La méthode B

La méthode B: Introduction

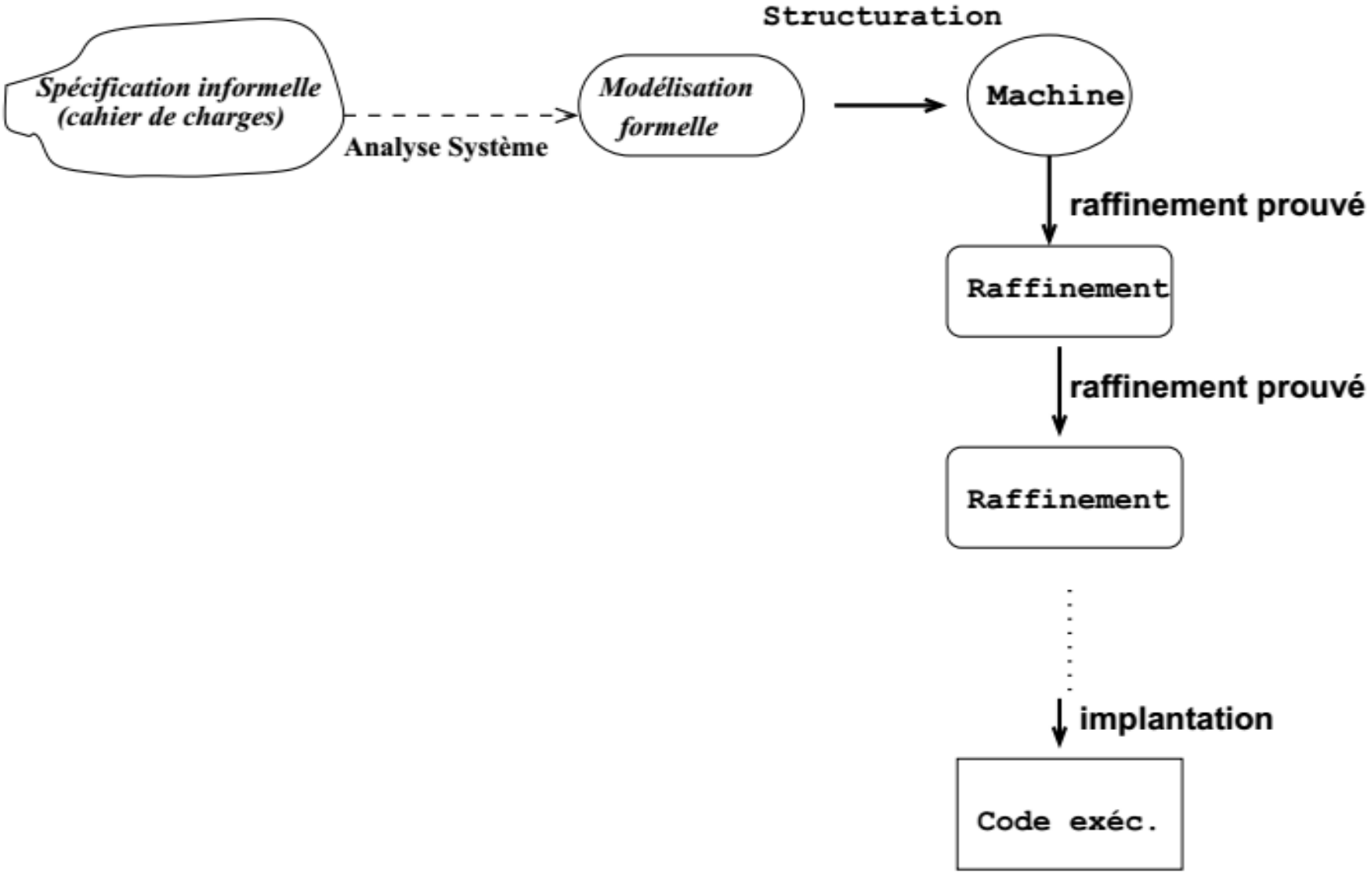
Méthode de spécification formelle :

- fondée sur la théorie des ensembles et
  la logique des prédicats du premier ordre

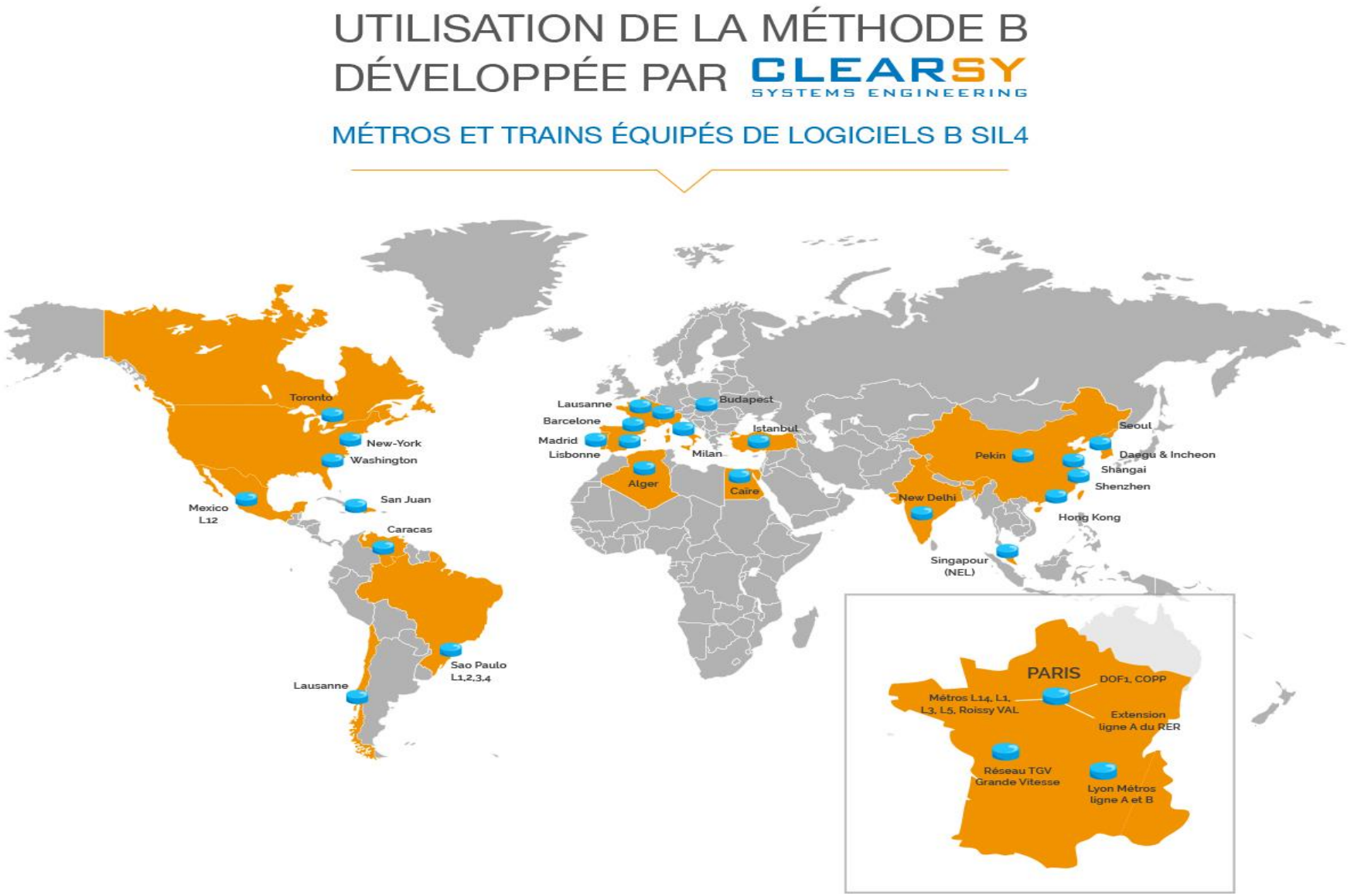
- permet la preuve de propriétés des modèles et
  leur transformation (raffinement) pour obtenir du code

La méthode B: Principe

Transformation systématique d'un modèle mathématique en code exécutable.



La méthode B: place dans le monde



La méthode B: Outils et applications

- **Dotée d'outils logiciels :**

  – Atelier B : logiciel disponible sous licence
  – B4Free + Click'n'Prove : logiciels gratuits pour l'enseignement

- **Quelques applications industrielles :**

  – cartes à puce (Gemplus, Fr)
  – ligne automatique de métro Météor
  – Avionique (GEC Marconi, UK)
  – TGV (Alsthom Transport, Fr)
  – informatique embarquée 206, 307 et 407 (Peugeot, Fr)

Formalisme B: Logique du premier ordre

- Opérateurs booléens: P $\Rightarrow$ Q, $P \wedge$ Q , $\neg P$, …
- Quantificateurs : $\forall x \bullet$ P, $\exists$ x • P
- Substitution dans un prédicat notée [S] P:

application de S à toutes les variables libres de P

Exemple: [X:=X+1] (x=5) donne X+1=5

S P

- Prédicat d'égalité : =

Formalisme B: Théorie des ensembles

- **Rappel:** Un ensemble est une collection d'objets sans ordre (non ordonné) et sans répétition (sans double)
  Exemple: $A = \{1,4,8,12\}$; Définition en extension
  $B = \{ x \geq 0\}$; définition en compréhension

- **Les opérations**:

  Appartenance $x \in E$, x est un élément de E.

  Ensemble des parties: P(E) est l'ensemble des parties de E
  $S \in P(E) <=> \forall e, (e \in S => e \in E)$

  Inclusion $S \subset E$

  Intersection, Union, Différence : $E_1 \cap E_2$, $E_1 \cup E_2, E_1 - E_2,$

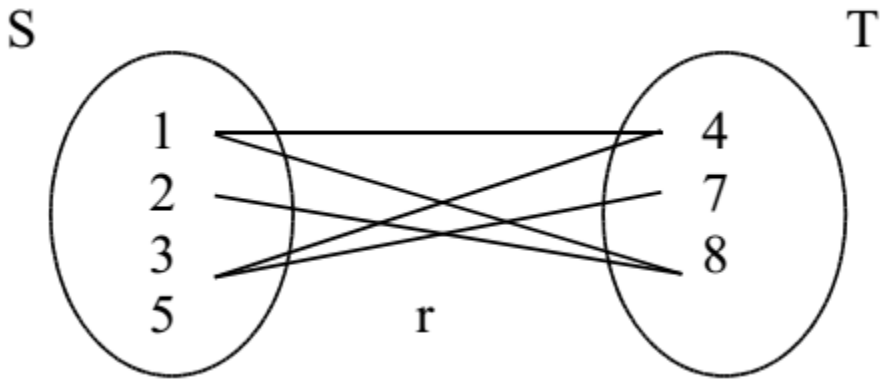
  Produit cartésien *:* $E_1 \times E_2 = [\, x \longmapsto y \,|x \in E_1 \; et \; y \in E_2]$

  Choix(E) : renvoie un élément de E

Formalisme B: modélisation des relations

- **<u>Une relation</u>** est un ensemble de paires d'éléments.

On note $S \leftrightarrow T = \mathrm{P}(S \times T)$, ensemble de toutes les relations entre S et T



$r \in S \leftrightarrow T$, r = {1 ⟼ 4, 1 ⟼ 8, 2 ⟼ 8, 3 ⟼ 4, 3 ⟼7}

dom(r)= {1,2,5}, **ran**(r)= {4,7,8}

Formalisme B: modélisation des fonctions

- **<u>Une fonction</u>** est une relation telle que chaque élément a une image au maximum.
  On note **S** $\twoheadrightarrow$ **T** l'ensemble des fonctions partielles de S dans T

$$S \twoheadrightarrow T = \{f / f \in S \leftrightarrow T \wedge \forall \mathrm{x} \in \mathrm{S}, (\mathrm{y}, \mathrm{z}) | \in \mathrm{T} \times \mathrm{T}. (\mathrm{x} \mapsto \mathrm{y} \in \mathrm{f}, x \mapsto z \in f \Rightarrow y = z)\}$$

Une fonction est une relation donc un ensemble: tous les opérateurs ensemblistes et relationnels s'appliquent sur une fonction.

Formalisme B: les substitutions

- **Définition**: un ensemble de constructions permettant de définir les opérations des machines abstraites. Chaque substitution est considérée comme une transformation de prédicats.
- similaires à la notion d'instruction

- **Quelques substitutions de base** :
  - **substitution simple:** $x := E$, x une variable, E une expression
  - **substitution pré conditionnelle:** $pre\ \varphi\ Then\ S\ End$

  On applique S si φ est vérifiée.
  - **substitution non déterministe** : $Any\ x_1 \dots x_n\ Where\ \varphi\ then\ S$

  consiste à choisir n'importe quelle $x_1 \dots x_n$ qui vérifie φ pour appliquer/exécuter S.
  - **Composition parallèle**: G || H, faire G et H simultanément

Formalisme B: les substitutions

- **Exemple** : On veut spécifier une opération qui alloue un mot dans une mémoire adressée et retourne l'adresse de l'emplacement alloué, s'il y a de la place en mémoire.

- On introduit des ensembles abstraits: *adresse, mémoire, libres* tels que:

$$memoire \subset adresse$$

$$libres \subset \text{mémoire}$$

entête de l'opération : $r \leftarrow alloué$

$pre\ libres \neq \emptyset\ Then$

$Any\ V\ Where\ V \in libres\ Then$

$libres = libres - \{V\}\ ||\quad r := V$

end.

end.

Formalisme B: les substitutions

- **Les substitutions** sont des **transformateurs de prédicats** qu'on note $[S]P$: produit une nouvelle formule résultat de l'application de la substitution S au prédicat P.

- Exemple:

  $[x := y + 1](x \in 0 \ldots 5) \equiv (y + 1 \in 0..5)$

  [S] P

- $Pre\ \varphi\ Then\ S \Rightarrow$ comme transformateur de prédicats

  $[Pre\ \varphi\ Then\ S]P \Leftrightarrow Pre\ \varphi\ \wedge [S]P$

Machine abstraite

- **Définition:**

une machine abstraite peut être assimilé à un objet comprenant un état interne (attributs - variables) et des moyens d'actions sur ces états (opérations).

MACHINE	nom de la machine
SETS	déclaration des ensembles dont se servira la machine
VARIABLES	déclaration des variables qu'utilise la machine
INVARIANTS	formule représentant la propriété globale que doit satisfaire la spécification
INITIALISATION	initialise les variables
OPERATION	liste des opérations ou les actions qui peuvent modifier/manipuler l'état de la machine

END

Machine abstraite

- **Exemple**

Machine Exemple
Sets Produits
Variables Produit
Invariant Produit $\subseteq$ Produits
Initialisation Produit $:= \emptyset$
Opérations
$Creer(Pdt)$
$\boldsymbol{Pre}\ pdt \in Produits - Produit\ \boldsymbol{Then}$
$Produit = Produit\ \cup \{Pdt\}$
End.
$Supp(Pdt)$
$\boldsymbol{Pre}\ pdt \in Produit\ \boldsymbol{Then}$
$Produit := Produit - \{Pdt\}$
End.

End

Preuve de propriétés: Cohérence des machines abstraites

- Principe: prouver que la partie dynamique (les opérations) respecte la partie statique (l'invariant).

- Etant donnée une machine abstraite M

Machine M
Sets S
Variables V
Invariants Inv
Initialisation Init
Operations $OP = P|S$
End.

La preuve de cohérence consiste à vérifier que:

- L'initialisation des états de la machine (Init) respecte l'invariant

$$[\boldsymbol{Init}]\ \boldsymbol{Inv}$$

- Chaque opération OP doit vérifier l'invariant

$$\boldsymbol{P} \wedge \boldsymbol{Inv} => [\boldsymbol{S}]\ \boldsymbol{Inv}$$

Preuve de propriétés: Cohérence des machines abstraites

- Machine Pixel
  Variables x, y
  Invariants $x \in 1..1280 \ \wedge \ \mathrm{y} \in 1..1024$
  Initialisation $x, y := 1 ; 1$
  Opérations
  $pos_x \leftarrow Abs \ = \ pos_x := x$; End
  $pos_y \leftarrow Ord \ = \ pos_y := y$; End
  $move(dx, dy) =$
  $Pre \ (dx \in Z \wedge \mathrm{dy} \in \mathrm{Z} \ \wedge (\mathrm{dx} + \mathrm{x}) \in 1..1280) \wedge (\mathrm{dy} + \mathrm{y}) \in 1..1024)$
  $Then \ x, y := (x + dx) ; (y + dy)$
  End
  End

Il faut montre que :

1) $[\boldsymbol{Init}] \ \boldsymbol{Inv}$

2) $\boldsymbol{P} \wedge \boldsymbol{Inv} => [\boldsymbol{S}] \ \boldsymbol{Inv}$

Preuve de propriétés: Cohérence des machines abstraites

- $[\boldsymbol{Init}]\ \boldsymbol{Inv}$

$[x, y := 1,1](x \in 1..1280) \wedge (\mathrm{y} \in 1..1024)$

- $\boldsymbol{P \wedge Inv} \Rightarrow [\boldsymbol{S}]\ \ \boldsymbol{Inv}$

$(dx \in Z \wedge \mathrm{dy} \in \mathrm{Z} \wedge (\mathrm{dx} + \mathrm{x}) \in 1..1280 \wedge (\mathrm{dy} + \mathrm{y}) \in 1..1024 \wedge (\mathrm{x} \in 1..1280) \wedge (\mathrm{y} \in 1..1024))$
$\Rightarrow [\ \mathrm{x};\mathrm{y}=(\mathrm{x}+\mathrm{dx});(\mathrm{y}+\mathrm{dy})\ ]\ (\mathrm{x} \in 1..1280) \wedge (\mathrm{y} \in 1..1024)$

$= (x + dx \in 1..1280) \wedge (y + dy \in 1..1024)$

Bibliographie

-Jean-Raymond Abrial. Modeling in Event-B. System and Software Engineering. CAMBRIDGE UNIVERSITY PRESS

-Frédéric Gervais. La méthode B

-Marie-Laure Potet Didier Bert . La méthode B. Cours donné à l'Ecole des Jeunes Chercheurs en Programmation Dinard

-Christian Attiogbé. La méthode B. Faculté des sciences Université de Nantes