Introduction; propositional and predicate logic

Franco Raimondi

Department of Computer Science
School of Science and Technology
Middlesex University
http://www.rmnd.net

Motivations for verification

(citing Amir Pnueli) If we want to delegate complex and sensitive tasks to computers, we need a high degree of confidence in their correctness. Software and hardware failures can cause

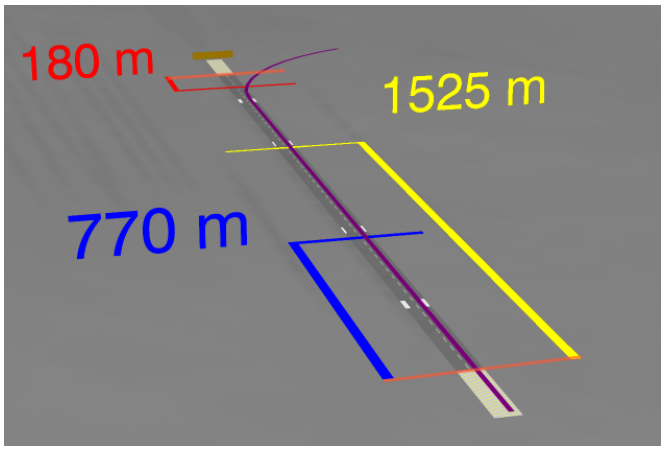
- Life threatening situations (power plants, of course, but also medical artifacts, on-board software for airplanes etc).
- Economic losses (mass produced chips, software controllers in cars, etc.)
- Mission failures (damages to reputation of a company, agency, nation, etc.)

Ariane 5 (1996)

On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded. [...] Loss of information was due to specification and design errors in the software of the inertial reference system. The internal software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value.

Lufthansa Flight 2904 (1993)

To compensate for the windshear, the pilots attempted to touch down with the aircraft banked slightly to the right [...]. But the weather report was not up to date. [...] The aircraft's right gear touched down 770 m from the runway 11 threshold. The left gear touched down 9 seconds later, 1525 m from the threshold. Only when the left gear touched the runway did the ground spoilers and engine thrust reversers deploy. [...] The computer did not actually know the aircraft had landed until it was already 125 meters beyond the half way point of runway.



Pentium FDIV bug (1994)

Certain floating point division operations performed with these processors would produce incorrect results. The floating point division algorithm uses a table of constants with 1066 rows. A bug in the initialization of the table caused only 1061 rows to be correctly initialized.
Cost: approx $500 million.

Validation and Verification

There is general agreement that validation and verification are required steps:

- *Validation* ensures that you built the right thing.
- *Verification* ensures that you built it right.

Various techniques are available, depending on the domain and other parameters.

V&V techniques

- *Simulation* using a model of the system.
- *Testing* a model or the real system.
- *Theorem proving*: the system is a "theory", "prove" the properties (automated)
- *Model checking*: encode properties using a logic formula and the system as a "model" of some logic.

Propositional Logic

Propositional (Boolean) Logic, an example

(Summary of Chapter 1 of Huth and Ryan)
In plain English:

1. *Either no traces of potassium were observed, or the sample did not contain chlorine*
2. *Neither did the sample contain chlorine, nor were traces of potassium observed*

Formally:

1. $((\neg k) \vee (\neg c))$
2. $\neg(c \vee k)$ (or $\neg c \wedge \neg k$)

The syntax of propositional logic

Or: what is a (Boolean) formula?
Well-formed formulae (wff's) of propositional logic are built using the following *connectives*[1]:

$$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$$

and a countable set of *atomic propositions* $a, b, c, \ldots$, and brackets "(" and ")".
These symbols are combined "in an appropriate way" to obtain wff's. For example, $(a \vee b) \Rightarrow c$ is a wff formula, while $(a\vee)bc \Rightarrow$ is not.
Convention: I will use lower case Greek letters to denote wff formulae (for example φ and ψ). I will use capital Greek letters to denote *sets of wff* (for example Σ and Δ).

[1] In theory, only two connectives are needed, for example $\neg$ and $\wedge$. The others could be derived.

Propositional logic semantics

Or: what is the meaning of a formula? When is a formula *true*?
Each atomic proposition can take the value *true* ($\top$) or *false* ($\bot$).
Connectives are evaluated as follows:

α	β	$\neg\alpha$	$\alpha \wedge \beta$
$\top$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$

A **truth assignment** v is a function assigning $\top$ or $\bot$ to each atomic proposition.

Propositional logic semantics – 2

Example: how many truth assignment are there for the formula:

$$\varphi = (a \vee b) \wedge c$$

There are 8 truth assignments. For some of them, $v(\varphi) = \top$, for others $v(\varphi) = \bot$.
Definition: A truth assignment v *satisfies* φ if $v(\varphi) = \top$.
Consider a set of wff formulae Σ and a formula φ.
Definition: Σ *tautologically implies* φ (written $\Sigma \models \varphi$) if every truth assignment that satisfies every member of Σ also satisfies φ.
Example: Let $\Sigma = \{(a \wedge b) \vee c, c \wedge d\}$. Then $\Sigma \models (a \vee c)$.

Propositional logic semantics – 3

Definition: A formula φ is a *tautology* (written $\models \varphi$) if $v(\varphi) = \top$ for every truth assignment.

Definition: A set of wff Σ is satisfiable if there is a truth assignment that satisfies every member of Σ.

Exercise: prove that the following is a tautology:

$$(((a \wedge b) \Rightarrow c) \Leftrightarrow (a \Rightarrow (b \Rightarrow c)))$$

Propositional logic – Satisfiability

SAT problem: Given a propositional formula φ, establish whether there exists a truth assignment v such that $v(\varphi) = \top$.
Example: $a \vee b$ is satisfiable, $a \wedge \neg a$ is not satisfiable.
This is probably the most famous NP-complete problem (Cook's theorem): if you can guess a solution, it takes a polynomial time to verify it.
A number of problems are reduced to SAT. See `http://www.satlive.org/` for problems, tools, discussions, papers, etc.

SAT solvers

Very simple idea: they take a Boolean formula and return either SAT or UNSAT. If SAT, they may print a witness. Examples that you can download and compile:

- PicoSAT: `http://fmv.jku.at/picosat/`
- minisat: `https://github.com/niklasso/minisat`
- glucose-syrup: `http://www.labri.fr/perso/lsimon/glucose/`

Notice: you need to provide formulae in CNF!

DIMACS format

```
c This is
c a comment
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

In the example above: 5 variables, 3 clauses. Let's try with a solver...

Applications of SAT

Too many to list...

- SAT-based (bounded) model checking.
- Planning
- *The packing chromatic number of the infinite square lattice is less than or equal to 16*, `http://arxiv.org/abs/1510.02374` [update: $13 \leq X \leq 15$.

For additional details, see:
`https://courses.cs.washington.edu/courses/csep573/11wi/lectures/ashish-satsolvers.pdf`

First-order logic

First Order Logic (FOL): Syntax

(Huth and Ryan, Chapter 2)
Example: *Every man is mortal*,
For every number x, $(x+2) < x^2$.
Definition: A *vocabulary* $S = (F, \mathcal{R}, r)$ for FOL consists of a set of *functions* F, a set of *relations* $\mathcal{R}$, and a function r called *arity*. r tells how many arguments each function or relation takes. Let $V = \{x, y, \ldots\}$ be a set of variables.
Example: $f(x) : \mathbb{N} \to \mathbb{N}, f(x) = x + 2$ is a 1-ary function.
$R(x, y) =< (x, y)$ is a binary relation (usually written in infix form, $x < y$).
0-ary functions are called *constants*.
Definition: *Terms*. Any variable in V is a term. If $f \in F$ is a k-ary function and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is a term.
Notice that any constant is also a term.

First Order Logic (FOL): Syntax –2

Definition: if $R \in \mathcal{R}$ is a k-ary relation and $t_1, \ldots, t_k$ are terms, then $R(t_1, \ldots, t_k)$ is called an *atomic expression*. *First order expressions* are defined inductively as follows: if φ and ψ are expressions, then so are $\neg\varphi$, $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$. Finally, if x is a variable, then $\forall x\varphi$ is also and expression. Notice: $\exists x\varphi$ is a shorthand for $\neg\forall x\neg\varphi$.

Example, a vocabulary (without any particular meaning):
$F = \{f_1, f_2\}$, $\mathcal{R} = \{R_1\}$, f_1 and R_1 have arity 2, f_2 has arity 3. A wff expression: $(\forall x \exists y R_1(f_1(x, y), y)) \wedge R_1(f_2(x, y, z), z)$.

Occurrence of a variable can be *free* or *bound*. If a variable is under the scope of a quantifier, it is bound. Otherwise, it is free. x and y are bound in the first conjunct of the expression above, $(\forall x \exists y R_1(f_1(x, y), y))$, while x, y, z are free in the second conjunct.

First Order Logic (FOL): Semantics

Or: when is an expression true?
The equivalent of a truth assignment is a *model*.
Definition: Consider a vocabulary S. A *model appropriate to* S is a pair $M = (U, \mu)$ where U is called the *universe* of M and μ is a function assigning to each variable, function and relation actual objects in U (see example above: how is f_1 defined?).
Definition: The definition of M satisfies φ ($M \models \varphi$) is given inductively. Suppose φ is an atomic expression, $\varphi = R(t_1, \ldots, t_k)$. Then $M \models \varphi$ if $(\mu(t_1), \ldots, \mu(t_k)) \in \mu(R)$. If φ is a Boolean combination of expressions: for example, $\varphi = \psi_1 \wedge \psi_2$. Then $M \models \varphi$ if $M \models \psi_1$ and $M \models \psi_2$. The other Boolean connectives are defined in the same way.

First Order Logic (FOL): Semantics –2

Definition (cont): $M \models \forall x \varphi$ if the following is true: let $M(x = u)$ be the model that is identical to M in all details except that $\mu_{M(x=u)}(x) = u$. Then the requirement is that, for all $u \in U$, $M(x = u) \models \varphi$.

Notice: whether a model satisfies or fails to satisfy an expression *does not depend on the values assigned to variables that are bound in the expression*.

Example: A model M for the example above would specify which values are allowed for x, y, z, how are f_1, f_2 defined, and what is R_1. Given these, one can evaluate FOL expressions (for that vocabulary).

Definition: An expression φ is *valid* if it is satisfied by any model. If φ is valid, we write $\models \varphi$.

Example: $\models (\forall x P(x) \vee \neg \forall x P(x))$

Axioms and proofs for FOL

Consider the following set Λ of axioms

- Any expression whose form is a Boolean tautology.
- $(t_1 = t'_1 \wedge \cdots \wedge t_k = t'_k) \Rightarrow (f(t_1, \ldots, t_k) = f(t'_1, \ldots, t'_k))$.
- $(t_1 = t'_1 \wedge \cdots \wedge t_k = t'_k) \Rightarrow (R(t_1, \ldots, t_k) \Rightarrow R(t'_1, \ldots, t'_k))$.
- Any expression of the form $\forall x \varphi \Rightarrow \varphi[x \leftarrow t]$.
- Any expression of the form $\varphi \Rightarrow \forall x \varphi$ with x not free in φ.
- Any expression of the form $(\forall x(\varphi \Rightarrow \psi) \Rightarrow (\forall x \varphi \Rightarrow \forall x \psi))$.

and the following rule:
Rule (*Modus Ponens*): From φ and $\varphi \Rightarrow \psi$, deduce ψ.

Axioms and proofs for FOL – 2

Definition: A *proof* S for an expression φ_n is a sequence $S = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ such that each expression $\varphi_i \in S$ is either an element of Λ, or can be obtained by MP from previous expressions in S. S is a proof of φ_n in Λ, φ_n is called a *first order theorem*, and we write $\vdash \varphi_n$.

What is the relation between $\models \varphi$ and $\vdash \varphi$? Some definitions first:

Definition: Let Δ be a set of expressions, and φ another expression. We say that φ *is a valid consequence of* Δ, written $\Delta \models \varphi$, if any model that satisfies each expression in Δ also satisfies φ. We write $\Delta \vdash \varphi$ if there is a finite sequence of expressions $S = \{\varphi_1, \ldots, \varphi_n\}$ such that $\varphi_n = \varphi$ and, for every $\varphi_i \in S$, either $\varphi_i \in \Lambda \cup \Delta$, or φ_i can be obtained by MP from previous expressions in S. In this case, φ is called a Δ-first-order theorem.

Axioms and proofs for FOL – 3

Definition: A set of expressions Δ is *consistent* if it is not the case that $\Delta \vdash \bot$.

Theorem (*Soundness of FOL*): If $\Delta \vdash \varphi$, then $\Delta \models \varphi$[2].

Theorem (*Completeness of FOL, Gödel theorem*): If $\Delta \models \varphi$, then $\Delta \vdash \varphi$[3].

Theorem (it is equivalent to the completeness theorem): If Δ is consistent, then Δ has a model.

[2] The proof is not difficult, see references if interested.

[3] This proof is a bit more complicated, see references if interested.

Undecidability of FOL

(Church-Turing theorem) There is no mechanical procedure to establish whether $\models \varphi$ holds for an arbitrary formula φ (it may not terminate). See Section 2.5 of Huth and Ryan for more information.
If you are interested in theoretical foundations: H. Enderton, *A Mathematical Introduction to Logic, Second Edition*.