers like "For all" ($\forall$) and "There exists" ($\exists$) are used to make statements about collections of objects.

Example: "For all kings, they are persons" can be represented as $\forall x \ \mathrm{King}(x) \Rightarrow \mathrm{Person}(x)$.

Existential quantification (∃)

Universal quantification makes statements about every object. Similarly, we can make a statement about some object without naming it, by using an existential quantifier. To say, for example, that King John has a crown on his head, we write

∃x Crown(x) ∧ OnHead(x,John).

1. **Consecutive Quantifiers**:
    - Example 1: "For every person, there is a friend." This can be represented as $\forall x \exists y \mathrm{Friend}(x, y)$, where x represents every person, and y represents a friend of that person.
    - Example 2: "Everyone loves someone." This can be represented as $\forall x \exists y \mathrm{Loves}(x, y)$, where x represents every person, and y represents someone they love.
2. **Mixed Quantifiers**:
    - Example 1: "There exists a student who passed all exams." This can be represented as $\exists x \forall y \mathrm{Passed}(x, y)$, where x represents a student, and y represents all exams.
    - Example 2: "Every team has at least one captain." This can be represented as $\forall x \exists y \mathrm{Captain}(x, y)$, where x represents every team, and y represents at least one captain of that team.

USING FOL:

Domains are some part of the world we want to express some knowledge.

Assertions and queries in FOL

Add sentence to a knowledge base using TELL.

Called ASSERTIONS

Assertions John is a king, Richard is a person, all kings are persons.

TELL(KB,King(John))

TELL(KB, Person(Richard))

ASK Questions of KB using ASK:

ASK(KB,King(John))

ASK(KB, ∃ x Person(x)).

SUBSITUTIONS

ASKVARS(KB, Person(x))

Gives: a “Stream” of answers:

{x/John},{x/Richard} – Substitution or binding list.

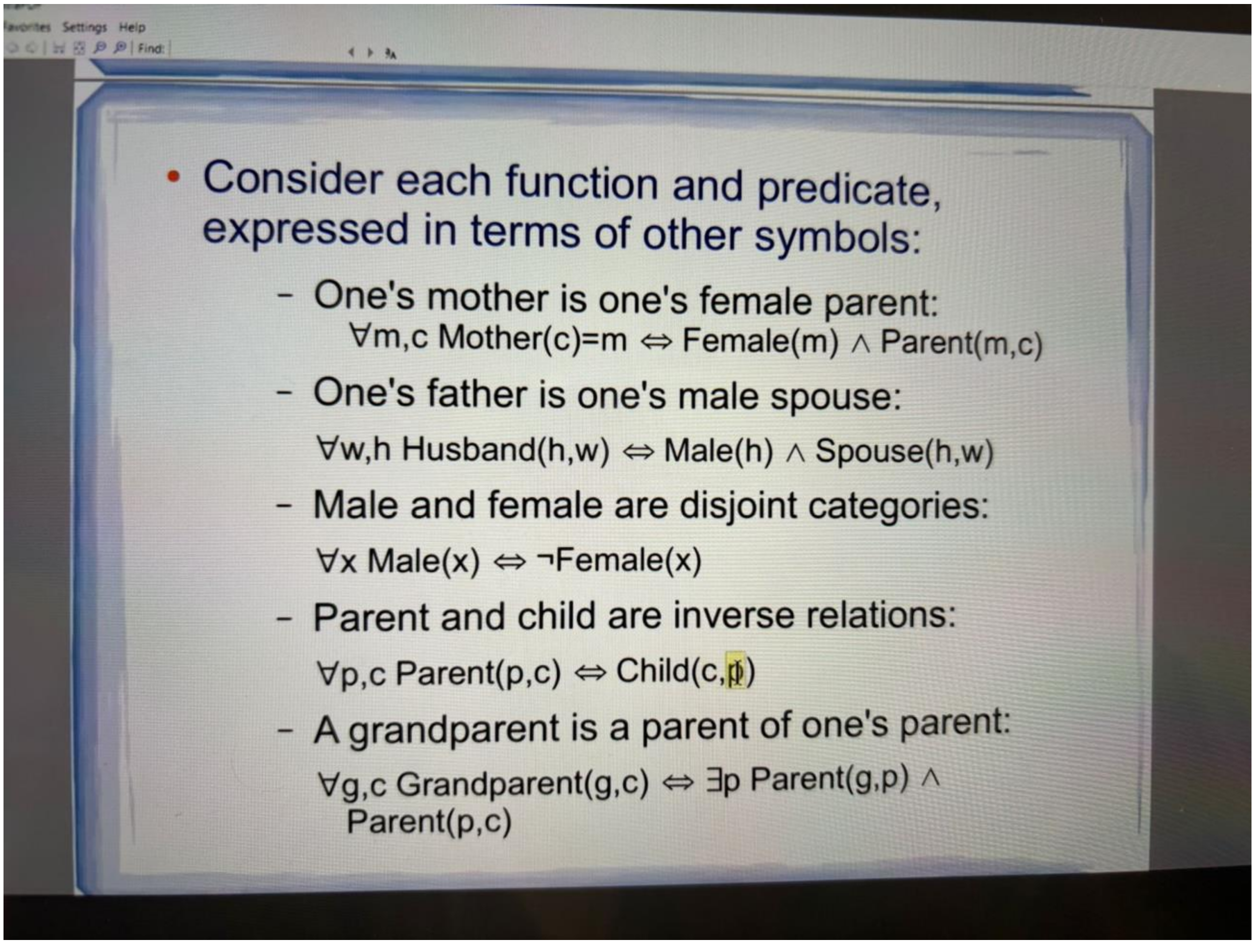
Kinship Axioms:

These are fundamental statements that define relationships and properties in the kinship domain. They can be viewed as axioms or definitions.

1. **Definition Axioms**:
   - Example: `∀m,c Mother(c) = m ⇔ Female(m) ∧ Parent(m,c)` defines the `Mother` function based on gender and parentage.
   - Example: `∀w,h Husband(h,w) ⇔ Male(h) ∧ Spouse(h,w)` defines the `Husband` predicate based on gender and marriage.
2. **Theorems**:
   - These are logical conclusions that follow from the axioms. For example, the assertion `∀x,y Sibling(x,y) ⇔ Sibling(y,x)` is a theorem that siblinghood is symmetric.

Example Use Case:

If we assert `Parent(Elizabeth, Charles)` and `Male(Charles)`, we should be able to infer `∃ x Male(x)` to be true. However, if the knowledge base doesn't contain enough information, we might not get the expected answers, indicating a missing axiom.



• Skip 8.3.3

• Read 8.3.4 about Wumpus world, but understand we have been representing the word using different notation, and only for the task of inferring new facts from "visits" and "breezes"

$\forall x,y,a,b\ Adjacent([x,y],[a,b]) \Leftrightarrow (x = a \land (y = b - 1 \lor y = b + 1)) \lor (y = b \land (x = a - 1 \lor x = a + 1))$.

• 8.4 is a good description of "knowledge engineering" and you should be able to write axioms for a simple domain, but you don't need to understand the details of the circuit domain example

KNOWLEDGE ENGINEERING :

Process of Knowledge base construction.
STEPS:
1. Identify the task
2. Assemble the relevant knowledge

3. Decide on vocab of predicates and 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.