# Introduction au raffinement - Méthode B

## Preuves et Constructions Formelles

J. Christian Attiogbé

Université de Nantes, November 2020


# Plan

1. Introduction
2. Refinement
   - Refinement proof obligations
3. Exercices
4. Conclusion

# Refinement: development technique

**The objective** is the construction of correct executable code.

- Start with an abstract machine : an abstract mathematical model,
- Refine this model to obtain : a concrete model
  - the abstract model is not executable.
    Why? (it is defined with mathematical objects)
  - to an equivalent model, wrt to functionalities, but more concrete.
    (it is described with programming objects)
- We should guaranty that the refinement is correct:
  $\Rightarrow$ refinement proof obligations

There is a well-defined Theory of refinement
[Morgan 1990; R-J. Back 1980; C. Ralph-Johan Back, Joakim Wright, 1998]

# Refinement: development technique

**Principle of refinement ($S_c$ refines $S_a$)**
Abstract machine : $\text{INV}_{abs}$, $\text{U}_{abs}$, $\text{OP}_{abs}$ =($\text{PRE}_{abs}$ | $\text{Subst}_{abs}$)
Refining machine : $\text{INV}_{conc}$, $\text{U}_{conc}$, $\text{OP}_{conc}$ =($\text{PRE}_{conc}$ | $\text{Subst}_{conc}$)

Refinement of : the invariant, the initialisation, the operations.
Each abstract operation is refined by a concrete operation.

$$\textit{if} \quad Subst_{abs} \sqsubseteq Subst_{conc} \quad \textit{then} \quad \forall R.([R]Subst_{abs} \Rightarrow [R]Subst_{conc})$$

The refinement preserves the invariant.

# Approach of refinement

What to refine in the model?

- The **variables and the invariant** (refining the state space)
- The **operations** (refinement of the generalized substitutions). Introduce **refinement substitutions** *(until programming substs)*.
- Use the clause REFINES to link the abstract machine with its refinement

```
REFINEMENT  MM_R1
REFINES     MM
...
INVARIANT
+binding abstract and concrete
variables
...
END
```

# Approach of refinement: How to refine?

Introduce data structures and replace abstract by concrete ones.

- **Think a lot about a strategy**
- Refining the state space:
  - introduce new (concrete) variables,
  - choice of (less abstract) structures,
  - link abstract and concrete variables by a binding invariant
- Refinement of the operations:
  - The operation interfaces should not be modified.
  - Introduce refinement substitutions to rewrite abstract operations with the new variables.
  - Remove non-determinism
  - Weak the preconditions of the abstract oper, until they disappear.

$\Rightarrow$ Need of the extension of the substitution language.

# Refinement substitutions

- **Sequential substitutions**
  Let $S$ and $T$ be substitutions,
  the sequential substitution is noted: $S\ ;\ T$
  Its semantic definition is expressed with:

$$\begin{aligned}[S;T]R &\equiv [S][T]R \\ &\equiv [S]([T]R)\end{aligned}$$

  $S$ establishes $[T]R$

- **Block with local variables**
  The notation is :

  var x in // introduction of local variables
  $S$
  end

# Refinement substitutions

- **Loop substitution**
  The loop substitution has the following shape:

  while $P$ do $S$
  invariant $I$
  variant $V$
  end

Semantically, it is

$$
\begin{array}{l}
I \wedge \\
\qquad \text{/* the variant is a natural */} \\
\forall\, x.(I \Rightarrow V \in NATURAL) \wedge \\
\qquad \text{/* the variant decreases after each step */} \\
\forall\, (x,n).(I \wedge P \Rightarrow [n := V][S](V < n)) \wedge \\
\qquad \text{/* continuation of the loop */} \\
\forall\, x.(I \wedge P \Rightarrow [S]I) \mid \\
\qquad @x'.([x := x'](I \wedge \neg P) \Rightarrow x := x')
\end{array}
$$

# Refinement proof obligations

Intuition: the concrete machine does not contradict the abstract one.

PO for the initialisation:

$$[U_{conc}]\neg([U_{abs}]\neg(Inv_{conc}))$$

PO for each operation:

$$INV_{abs} \wedge INV_{conc} \wedge PRE_{abs} \Rightarrow PRE_{conc}$$

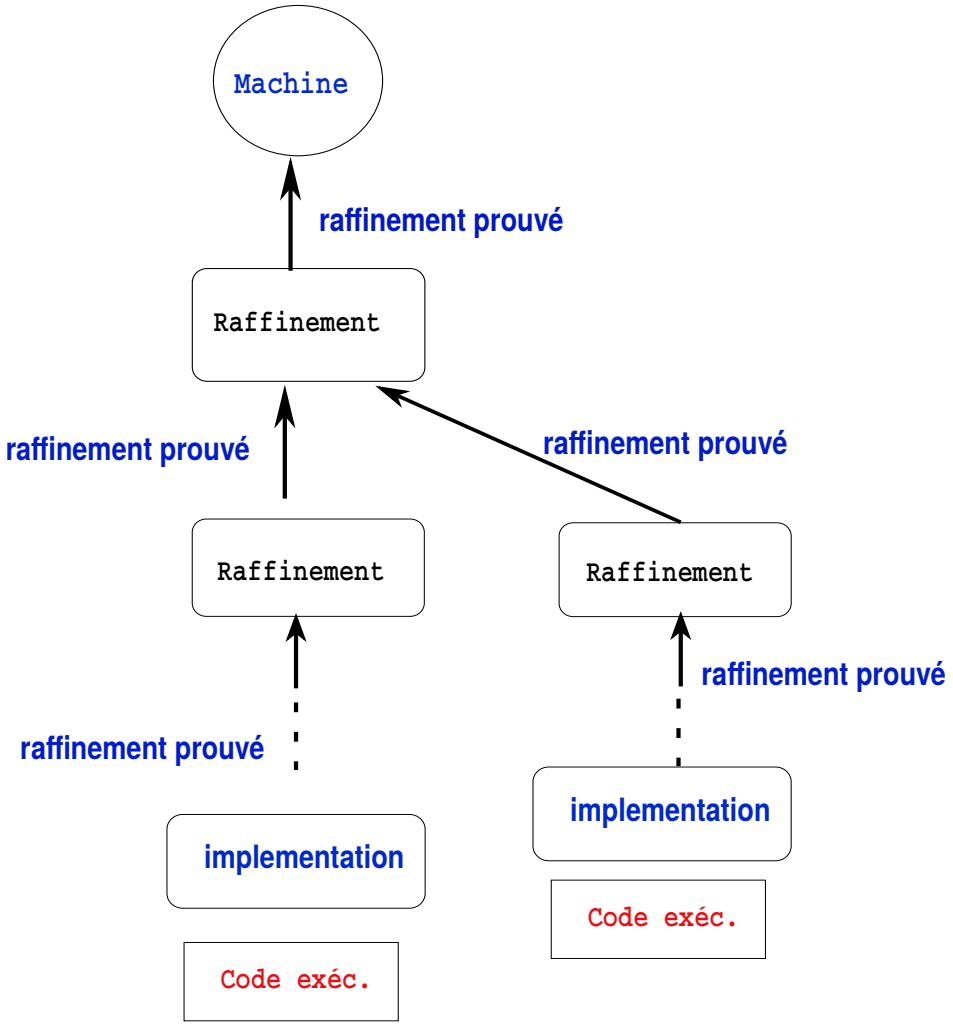
$$INV_{abs} \wedge INV_{conc} \wedge PRE_{abs} \Rightarrow [Subst_{conc}]\neg([Subst_{abs}]\neg(INV_{conc}))$$

# Refinement of operations: practically

| Abstract | Refinement |
|---|---|
| S ‖ T | S ; T |
| PRE P<br>THEN Subst<br>END | BEGIN<br>Concrete(Subst)<br>END |
| ANY ... WHERE ... END | WHILE ... DO ... END |
| CHOICE S OR T END | IF (...) THEN S ELSE T |
| ... | ... |

# Implementation

- Stop the refinement when no more abstract structures
- Implementation = the last refinement step
- Several alternative refinements
- B0 Language
- Translation (at least) into C code.



# Exercices

En TD/TP, mais pas seulement !

# Synthèse

Une introduction à la Méthode B

- Modélisation formelle : approche par états définis par Invariant ; machine abstraite/concrète
- Approche de la correction par construction (à priori)
- Très forte expressivité : Logique, Théorie des ensembles
- Raffinement au code du logiciel (de l'abstrait au concret)
- Preuves de cohérence et des raffinements
- Méthode outillée

Mais on fait encore mieux avec Event-B !