Specification et Verification Formelle

Chapter 03: Method B

Dr. Hichem Debbi

hichem.debbi@gmail.com

April 1, 2022

Spreading in Industry:

Method B

The method B is a software development method that aims to specify a software through many versions or steps. Its syntax is based on set theory and first order logic, and employs abstract machines. It was developed by Jean-Raymond Abrial in the 1980s, and it is very related to its predecessor, Z language.

Method B Principle

The B method aims to verify verifiable and guide software development, so that the resulting final code would be proved to be consistent with the original specification. To do so, Method B uses the Abstract Machine Notation (AMN), which is a famous technique in programming languages.

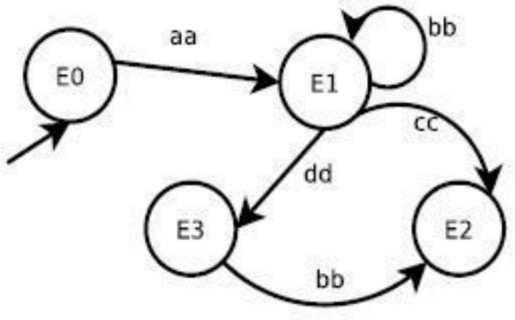
1985
جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Method B Origins

In addition to Z of Jean-Raymond Abrial, It has origins in the following approaches

- Pre and post conditions of Jones
- Guarded commands of Dijkstra
- Data Refinement of He Hoare Sanders
- Refinement Calculus Morgan

The system developped in B describes a state-transition system, in which the nodes represent the system's states, and the transitions equipped by guards are enabled through operations.

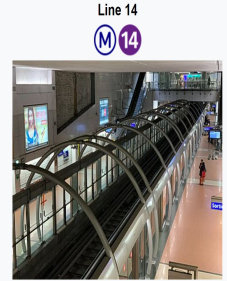


جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Spreading in Industry:

- B is extensively used in Industries (by Clearsy and others)
- Clearsy claims to make 30% of its business with B
- The main industrial activity is with train systems
- Alstom and Siemens Transport actively participate in these activities
- Train systems with B in Europe, North and South America, Asia

جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila



Cour Saint-Émilion in the 12th arrondissement

Metro line 14 in Paris Formal methods have been used for the automatic train operation system for metro line 14, since 1998.
The specification of running and stopping of trains, and the opening and closing of the train doors as well as platform doors, was performed in method-B .

Railways

RER line A in Paris Formal methods have been used for railways for the SACEM system, which is an automatic train protection system that controls the speed of all trains. The system permanently ensures the safety of 0.8 million passengers per day. The formal specification was B method to make the informal requirement specifications more precise.

Metro line 14 in Paris Formal methods have been used for the automatic train operation system for metro line 14, since 1998.
The specification of running and stopping of trains, and the opening and closing of the train doors as well as platform doors, was performed in method-B .

Railway systems in Denmark Formal methods have been used for computer based interlocking systems for stations
RAISE formal method has been used for specifying that no derailing or collisions of trains can happen

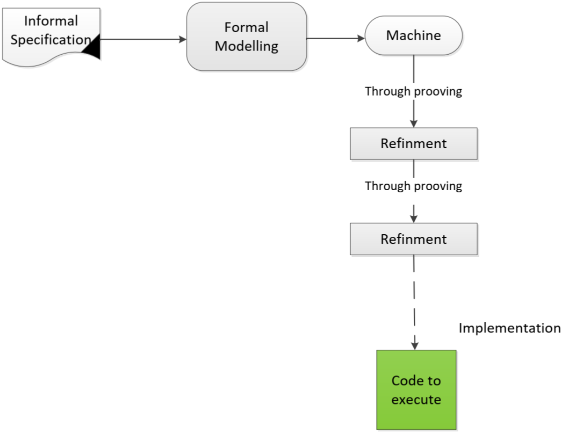
B method constructs

- First order logic
- Set theory
- Theory of substitution for operations
- Theory of refinement

B method constructs

- A predicate is a function from some set X to Boolean.
- Set theory
- Theory of substitution for operations
- Theory of refinement

Basic logical notations



Abstract machine

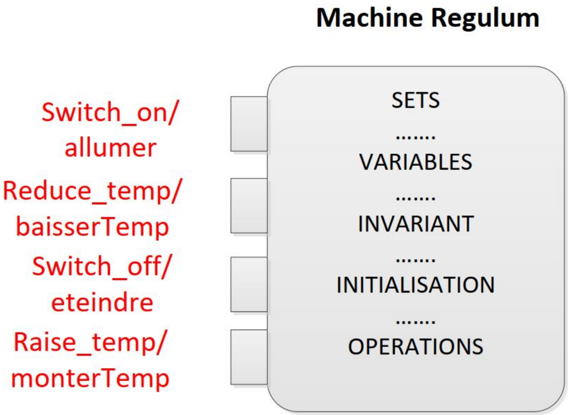

```
MACHINE ...
SETS ...
VARIABLES

....
INVARIANT
... predicate
INITIALISATION

...
OPERATIONS

...
END
```

Regulum machine



Regulum abstract machine


```
MACHINE ReguLum
     SETS MODEJ = {jour, nuit}; ETATLUM = {eteint, allume}
 VARIABLES mode, lumiere, temp
 INVARIANT mode : MODEJ & lumiere : ETATLUM & temp : INT
 INITIALISATION mode := jour || temp := 20 || lumiere := eteint
 OPERATIONS
 changerMode = CHOICE mode := jour OR mode := nuit END;
 allumer = lumiere := allume;
 eteindre = lumiere := eteint;
 baisserTemp = temp := temp - 1;
 monterTemp = temp := temp + 1
END
```

1985
جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Regulum implementation


```
IMPLEMENTATION ReguLum_i
REFINES ReguLum

CONCRETE_VARIABLES mode, lumiere, temp
INITIALISATION mode := jour; temp := 20; lumiere := eteint
OPERATIONS
    changerMode = IF mode = jour THEN
    mode := nuit
    ELSE
    mode := jour
    END;
    allumer = lumiere := allume;
    eteindre = lumiere := eteint;
    baisserTemp = temp := temp - 1;
    monterTemp = temp := temp + 1
 END
```

جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

B Constructs explained

- Each machine has a name
- *SETS* clause considers abstract or enumerable sets, which are used for variables typing. Predefined sets: NAT, INTEGER, BOOL, etc.
- *VARIABLES* clause lists the variables
- *INVARIANT* clause lists the predicates that describe properties invariants of the abstract machine.
- *INITIALISATION* clause initializes all the listed variables. Modifications cn be done later through OPERATIONS.
- *Operations* clause aim to describe of changes of the machine's state, through logical substitutions $:=$. Postconditions must always respect the invariants.

- This machine has four(04) main operations: (Switch_oN, Reduce_temp, Switch_off, Raise_temp) and another operation: changeMode.
- Invariants on the machine's behavior can be expressed such as:
  - Switchon only at night
  - The temperature should not exceed 29 degree by day.

جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Substitution

- Logical substitution is an operation that transforms a state into another correct state
- Transforming a state = transforming a predicate.
- Operation = Substitution = Predicate transformer

Substitution

- An operation can not be accessed from another operation in the same machine (Respecting the precondition)
- From an external machine, we can not call or access two operations of the same machine in the same time (e.g. Increment/Decrement)
- A machine can include secondary operations for verifying preconditions of principal operations.
- The caller of an operation must verify its preconditions (e.g. division by zero)
- Through *Refinement* we try to weakness the preconditions until they disappear.

Logical context

In the logic Hoare/Floyd/Dijkstra and triplet of Hoare , we have the concept of state, space states, commands and execution. We write
$\{P\}S\{R\}$
Where R refers to the result's predicate of S, and $P = wp(S, R)$ characterizes the set of all states before executing S, such that . executing S results in a state satisfying R.
We consider $wp(S, R)$ as the weakest precondition of S with respect to R
The weakest precondition was introduced by Dijkstra in 1975.

Substitution

Université Mohamed Boudiaf - M'sila
جامعة محمد بوضياف - المسيلة
1985

Examples

- Let S be an affectation and R the predicate $i \leq 1$ then: $wp(i := i + 1, i \leq 1) = (i <= 0)$
- Let S be the conditional: $if\ x \geq y$ **then** $z := x\ else\ z := y$ and R the predicate $z = max(x, y)$, then $wp(S, R) = true$.

Université Mohamed Boudiaf - M'sila

Genralized Substitutions

- Simple substitution: semantically: $S[R]$: S transforms R
- Multiple substitution: $x, y := E, F$: semantically $[x, y := E, F]R$.

Substitution

Substitution simple

BEGIN
  S
END

Simultanious Substitution

Let S and T two substitutions. S being $x := E$and T being $y := F$ we note $S||T$

Substitution with precondition ($P|S$)

```
PRE
   P
THEN
   S
END
```

Substitution

Bounded choice $(S || T)$

Choice
  S
OR
  T
END

Substitution with guard $(S || T)$

IF P
THEN T
ELSE S
END

Substitution

Université Mohamed Boudiaf - M'sila
جامعة محمد بوضياف - المسيلة
1985

Non-determinsitic $(@x.(P \implies S))$

ANY X
WHERE P
THEN S
END

Machine - General form


```
MACHINE
    M (prm) /* Name of the machine and its parameters*/
CONSTRAINTS
    C /* list of clauses such as: uses, sees, includes,*/
SETS
    S /* list Of Sets*/
CONSTANTS
    K /* list of Identifiers or constants */
PROPERTIES
   P /* Predicates on K*/
VARIABLES
    V /* list of Variables */
DEFINITIONS
    D /* LIST OF Definitions*/
```

1985
جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Machine - General form


```
INVARIANT
    I /* LIST OF Invariants*/
INITIALISATION
    U
OPERATIONS
    u <- O(pp) =
    PRE
        Q
    THEN
        Subst
    END;
    . . .
END
```

Université Mohamed Boudiaf - M'sila

Machine Consistency

Constraints

Parameters values should satisfy the constraints.
$\exists prm.C$

Properties

A set of constants should satisfy the properties.
$C \implies \exists(S, K).P$

Invariants

There exists a state satisfying the invariant.
$P \wedge C \implies \exists V.I$

Machine Consistency

Université Mohamed Boudiaf - M'sila
جامعة محمد بوضياف - المسيلة

Initialization

The Initialization establishes the Invariant.

$P \wedge C \implies [U]I$

Operations

For each operation,

$P \wedge C \wedge I \wedge Pre \implies [Subs]I$ Every called operation preserves the invariant under its precondition

Proof obligations

جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Are a set of predicates, which helps to guarantee the consistency of the abstract machine mathematically.
Two essential types of obligations :

- The INITIALIZATION establish the Invariant.
- Each operation, should preserve the Invariant under its Precondition.

Proof obligations

Are a set of predicates, which helps to guarantee the consistency of the abstract machine mathematically.
Two essential types of obligations :

- The INITIALIZATION establish the Invariant.
- Each operation, should preserve the Invariant under its Precondition.

Resources Example

n_rsrc ∈ 0..100
n_rsrc = cardinal Of the set

reserve → - 1 element
free → + 1 element

Resources Example


```
MACHINE
    Reservation
VARIABLES
    n_rsrc
INVARIANT
    n_rsrc : 0..100
INITIALISATION
    n_rsrc := 100
OPERATIONS
    reserve =
    PRE n_rsrc > 0
    THEN
        n_rsrc := n_rsrc - 1
    END;
free =
       PRE n_rsrc < 100
       THEN
           n_rsrc := n_rsrc + 1
       END;
bb <-- disponibility =
    bb := bool(0 < n_rsrc)
END
```

Université Mohamed Boudiaf - M'sila

We have to prove that the INITIALISATION respects the invariant :

$[n\_rsrc := 100](\mathbf{n\_rsrc} \in \mathbf{0..100})$

We have to prove that:

$100 \in 0..100$

Resources - Proof coherence

Université Mohamed Boudiaf - M'sila

We have to prove that each operation once called, its precondition should satisfy the invariant.

Reserve

We have to prove:
$\mathbf{n\_rsrc} \in \mathbf{0..100} \wedge 0 < n\_rsrc \implies n\_rsrc - 1 \in 0..100$

Disponibility

We have to prove:
$\mathbf{n\_rsrc} \in \mathbf{0..100} \wedge (n\_rsrc > 0 \vee \neg(n\_rsrc > 0)) \implies n\_rsrc \in 0..100$

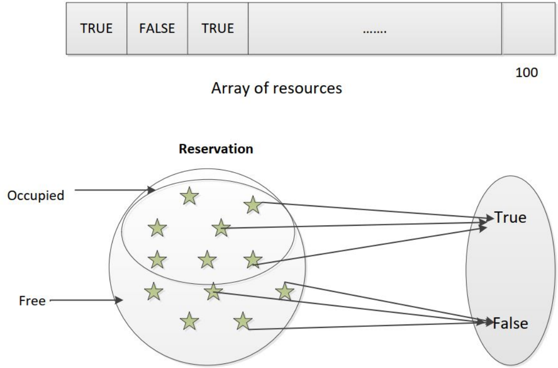
Resources Refinement


```
REFINEMENT
    Reservation_1
REFINES
    Reservation
VARIABLES
    n_rsrc, r_libres, r_occupees
INVARIANT
    r_free : POW(INTEGER)
    & r_occupied : POW(INTEGER)
    & r_free /\ r_occupied = {}
    & n_rsrc = card(r_free)
INITIALISATION
r_free, r_occupied, n_rsrc := 1..100, {}, 100
OPERATIONS
    reserve =
    ANY ss WHERE
        ss : r_libres
    THEN
    r_libres := r_libres - {ss}
    || r_occupees := r_occupees \/ {ss}
    || n_rsrc := n_rsrc - 1
END;
```

Resources Refinement


```
free =
    ANY ss WHERE
        ss : r_occupied
    THEN
        r_free := r_free \/ {ss}
        || r_occupied := r_occupied - {ss}
        || n_rsrc := n_rsrc + 1
    END ;
bbp <-- disponibility =
    IF 0 < n_rsrc
    THEN
        bbp := TRUE
    ELSE
        bbp := FALSE
    END
END
```

Resources Implementation



Université Mohamed Boudiaf - M'sila

Refinement

We have to prove that each operation once called, its precondition should satisfy the invariant.

Definition

Refinement is a technique of transforming abstract models of a software into more concrete model. we call the new resulting model a refinement, and should preserve the same behavior of the previous model.

The concrete model or refinement should characterized by:

- It introduces more details on the specification
- It is closer to the final implementation
- It reduces Non-determinism

1985
جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Through refinement wa can reach the specification SP_n starting from SP_1 progressively:
$SP_1 \rightarrow SP_2 \rightarrow ..... \rightarrow SP_n$
By applying the proof obligations at each step: SP_{i+1} must preserve the behavior of SP_i.

Types of refinement:

- Data Refinement
- Control Refinement
- Algorithmic refinement

Refinement - Data

- Introduction of more concrete variables and sets
- Define operations employing concrete variables
- Define abstraction relation between the variables of abstract machine and the variables of the refined one through the linkage invariant.

Refinement - Control

- The operations conserve the same signature: same operations and same parameters
- Weaken the precondition: A new precondition of the substitution T should be weaker than the previous substitution S
- Reducing the non-determinism: T must be more deterministic than S.

Refinement

Université Mohamed Boudiaf - M'sila

جامعة محمد بوضياف - المسيلة

Reduce nondeterminism

Nondeterminism is about having a choice, where any of the outcomes might be satisfactory, thus we can those among these values.

Simple example

$S\_Res \leftarrow ChoiceValue = result \in \{1, 2, 3\}$ can be refined by $T\_Res \leftarrow ChoiceValue = result := 3$

Concrete Example

If an architecture wants to design a home, he let some details concerning the materials not defined until further steps in design. For instance, Always there is always choice between tiled roofs and slates. Such a final decision might be identified just before construction.

Machine abstraite	Machine raffinement
MACHINE MaisonAbstraite SETS TYPE_TOIT = {ardoises, tuiles} VARIABLES toit INVARIANT toit : TYPE_TOIT INITIALISATION toit :: TYPE_TOIT OPERATIONS choix_toit = CHOICE toit := ardoises OR toit := tuiles END END	REFINEMENT MaisonRaff REFINES MaisonAbstraite VARIABLES letoit INVARIANT letoit = toit INITIALISATION letoit := ardoises OPERATIONS choix_toit = letoit := tuiles END

1985
جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Weak precondition

given that the behavior of a refined operation is specified on the assumption of its precondition, the refinement can do anything outside of the precondition

Example

$$\begin{array}{l} \textit{result} \longleftarrow \textbf{Divide}\ (\ \textit{n1}\ ,\ \textit{n2}\ )\ \widehat{=} \\ \quad \textbf{PRE}\ \ \textit{n1} \in \mathbb{N} \land \textit{n2} \in \mathbb{N} \land \textit{n2} \neq \textit{0} \\ \quad \textbf{THEN}\ \ \textit{result} := \textit{n1}\ /\ \textit{n2} \\ \quad \textbf{END} \end{array}$$

$$\begin{array}{l} \textit{result} \longleftarrow \textbf{Divide}\ (\ \textit{n1}\ ,\ \textit{n2}\ )\ \widehat{=} \\ \quad \textbf{IF}\ \ \textit{n2} \neq \textit{0}\ \textbf{THEN}\ \ \textit{result} := \textit{n1}\ /\ \textit{n2} \\ \quad\quad \textbf{ELSE}\ \ \textit{result} := 27 \\ \quad\quad \textbf{END} \end{array}$$

Université Mohamed Boudiaf - M'sila

Concrete Example

Machine abstraite	Machine raffinement
MACHINE MAbstraite VARIABLES xx INVARIANT xx : 1.. 10 INITIALISATION xx := 1 OPERATIONS plus1 = PRE xx < 10 THEN xx := xx + 1 END END	REFINEMENT MoinsAbstraite REFINES MAbstraite VARIABLES yy INVARIANT yy = xx INITIALISATION yy := 1 OPERATIONS plus1 = PRE yy < 15 THEN yy := yy + 1 END END

Université Mohamed Boudiaf - M'sila

Generalities on Refinement

$$\left\{\begin{array}{l} MACHINE\ MA \\ \quad SETS\ C \\ \quad VARIABLES\ a \\ \quad INVARIANT\ I \\ \quad INITIALISATION\ Init \\ \quad OPERATIONS \\ \quad\quad Op\ =\ P|S \\ END \end{array}\right\} \sqsubseteq \left\{\begin{array}{l} REFINEMENT\ MB\ REFINES\ MA \\ \quad SETS\ D \\ \quad VARIABLES\ b \\ \quad INVARIANT\ J \\ \quad INITIALISATION\ Init' \\ \quad OPERATIONS \\ \quad\quad Op\ =\ P'|S' \\ END \end{array}\right\}$$

Conclusions on Refinement

- Abstract sets in *MA* are implicitly present in *MB*
- the abstract variables *a* are refined by the concrete variables *b*
  - Typing the concrete variables introduced by the refinement
  - Express properties on the concrete variables
  - introduce the *linking invariant*, which relates the concrete variables to the abstract ones
- The concrete initialisation $Init'$ refines the abstract one *Init*
- The operation *Op* is refined by the operation Op' preserving the same signature.

Implementation- Forbidden constructs

- Non-determinism is not allowed, since Implementations must contain only constructs that can be executed.
- simultaneous substitution ($||$)
- preconditioned substitutions
- Assignments involving abstract variable types

Implementation- Concrete Types

- Enumerable sets
- Integers: **NAT1**, **NAT** or **INT** between MININT and MAXINT
- Booleans: **BOOL**
- subsets of concrete types
- Concrete integer intervals
- **Arrays**: a total function of the form $T_1 X T_2 X ....T_n \rightarrow T$ where each T_i is a concrete set.

Université Mohamed Boudiaf - M'sila

References

- The B-Book: Assigning Programs to Meanings, J. R. Abrial
- Specification en B Support de cours Ecole des Jeunes Chercheurs en Programmation EJCP 2007
- La Methode B, Christian Attiogbe Faculte des sciences – Universite de Nantes
- METHODE B – CLEARSY
- http://www.computing.surrey.ac.uk/personal/st/H.Treharne/te
- M. Bourahla, Cours en methode B. Université de M'sila

جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila