L11 predicate Logic - the language

Chapter II - 1 (Chapter 1 of Part II)

1. Motivations

In propositional logic, the most basic syntax unit is proposition letter which has a meaning of true or false. Such a language lacks the ability of more detailed structure. For example, when we say Joann is the mother of John, and John is the father of Peter. With these detailed structures, we can do reasoning such as Joann is Peter's grandmother. If we are allowed to use propositional letters only, let P denote “Joann is the mother of John”, and Q “John is the father of Peter.” For P and Q, we simply have no idea about how Joann is related to Peter.

For these detailed structures, they are characterized by

- A set of objects
- Relations among these objects.

In the mother example, we have

- Objects: Joann, John, Peter
- Relations
  - Mother-child relation among the objects
  - Father-child relation among the objects
- We introduce a form to represent the relationship
  - *relationName(obj1, obj2, …, objn)*: *obj1, …, and objn* are related under *relationName*. The relation has *n* parameters or has *n-arity*.
  - E.g., *isMother(joann, john)* can be use to represent the relationship between Joann and John, i.e., Joann is the mother of John. The relation name can be any but is good to reflect the intended meaning.
- Meaning: the meaning of a relation among objects is true or false - we call sth like *isMother(joann, john)* an atom or atomic formula because it is the smallest unit that has the truth value.

We also need functions, e.g., +, *, …

- A function has a function name and arity. For example, x*y, y*x
- We have a relation = between two objects,
  e.g., =(x*y, y*x) [or in traditional form: x*y = y*x]

We also need variables, e.g., x is parent of y if x is the father of y or x is the mother of y (when using notations on relations, we have parent(x, y) if father(x, y) or mother(x, y))

For any sets A and B, A is a subset of B if for any x, x \in A implies x \in B.

We need new constructs

- Constants (motivation: we need them to represent a person, a number or an object in general)
- Variables (motivation: we need variables to refer to sth, e.g., for generality in parent example)
- Functions (motivation: we use functions in mathematics so often.)
- Relations (predicates) (Motivation: for human beings, we mainly care about objects and their relationships)
- Quantifiers (motivation: we need quantifiers, such as, for any x, there exist y, y is the mother of x)

We would like to represent "object":

- 1,
- 1+1 or +(1, 1),
- x,
- x+1 or +(x, 1),
- 5 * (x + y) or *(5, +(x, y))

Every "object" has a "value"

Exercise

- Give an example of an arithmetic expression, can rewrite it using the general form of functions, e.g., f(x, y)

We would like to represent relations among objects

- x + y = y + x, i.e., =(+(x, y), +(y, x)) (substitution: x -> x1+x2 y-> x1*x2: x1+x2 + x1*x2 = x1*x2 + x1 + x2)
    - 5 = 5, 4 = 5,
    - +(x, y)   add(x, y, z): z is the sum of x and y
    - *(5, +(x, y)): add(x , y, z) **and** time(z, 5, result)
- Peter is the father of John
    - By relation: father(peter, john), or
    - By function: father(peter) is john
        - Peter's father is brother of Sara's father:
            - Need relation brother(X, Y) meaning X is the brother of Y
            - brother(father(peter), father(sara))

We still need logical connectives

- E.g., brother(father(peter), father(sara)) and brother(father(sara), father(dina))

We also need quantifiers

- For all x, y, x is a parent of y if x is the father of y. (\forall x \forall y, parent(x, y) ← father(x, y))

How about using the new constructs to represent the N-queen problem? Is it easier?

By new constructs, we can represent information a lot easier than using “propositional variables/letters” in proposition logic.

Now read all definitions in the chapter to formalize the constructs/ideas above. The new language is called **predicate calculus**. In fact, it is good idea for you write all formal definitions for all the ideas above.

2. Study the definitions of *formulas*

Now we will give definitions which will allow us to write statement as:
For all x, for all y, parent(x, y) if mother(x, y) or father(x, y).

Definition 2.1: A *language* $\mathcal{L}$ consists of the following disjoint sets of distinct primitive symbols:

(i) Variables: $x, y, z, v, x_0, x_1, \ldots, y_0, y_1, \ldots, \ldots$ (an infinite set)

(ii) Constants: $c, d, c_0, d_0, \ldots$ (any set of them)

(iii) Connectives: $\wedge, \neg, \vee, \rightarrow, \leftrightarrow$

(iv) Quantifiers: $\forall, \exists$

(v) Predicate symbols: $P, Q, R, P_1, P_2, \ldots$ (some set of them for each arity $n = 1, 2, \ldots$. There must be at least one predicate symbol in the language but otherwise there are no restrictions on the number of them for each arity).

(vi) Function symbols: $f, g, h, f_0, f_1, \ldots, g_0, \ldots$ (any set of them for each arity $n = 1, 2, \ldots$. The 0–ary function symbols are simply the constants listed by convention separately in (ii). The set of constant symbols may also be empty, finite or infinite).

(vii) Punctuation: the comma , and (right and left) parentheses) , (.

Definition 2.2: Terms.

(i) Every variable is a term.

(ii) Every constant symbol is a term.

(iii) If f is an n-ary function symbol ($n = 1, 2, \ldots$) and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is also a term.

Definition 2.3: Terms with no variables are called variable-free terms or ground terms.

Definition 2.4: An atomic formula is an expression of the form $R(t_1, ..., t_n)$ where R is an n-ary predicate symbol and $t_1, ..., t_n$ are terms.

Definition 2.5: *Formulas*.

(i) Every atomic formula is a formula.

(ii) If α, β are formulas, then so are $(\alpha \wedge \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$, $(\neg\alpha)$ and $(\alpha \vee \beta)$.

(iii) If v is a variable and α is a formula, then $((\exists v)\alpha)$ and $((\forall v)\alpha)$ are also formulas.