Introduction à la spécification et à la conception: la Méthode B

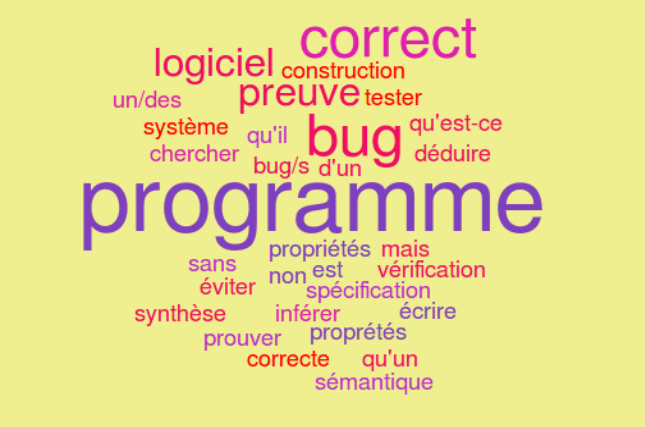
Preuves et Constructions Formelles

J. Christian Attiogbé

Université de Nantes, November 2020

Plan

De quoi allons nous parler



Tout un (petit) univers

Qu'est-ce qu'un logiciel ? qu'est-ce qu'un système ? qu'est-ce un *bug* ? logiciel correct ? logiciel non correct ? éviter bug ? chercher un/des bug/s ? mais qu'est-ce qu'un bug ? tester programme ? écrire programme sans bug ? prouver programme (qu'il est correct) ? spécification ? vérification (de proprétés) ? preuve de propriétés ? système de preuve ? construction correcte ? synthèse de programme ? sémantique d'un programme ? inférer ? déduire ?

☞ Logique, Modèle, Langage

☞ Analyser, exprimer, raisonner, concevoir, construire

“Ce que l’on conçoit bien s’énonce clairement,
et les mots pour le dire arrivent aisément”,

(Art poétique, Nicolas Boileau, 1674)

Formal Methods are required in some contexts

Norme EN 50128 : "Systèmes de signalisation, de télécommunication et de traitement" :
Cette norme traite en particulier des méthodes qu’il est nécessaire d’utiliser pour fournir des logiciels répondant aux exigences d’intégrité de la sécurité appliquées au domaine du ferroviaire. L’intégrité d’un logiciel est répartie sur cinq niveaux SIL, allant de SIL 0 à SIL 4.
Ces niveaux SIL sont définis par association, dans la gestion du risque, de la fréquence et de la conséquence d’un événement dangereux.
Afin de définir précisément le niveau de SIL d’un logiciel, des techniques et des mesures sont définies dans cette norme.
(cf. ClearSy)

SIL : Safety Integrity Level

Standards in Software construction

EN xxx : Normes européennes / IEC xxx : Normes internationales (avec des correspondances) et aussi des DO xxx
DO178C : *Software Considerations in Airborne Systems and Equipment Certification*

Normes	Domaines
IEC 62304	*Medical Device Software Development* Systèmes embarqués, médical
IEC 62279 (EN 50128)	*Railway applications*
IEC 61506	*Industrial-process measurement and control*
...	...

SIL : Safety Integrity Level ☞ Formal methods

The EN 50128 : Software Aspect of Control Systems

Standard NF EN 50128
Titre : Railway Applications, system of signaling, telecommunication and processing equipped with software for the control and the security of railway systems.
Domain: Exclusively applicable to software and to the interaction between software and physical devices;
5 levels of criticity:
Not critical: SIL0,
No dead danger for humans: SIL1, SIL2,
Critical : SIL3, SIL4
Applicable to: the software application; the operating systems ; the CASE[1] tools;

Depending on the projects/contexts, we will **need formal methods** to build dependable systems.

[1] computer-aided software engineering

La logique : construire et raisonner

Logique du premier ordre :

- propositions + Calcul propositionnel (procédure de décision)
- prédicats + Système de preuve

Propriétés :

- exprimées à l'aide de prédicats
  ⚠ différentes catégories de propriétés (différentes logiques).
- utilisées pour construire/analyser des objets

Opérations de base sur les prédicats :

- Substitution $P_X[X = V]$ ou $[V/X]P_X$
- Quantification $\forall x.(P(x) \Rightarrow G(x))$ $\quad$ $\exists y.(P(y) \wedge F(y))$

Logique : Construire et Raisonner sur des objets.

Hoare Logic

Spécification de correction partielle d'un prog. : "triplet de Hoare"
(Floyd/Hoare-Dijkstra - système de déduction / axiomes + règles)

$$\{P\}\ \mathbf{S}\ \{R\}$$

S : instruction ou programme (*statement*)
P : Précondition et R : Postcondition ; ce sont des prédicats.
Correction partielle : car hypothèse de terminaison.
Règles de Hoare (ax. affectation, séquence, conditionnelle, boucle).
Raisonnement sur la structure du programme.

⚠ Terminaison (à prouver, si on veut la correction totale !)

Trouver ou **calculer les préconditions** et les postconditions.
Des conditions plus faibles (*weakening*), plus fortes (*strengthening*)?

Weakest preconditions (Les plus faîbles préconditions)

Context: HOARE/FLOYD/DIJKSTRA LOGIC - HOARE triple
(State, state space, statements, execution, HOARE triple)

$$\{P\}\ S\ \{R\}$$

S a **statement** and R a **predicate that denotes the result of** S.

$wp(S,R)$, is the predicate that describes:
the set of all states such that the execution of S begining with one of them **terminates** in a *finite time* in a state satisfaying R.

$wp(S,R)$ is the *weakest precondition* of S with respect to R.

Some examples

Let S be an assignment (i:=i+1) and R the predicate $i \leq 1$

$$wp(i := i+1, i \leq 1) \ = \ (\mathbf{i \leq 0})$$

Let S be the conditional: if $x \geq y$ then $z := x$ else $z := y$
and R the predicate $z = max(x,y)$

$$wp(S,R) \ = \ \mathbf{Vrai}$$

The wp is a **predicate**; it is computed!

Weakest preconditions - meaning

The meaning of $wp(S, R)$ can be make precise with two properties:

- $wp(S, R)$ is a precondition guarantying R after the execution of S, that is:

$$\{wp(S, R)\}\ S\ \{R\}$$

- $wp(S, R)$ is the weakest of such preconditions, that is:
  if $\{P\}\ S\ \{R\}$ then $P \Rightarrow wp(S, R)$

Weakest preconditions - Predicate Transformer

In practice, a program S establishes a postcondition R;
hence the interest for the precondition that permits to establish R.

wp is a ***function*** with **two parameters**:

a statement (or a program) S and

a predicate R.

For a given S,
$wp(S,\ R)$ viewed as a function with only **one parameter** $wp_S(R)$.
This function wp_S is called *predicate transformer* - DIJKSTRA

It is the function which associates to every predicate R the weakest precondition such that $\{wp_S(R)\}\ S\ \{R\}$.

Predicate Transformer - Programming

- Un prédicat définit des états.
- P sur les entrées du programme, R sur les sorties du programme.
- Lorsque $\{P\}$ **S** $\{R\}$ vrai, **S** transforme l'état décrit par P en l'état décrit par R.

Un programme S se comporte comme un transformateur de prédicat : il transforme un état en un autre état.

☞ Sémantique (des instructions) des programmes ! en Logique !

Introduction: proving the correction of a software

Build correctly a software or
Prove the correction of a software S via its model.

- The **model** of the software : M
- The **properties** : P
- $M \models P$
  proof depending on the structure of the model
  ex: prove that P is true in all the (reachable) states of M
  (if M is a state model)

Anyway, you need a formal model; or rigorous dev. methods.
Do you know some?

☞ Learn how to build M, P and how to prove (Modelling + Verification) (using dedicated tools or not).

Deductive approaches (logic-based) - an overview

- Build a logic model (formal specification) of a system M (or software, product, system...)
- State the required properties for the system P.
- Prove that the model satisfies the properties $M \models P$.
  ?= demonstrate a theorem (from which axioms?)

The proved properties become theorems!

Need the **assistance of provers** (interactive theorem proving)
But, we may want to build [**correct**] programs/software!

Heavy trends: Correct-by-construction (model, prove, refine until code)

Foundation of formal approaches (proof)

Interpretation of the Curry-Howard's Isomorphism:

$$\text{Proof} \quad \frac{\textit{Axioms}}{\textit{Theorems}} \qquad \text{equiv. to} \qquad \text{Development} \quad \frac{\textit{Specifications}}{\textit{Programs}}$$

→ Proof assistants needed! not only editors and compilers

Models and what they are good for?

What is a model?

- An approximation of a reality. It can be built mathematically or not.
- Useful (meaningful) or not.

Given a model M, it can be a reference:

- to build a product/software;
- to reason about a system (a software): predict its behavior: $M \models P$

But

- Can we use it as the program?
- can we transform it into the program?

☺ ☺ ☺

Models: a few definitions

Modelling:
HOARE: A **scientific theory is** formalised as **a mathematical model of reality**, from which can be deduced or calculated the observable properties of a well-defined class of processes in the physical world.

There are two main notions of models in computer science.

1. Model = an approximation of the reality by a math. structure.
   An object O is a model of a reality R, if O allows one to answer all the questions about R.

In Mathematics, Physics, ... models are built with equation systems using quantities (masses, energy, ...) or hypothetic laws.
⇒ State exploration, simulation

Models: a few definitions (continued)

2. Logics, Theory of models
   A model of a theory T is a structure in which the axioms of T are valid.
   A *structure* S is a model of a theory T, or S satisfies T if all formula of T is satisfied in S.
   The reality is a model of a theory!
   First Order Theory = any set of logic formula in a given language (precisely defined).
   Model as an interpretation of a specification - an algebra as a model of an algebraic specification (or an axiomatisation).

⇒ deductive approach, theorem proving

Models: a few definitions (continued ... end)

These two notions of *model* are encountered in the model-oriented (or state-oriented) and property-oriented approaches of Soft. Eng.

In current practical/pragmatic uses,

- *model* = (archetype), what serves or is used for imitation to reproduce orther instances.
- *model* = (paradigm), declination model, conjugation model, etc
- *model* = (reference), . . .