Logic for computer programming

Angelo Gargantini

Testing e verifica del software

AA 2015-16

Section 1

Propositional logic

The aim of logic in computer science is to develop languages to model the situations we encounter as computer science professionals, in such a way that we can reason about them formally. Reasoning about situations means constructing arguments about them; we want to do this formally, so that the arguments are valid and can be defended rigorously, or executed on a machine. Consider the following argument:

Example 1.1

If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

Valid argument

Intuitively, the argument is valid, since if we put the first sentence and the third sentence together, they tell us that if there are no taxis, then John will be late. The second sentence tells us that he was not late, so it must be the case that there were taxis.

Another example

Example 1.2 If it is raining and Jane does not have her umbrella with her, then she will get wet. Jane is not wet. It is raining. Therefore, Jane has her umbrella with her.

Propositions

Example 1.1	Example 1.2
the train is late	it is raining
there are taxis at the station	Jane has her umbrella with her
John is late for his meeting	Jane gets wet.

The argument in each example could be stated without talking about trains and rain, as follows:
If p **and not** q, **then** r. **Not** r. p. **Therefore**, q.

Section 2

Declarative sentences

Declarative sentences

The language we begin with is the language of propositional logic. It is based on propositions, or declarative sentences which one can, in principle, argue as being true or false.

1. The sum of the numbers 3 and 5 equals 8.
2. Jane reacted violently to Jack's accusations.
3. Every even natural number >2 is the sum of two prime numbers.
4. All Martians like pepperoni on their pizza.
5. Albert Camus´etait un´ecrivain fran¸cais.
6. Die W¨urde des Menschen ist unantastbar.

NOT Declarative sentences

The kind of sentences we won't consider here are non-declarative ones, like

- Could you please pass me the salt?
- Ready, steady, go!
- May fortune come your way.

Atomic sentences

Our strategy is to consider certain declarative sentences as being *atomic*, or *indecomposable*, like the sentence

'The number 5 is even.'

We assign certain distinct symbols p, q, r, . . ., or sometimes p1, p2, p3, . . . to each of these atomic sentences and we can then code up more complex sentences in a compositional way. For example, given the atomic sentences

p: 'I won the lottery last week.'

q: 'I purchased a lottery ticket.'

r: 'I won last week's sweepstakes.'

we can form more complex sentences according to the rules below:

Logical connectives

- $\neg$ The negation of p is denoted by $\neg$p and expresses 'I did not win the lottery last week,' or equivalently 'It is not true that I won the lottery last week.'
- $\bigvee$ Given p and r at least one of them is true: 'I won the lottery last week, or I won last week's sweepstakes;' we denote this by p $\bigvee$ r and call it the disjunction of p and r.
- $\bigwedge$ Dually, the formula p $\bigwedge$ r denotes the conjunction of p and r: 'Last week I won the lottery and the sweepstakes.'
- $\rightarrow$ 'If I won the lottery last week, then I purchased a lottery ticket.' expresses an implication between p and q, suggesting that q is a logical consequence of p. We write p $\rightarrow$ q for that . We call p the assumption of p $\rightarrow$ q and q its conclusion.

Section 3

Natural deduction

Natural deduction

- How do we go about constructing a calculus for reasoning about propositions, so that we can establish the validity of propositions
- We would like to have a set of rules each of which allows us to draw a conclusion given a certain arrangement of premises.
- They allow us to infer formulas from other formulas. By applying these rules in succession, we may infer a conclusion from a set of premises
- Suppose we have a set of formulas $\varphi 1$, $\varphi 2$, $\varphi 3$, . . . , φn , which we will call *premises*, and another formula, ψ, which we will call a *conclusion*.
- By applying proof rules to the premises, we hope to get some more formulas, to eventually obtain the conclusion.
- We denote this intention using *sequents.*

Natural deduction

- This intention we denote by

$$\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$$

  This expression is called a *sequent*;
- Constructing such a proof is a creative exercise, a bit like programming. It is not necessarily obvious which rules to apply, and in what order, to obtain the desired conclusion.
- However their correct application can be automatically performed/checked (see theorem provers).
- Additionally, our proof rules should be carefully chosen; otherwise, we might be able to 'prove' invalid patterns of argumentation.
- We present about fifteen rules of them in total; we will go through them in turn and then summarise at the end of this section.

The rules for conjunction

- Our first rule is called the rule for conjunction ($\bigwedge$): and-introduction.
- It allows us to conclude $\varphi \bigwedge \psi$, given that we have already concluded φ and ψ separately:

$$\frac{\varphi \qquad \psi}{\varphi \bigwedge \psi} \bigwedge i.$$

- Above the line are the two premises of the rule.
- Below the line goes the conclusion.
- To the right of the line, we write the name of the rule; $\bigwedge$i is read 'and-introduction'.

and-elimination

$$\frac{\varphi \wedge \psi}{\varphi} \wedge e1. \qquad \frac{\varphi \wedge \psi}{\psi} \wedge e2.(1.1)$$

- The rule $\wedge$e1 says: if you have a proof of $\varphi \wedge \psi$, then by applying this rule you can get a proof of φ.

Example Natural Deduction

example

Let's use these rules to prove that $p \bigwedge q, r \vdash q \bigwedge r$ is valid.

- We start by writing down the premises; then we leave a gap and write the conclusion:

$p \bigwedge q$
r

...

$q \bigwedge r$

- Complete proof

1	$p \bigwedge q$	premise
2	r	premise
3	q	$\bigwedge$e2 1
4	$q \bigwedge r$	$\bigwedge$i 3, 2

Double negation

$$\frac{\neg\neg\varphi}{\varphi}\neg\neg e \qquad \frac{\varphi}{\neg\neg\varphi}\neg\neg i$$

- The sentence 'It is not true that it does not rain.' is just a more contrived way of saying 'It rains.'

Eliminating implication

- Latin name modus ponens. We will usually call it by its modern name, implies-elimination

$$\frac{\varphi \qquad \varphi \to \psi}{\psi} \to e$$

- Let us justify this rule by spelling out instances of some declarative sentences p and q.
  - Suppose that p : It rained.
  - p $\to$ q : If it rained, then the street is wet.
  - so q is just 'The street is wet.'
  - Now, if we know that it rained and if we know that the street is wet in the case that it rained, then we may combine these two pieces of information to conclude that the street is indeed wet.

modus tollens

- Modus tollens

$$\frac{\varphi \rightarrow \psi \qquad \neg\psi}{\neg\varphi}\, MT$$

- Again, let us see an example of this rule in the natural language setting: 'If Abraham Lincoln was Ethiopian, then he was African. Abraham Lincoln was not African; therefore he was not Ethiopian.'

Exercises

- Individua le preposizioni, scrivi e dimostra i sequenti (usando le regole sopra):
  - Se non c'è sole allora ho freddo. Non c'è sole. Ho freddo.
    $\neg\varphi \rightarrow \psi,\ \neg\varphi \quad \vdash \psi$
  - Se studio e sono fortunato, allora prendo 30. Ho studiato. Sono fortunato. Prendo 30.
    $p \wedge q \rightarrow r,\ p,\ q \quad \vdash r$
  - Se piove, allora se ho l'ombrello sto asciutto. Piove. Non sono asciutto. Non avevo l'ombrello.
    $p \rightarrow (q \rightarrow r),\ p,\ \neg r \quad \vdash \neg q$

Other rules

- rule *implies introduction*
- rules for *disjunction*
- rules for *negation*
  - Contradictions are expressions of the form $\varphi \bigwedge \neg\varphi$ or $\neg\varphi \bigwedge \varphi$, where φ is any formula.
  - Other contradictions can be derived from contradictions;
  - Any formula can be derived from a contradiction.

First-Order Logic (FOL)

Extends propositional logic by:

- **Types** , other than boolean
  e.g. int, BankCard, Bunny...
- **Functions** (mathematical)
  e.g. +, max, abs, bonacci,...
  - **Constants** are functions with no arguments
    e.g. 0, 1,
- **Predicates** (functions returning a boolean)
  e.g. isEven, $>$, isPrime...
- Quantiers
  for all ($\forall$), there exists ($\exists$)