Introduction aux méthodes formelles

Découverte de la méthode B – TDs

Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence. (Edsger Dijkstra)

Nicolas Stouls et Stéphane Frénot

Pour toute question : Nicolas.Stouls@insa-lyon.fr

Université de Lyon

citi lab

INSA Lyon

Introduction générale

Principes généraux d'un modèle B

Un modèle est constitué de :

- Données *(Logique 1er ordre ensembliste)*
  - Constantes & variables
  - Caractérisés par : Typage & propriétés invariantes
- Actions *(Substitutions généralisées)*
  - Initialisation
  - Opérations

Que cherche-t-on à vérifier ?

Idées générale

- Invariant : *condition décrivant les états autorisés*
- Objectif : *démontrer qu'il n'existe pas de séquence d'appels des opérations permettant de violer l'invariant.*

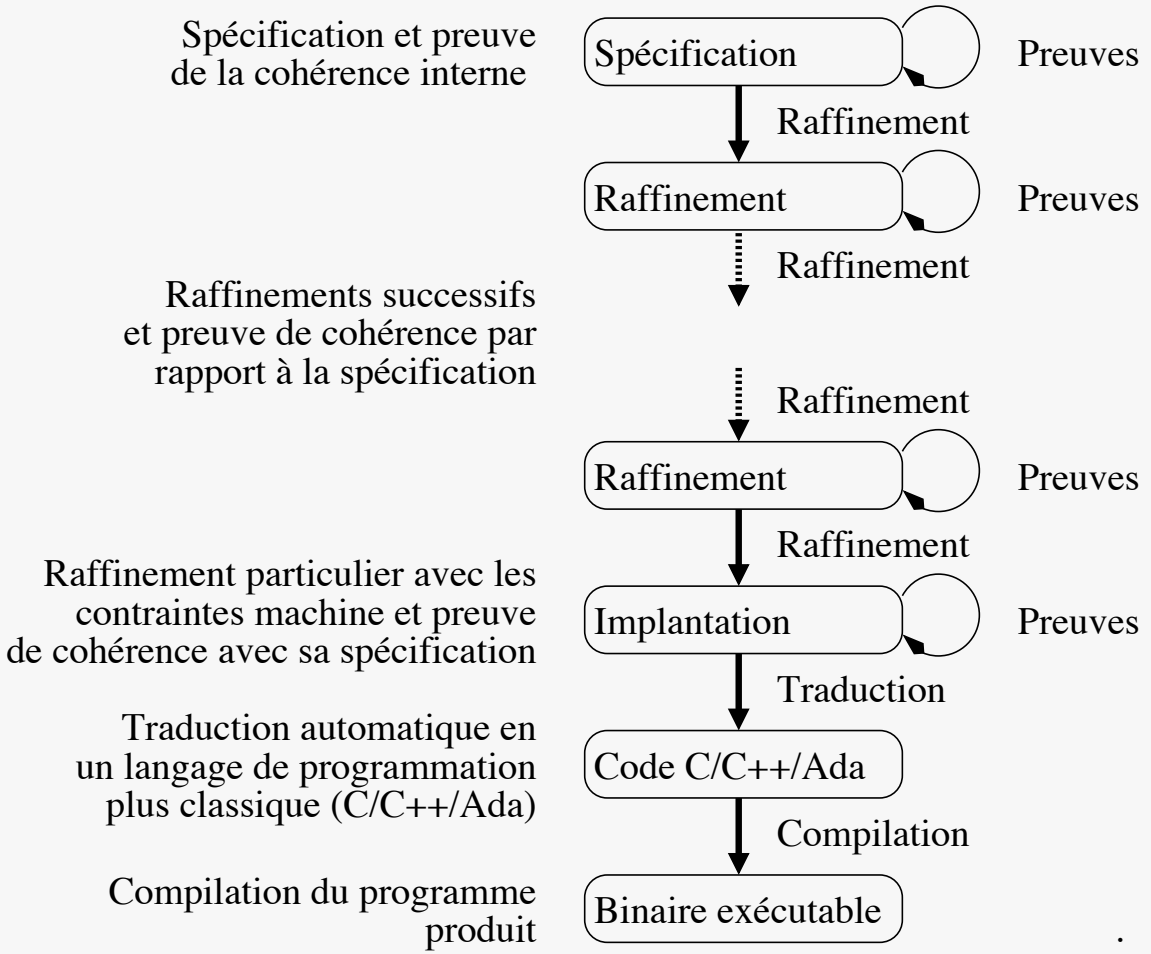
Contrainte

Contrôle externe : *on ne sait pas dans quel ordre sont appelées les opérations*

Principe

- Initialisation : *doit établir l'invariant.*
- Pour chaque opération : *si l'invariant et la pré-condition sont vrais, alors elle doit mener dans l'invariant.*

Approche *classique* d'un développement à la B



Structure d'un modèle abstrait B

MACHINE M /* *Nom du modèle* */

SETS T /* *Ensembles donnés* */
CONSTANTS C /* *Noms des constantes* */
PROPERTIES P_C /* *Propriété des constantes* */

VARIABLES V /* *Noms des variables* */
INVARIANT I /* *Propriété des variables* */
INITIALISATION init /* *Initialisation des variables* */

OPERATIONS /* *Ensemble des opérations* */

$Res \longleftarrow Op_1(Params) \mathrel{\hat{=}}$ **PRE** C_{Op_1} **THEN** S_1 **END**

...

$Res \longleftarrow Op_n(Params) \mathrel{\hat{=}}$ **PRE** C_{Op_n} **THEN** S_n **END**

END

Petite lecture introductive

Fiche TD, exercice 1

Langage des substitutions

Langage de spécification B

Caractéristiques générales

- Langage de spécification (Non-déterministe)
- Langage de programmation (sous langage : B0)
- Pas d'alias
- Instructions : appelées *substitution généralisée*

Substitutions généralisées $\Leftrightarrow$ Instructions

Nom	Substitution
skip	**skip**
Affectation	$x := E$
Conditionnelle simple	**IF** C **THEN** S_1 **ELSE** S_2 **END**
Pré-condition	**PRE** C **THEN** S **END**
Choix non-déterministe	**ANY** x **WHERE** C **THEN** S **END**
Choix borné non-déterministe	**CHOICE** S_1 **OR** S_2 **END**
Devient élément de	$x :\in E$
Séquence	$S_1 \ ; \ S_2$
Parallèle	$S_1 \parallel S_2$

Légende : x variable, E expression, C prédicat, S substitution

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
2. Comment déclare-t-on une variable locale ?

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
   - Clause **VARIABLES** du modèle
2. Comment déclare-t-on une variable locale ?

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
   - Clause **VARIABLES** du modèle
2. Comment déclare-t-on une variable locale ?
   - Substitution **ANY**

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
   - Clause **VARIABLES** du modèle
2. Comment déclare-t-on une variable locale ?
   - Substitution **ANY**
3. Comment initialise-t-on une variable globale ?

4. Comment initialise-t-on une variable locale ?

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
   - Clause **VARIABLES** du modèle
2. Comment déclare-t-on une variable locale ?
   - Substitution **ANY**
3. Comment initialise-t-on une variable globale ?
   - Clause **INITIALISATION** du modèle
4. Comment initialise-t-on une variable locale ?

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
   - Clause **VARIABLES** du modèle
2. Comment déclare-t-on une variable locale ?
   - Substitution **ANY**
3. Comment initialise-t-on une variable globale ?
   - Clause **INITIALISATION** du modèle
4. Comment initialise-t-on une variable locale ?
   - **On ne le fait pas.** On la contraint par la condition du **ANY**

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
   - Clause **VARIABLES** du modèle
2. Comment déclare-t-on une variable locale ?
   - Substitution **ANY**
3. Comment initialise-t-on une variable globale ?
   - Clause **INITIALISATION** du modèle
4. Comment initialise-t-on une variable locale ?
   - **On ne le fait pas.** On la contraint par la condition du **ANY**
5. Non-déterminisme $\Leftrightarrow$ programme flou ?

Avez vous suivi ?

1. Comment déclare-t-on une variable globale ?
   - Clause **VARIABLES** du modèle
2. Comment déclare-t-on une variable locale ?
   - Substitution **ANY**
3. Comment initialise-t-on une variable globale ?
   - Clause **INITIALISATION** du modèle
4. Comment initialise-t-on une variable locale ?
   - **On ne le fait pas.** On la contraint par la condition du **ANY**
5. Non-déterminisme $\Leftrightarrow$ programme flou ?
   - ~~Et ta mère !? Elle est floue !?~~
     Non, c'est plus proche du quantique ~~(Pas ta mère ... le non-déterminisme)~~. On s'intéresse à caractériser l'univers des possibles.

Description des données : Logique du premier ordre ensembliste

Un classement des principales logiques

Logique propositionnelle

Variables + prédicats + fonctions

Logique du premier ordre

Propositionnelle + quantification des variables ($\forall$ et $\exists$)

Logique du premier ordre ensembliste

1er ordre + ensembles

Logique d'ordre supérieur

1er ordre (ensembliste) + quantification des prédicats (et ensembles)

Logiques modales

1er ordre + opérateurs modaux ($\Box$, $\Diamond$, $\bigcirc$, ...)

Logique du premier ordre

Opérateurs logiques

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

Fonctions

$+, -, *, /$

Prédicats

$=, <, >$

Variables

Quantificateurs de variables

$\forall, \exists$

Logique du premier ordre ensembliste

Principes

- Logique du premier ordre
- Ajout de la notion d'ensemble
  - Variables de type ensemble autorisées
  - Prédicats sur les ensembles ($\in$, $\subseteq$)
  - Fonctions sur les ensembles ($\times$, $\{\ |\ \}$, $\mathbb{P}$, $\cup$, $\cap$)
    - $\times$: *Produit cartésien.*
      $A \times B$: ensemble des couples $a \mapsto b$ avec $a \in A$ et $b \in B$
    - $\{\ |\ \}$: *Définition en compréhension*
      $\{x \mid x > 4\}$: ensemble des entiers supérieurs à 4
    - $\mathbb{P}$: *ensemble des parties d'un ensemble*
      $\mathbb{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$

Rappels ensembles VS n-uplets

Principes

- Tous les éléments d'un ensemble ont le même type
  - $\mathbb{N}$: *ne contient que des entiers naturels*
- Pas d'ordre ou de doublons dans un ensemble
  - $\{1, 2, 3, 3, 1\} = \{3, 2, 1\} = \{1, 3, 2\}$
- Un ensemble peut contenir des ensembles
  - $\{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$: *ensemble des parties de* $\{1, 2\}$
- Ne pas confondre n-uplet et ensemble ! !
  - $1 \mapsto 2 \neq \{1, 2\}$: *le couple* $1 \mapsto 2$ *n'est pas l'ensemble* $\{1, 2\}$
  - $1 \mapsto 2$: *Peut aussi s'écrire* $(1, 2)$

Entrainons nous rapidement...

Quelles sont les propriétés des valeur x suivantes :

- $x \in \{p \mid p \in \mathbb{N} \land \exists y \cdot (y \in \mathbb{N} \land p = 2 * y)\}$
- $x \in \{p \mapsto q \mid p \in \mathbb{N} \land q \in \mathbb{N} \land q = 2 * p\}$
- $x \in \{p \mid \neg \exists d \cdot (d \in \mathbb{N} - \{0, 1, p\} \land p/d \in \mathbb{N})\}$

Entrainons nous rapidement...

Quelles sont les propriétés des valeur x suivantes :

- $x \in \{p \mid p \in \mathbb{N} \land \exists y \cdot (y \in \mathbb{N} \land p = 2 * y)\}$
  x est un nombre pair
- $x \in \{p \mapsto q \mid p \in \mathbb{N} \land q \in \mathbb{N} \land q = 2 * p\}$
- $x \in \{p \mid \neg \exists d \cdot (d \in \mathbb{N} - \{0, 1, p\} \land p/d \in \mathbb{N})\}$

Entrainons nous rapidement...

Quelles sont les propriétés des valeur x suivantes :

- $x \in \{p \mid p \in \mathbb{N} \wedge \exists y \cdot (y \in \mathbb{N} \wedge p = 2 * y)\}$
  x est un nombre pair
- $x \in \{p \mapsto q \mid p \in \mathbb{N} \wedge q \in \mathbb{N} \wedge q = 2 * p\}$
  x est un couple d'entiers, où le $2^{\text{ème}}$ élément est le double du 1^{er}
- $x \in \{p \mid \neg \exists d \cdot (d \in \mathbb{N} - \{0, 1, p\} \wedge p/d \in \mathbb{N})\}$

Entrainons nous rapidement...

Quelles sont les propriétés des valeur x suivantes :

- $x \in \{p \mid p \in \mathbb{N} \wedge \exists y \cdot (y \in \mathbb{N} \wedge p = 2 * y)\}$
  x est un nombre pair
- $x \in \{p \mapsto q \mid p \in \mathbb{N} \wedge q \in \mathbb{N} \wedge q = 2 * p\}$
  x est un couple d'entiers, où le $2^{\text{ème}}$ élément est le double du 1^{er}
- $x \in \{p \mid \neg\exists d \cdot (d \in \mathbb{N} - \{0, 1, p\} \wedge p/d \in \mathbb{N})\}$
  x est un nombre premier

Relations : notion fondamentale de l'ensembliste

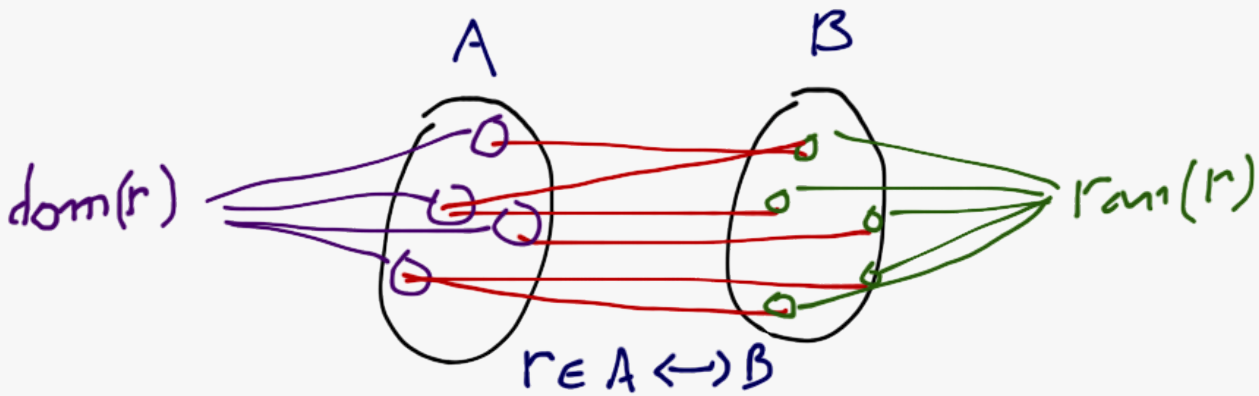
Relation générale ($\leftrightarrow$)

- Relation : *Ensemble de couples*
  - $E_1 \leftrightarrow E_2 = \{a \mapsto b \mid a \in E_1 \ \wedge \ b \in E_2\}$
- Définition d'une relation r, 3 syntaxes équivalentes :
  - $r \in E_1 \leftrightarrow E_2$
  - $r \subseteq E_1 \times E_2$
  - $r \in \mathbb{P}(E_1 \times E_2)$
- Exemple :
  - $\{(2 \mapsto 1), (4 \mapsto 6), (8 \mapsto 10)\} \in \mathbb{N} \leftrightarrow \mathbb{N}$

Relations : notion fondamentale de l'ensembliste

Identité et opérateurs

- Identité : $\mathrm{id}(\mathbb{N}) = \{(0 \mapsto 0), (1 \mapsto 1), (2 \mapsto 2), (3 \mapsto 3), \ldots\}$
- Domaine : $\mathrm{dom}(r) = \{x \mid \exists y \cdot ((x \mapsto y) \in r)\}$
- Co-domaine : $\mathrm{ran}(r) = \{y \mid \exists x \cdot ((x \mapsto y) \in r)\}$
- Image : $r[E] = \{y \mid \exists x \cdot (x \in E \ \wedge \ (x \mapsto y) \in r)\}$
- Inverse : $r^{-1} = \{y \mapsto x \mid x \mapsto y \in r\}$



Fonctions : relations particulières

Fonction ($\mapsto$ $\rightarrow$)

- Une fonction est une relation particulière :
  - Chaque antécédent a au plus 1 image
    - $r \in E_1 \leftrightarrow E_2 \wedge$
      $\forall x, y_1, y_2 \cdot ((x \mapsto y_1) \in r \wedge (x \mapsto y_2) \in r \Rightarrow y_1 = y_2)$
    - $r \in E_1 \mapsto E_2$
  - Une fonction portant sur tous les éléments de son domaine est dite **totale** ($\rightarrow$). Sinon elle est **partielle** ($\mapsto$).
    - $r \in E_1 \rightarrow E_2 \equiv (r \in E_1 \mapsto E_2 \wedge \mathrm{dom}(r) = E_1)$

Fonctions : relations particulières

Exemples et syntaxe

- Soit : $f = \{(x \mapsto y) \mid x \in \mathbb{N} \ \wedge \ y \in \mathbb{N} \ \wedge \ y = 2 * x\}$
- Alors :
  - $f \in \mathbb{N} \leftrightarrow \mathbb{N}$
  - $f \in \mathbb{N} \nrightarrow \mathbb{N}$
  - $f \in \mathbb{N} \rightarrow \mathbb{N}$
  - $f(4) = 8$
  - $(5 \mapsto 10) \in f$

Fonctions particulières

Fonctions injectives ($\rightarrowtail$ $\rightarrowtail$)

- Toute valeur du co-domaine a au plus un antécédent.
  - f^{-1} est aussi une fonction
  - $r \in E_1 \mapsto E_2 \wedge \forall x_1, x_2, y \cdot (r(x_1) = y \wedge r(x_2) = y \Rightarrow x_1 = x_2)$

Fonctions surjectives ($\twoheadrightarrow$ $\twoheadrightarrow$)

- Toute valeur du co-domaine a au moins un antécédent.
  - f^{-1} est totale *(mais est peut être une relation)*
  - $r \in E_1 \mapsto E_2 \wedge \forall y \cdot (y \in E_2 \Rightarrow \exists x \cdot r(x) = y)$

Fonctions bijectives ($\rightarrowtail\!\!\!\!\twoheadrightarrow$ $\rightarrowtail$)

- Surjective et injective

Fonctions particulières : qui a suivi ?

Exercice ....

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f = \{1 \mapsto 2, 4 \mapsto 2, 3 \mapsto 4\}$
  - Totale ou partielle ?
  - Normale, surjective, injective ou bijective ?
- $f \in \mathbb{N} \leftrightarrow \mathbb{N} - \{0\} \ \wedge \ f(x) = x + 1$
  - Totale ou partielle ?
  - Normale, surjective, injective ou bijective ?
- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f(x) = x - 1$
  - Totale ou partielle ?
  - Normale, surjective, injective ou bijective ?

Fonctions particulières : qui a suivi ?

Exercice ....

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f = \{1 \mapsto 2, 4 \mapsto 2, 3 \mapsto 4\}$ $(f \in \mathbb{N} \nrightarrow \mathbb{N})$
  partielle
  - Normale
- $f \in \mathbb{N} \leftrightarrow \mathbb{N} - \{0\} \ \wedge \ f(x) = x + 1$
  - Totale ou partielle ?
  - Normale, surjective, injective ou bijective ?
- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f(x) = x - 1$
  - Totale ou partielle ?
  - Normale, surjective, injective ou bijective ?

Fonctions particulières : qui a suivi ?

Exercice ....

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f = \{1 \mapsto 2, 4 \mapsto 2, 3 \mapsto 4\}$ $(f \in \mathbb{N} \nrightarrow \mathbb{N})$

  partielle
  - Normale

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} - \{0\} \ \wedge \ f(x) = x + 1$ $(f \in \mathbb{N} \rightarrowtail \mathbb{N})$
  - Totale

  bijective

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f(x) = x - 1$
  - Totale ou partielle ?
  - Normale, surjective, injective ou bijective ?

Fonctions particulières : qui a suivi ?

Exercice ....

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f = \{1 \mapsto 2, 4 \mapsto 2, 3 \mapsto 4\}$ $(f \in \mathbb{N} \nrightarrow \mathbb{N})$

  partielle
  - Normale

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} - \{0\} \ \wedge \ f(x) = x + 1$ $(f \in \mathbb{N} \rightarrowtail \mathbb{N})$
  - Totale

  bijective

- $f \in \mathbb{N} \leftrightarrow \mathbb{N} \ \wedge \ f(x) = x - 1$ $(f \in \mathbb{N} \rightarrowtail\!\!\!\!\!\rightarrow \mathbb{N})$

  partielle *(f(0) n'existe pas)*

  bijective

Composition de relations (ou fonctions)

Composition simple (;)

- Soit deux relations r_1 et r_2 telles que
$$r_1 \in E_1 \leftrightarrow E_2 \ \wedge \ r_2 \in E_2 \leftrightarrow E_3$$
- $r_1 \ ; \ r_2$: composition de chaque paire de couples ayant l'élément central commun *(en maths : $r_1 \circ r_2$)*

$$\{(1 \mapsto 4), (2 \mapsto 5), (2 \mapsto 4)\} \ ; \ \{(4 \mapsto 8), (6 \mapsto 9)\} = \{(1 \mapsto 8), (2 \mapsto 8)\}$$

Fermetures (soit $r \mathrel{\widehat{=}} E \leftrightarrow E$)

- Itération : $r^0 \mathrel{\widehat{=}} \mathrm{id}(E)$ $\qquad r^n \mathrel{\widehat{=}} r \ ; \ r^{n-1}$
- Transitive : $r^+ \mathrel{\widehat{=}} r \ \cup \ r;r \ \cup \ r;r;r \ \cup \ r;r;r;r \ldots$

$\equiv \ r^+ \mathrel{\widehat{=}} \bigcup_{n=1}^{\infty} r^n$

- Réflexive transitive : $r^* \mathrel{\widehat{=}} \mathrm{id}(E) \cup r^+$

Exemple de modélisation

Modélisation d'un parking

- Soit BADGES, CLIENTS et PLACES trois ensembles
- Les abonnés sont des clients particuliers
- Chaque badge appartient à un unique abonné :
- Certaines places sont réservées :
- Plan d'occupation des places :
- Respect des réservations :

Exemple de modélisation

Modélisation d'un parking

- Soit BADGES, CLIENTS et PLACES trois ensembles
- Les abonnés sont des clients particuliers
  - $ABONNES \subseteq CLIENTS$
- Chaque badge appartient à un unique abonné :
- Certaines places sont réservées :
- Plan d'occupation des places :
- Respect des réservations :

Exemple de modélisation

Modélisation d'un parking

- Soit BADGES, CLIENTS et PLACES trois ensembles
- Les abonnés sont des clients particuliers
  - $ABONNES \subseteq CLIENTS$
- Chaque badge appartient à un unique abonné :
  - $badges \in BADGES \twoheadrightarrow ABONNES$ *(Surjection totale)*
- Certaines places sont réservées :
- Plan d'occupation des places :
- Respect des réservations :

Exemple de modélisation

Modélisation d'un parking

- Soit BADGES, CLIENTS et PLACES trois ensembles
- Les abonnés sont des clients particuliers
  - $ABONNES \subseteq CLIENTS$
- Chaque badge appartient à un unique abonné :
  - $badges \in BADGES \twoheadrightarrow ABONNES$ *(Surjection totale)*
- Certaines places sont réservées :
  - $reservees \in PLACES \twoheadrightarrow ABONNES$ *(Surjection partielle)*
- Plan d'occupation des places :
- Respect des réservations :

Exemple de modélisation

Modélisation d'un parking

- Soit BADGES, CLIENTS et PLACES trois ensembles
- Les abonnés sont des clients particuliers
  - $ABONNES \subseteq CLIENTS$
- Chaque badge appartient à un unique abonné :
  - $badges \in BADGES \twoheadrightarrow ABONNES$ *(Surjection totale)*
- Certaines places sont réservées :
  - $reservees \in PLACES \twoheadrightarrow ABONNES$ *(Surjection partielle)*
- Plan d'occupation des places :
  - $occupees \in PLACES \mapsto CLIENTS$ *(Fonction partielle)*
- Respect des réservations :

Exemple de modélisation

Modélisation d'un parking

- Soit BADGES, CLIENTS et PLACES trois ensembles
- Les abonnés sont des clients particuliers
  - $ABONNES \subseteq CLIENTS$
- Chaque badge appartient à un unique abonné :
  - $badges \in BADGES \twoheadrightarrow ABONNES$ *(Surjection totale)*
- Certaines places sont réservées :
  - $reservees \in PLACES \twoheadrightarrow ABONNES$ *(Surjection partielle)*
- Plan d'occupation des places :
  - $occupees \in PLACES \mapsto CLIENTS$ *(Fonction partielle)*
- Respect des réservations :
  - $\forall p \cdot (p \in \mathrm{dom}(occupees) \cap \mathrm{dom}(reservee) \Rightarrow occupees(p) = reservees(p))$

S'il reste du temps en fin de TD1 ... (qui a ri ! ?)

Fiche TD, exercice 2

Sémantique des substitutions généralisées

Rappel : Que cherche-t-on à vérifier ?

Idées générale

- Objectif : *démontrer qu'il n'existe pas de séquence d'appels des opérations permettant de violer l'invariant.*
- Invariant : *condition décrivant les états autorisés*

Contrainte

Contrôle externe : *on ne sait pas dans quel ordre sont appelées les opérations*

Principe

- Initialisation : *doit établir l'invariant.*
- Pour chaque opération : *si l'invariant et la pré-condition sont vrais, alors elle doit mener dans l'invariant.*

Logique de Hoare (1969)

- Permet de raisonner sur les programmes

Triplet de Hoare

- Soit le triplet de Hoare : $\{P_1\}\ i\ \{P_2\}$
  - i est une instruction
  - P_1 est un prédicat appelé pré-condition
  - P_2 est un prédicat appelé post-condition
- Interprétation : *Si P_1 est vrai alors P_2 doit être vrai après exécution de I*

De la logique de Hoare à la preuve

Triplet de Hoare

- Soit le triplet de Hoare : $\{P_1\}\ i\ \{P_2\}$
  - i est une instruction
  - P_1 est un prédicat appelé pré-condition
  - P_2 est un prédicat appelé post-condition

Problèmes

- D'où viennent P_1 et P_2 ?
- Comment calculer si la propriété décrite est vraie ?

De la logique de Hoare à la preuve

Triplet de Hoare

- Soit le triplet de Hoare : $\{P_1\}\ i\ \{P_2\}$
  - i est une instruction
  - P_1 est un prédicat appelé pré-condition
  - P_2 est un prédicat appelé post-condition

Problèmes

- D'où viennent P_1 et P_2 ? Invariant et clause **PRE**
- Comment calculer si la propriété décrite est vraie ?

De la logique de Hoare à la preuve

Triplet de Hoare

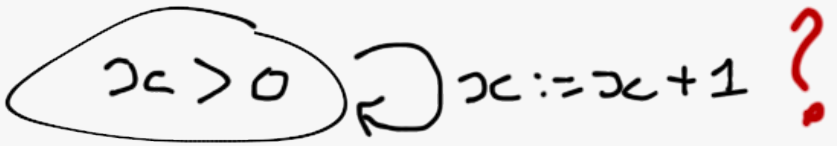
- Soit le triplet de Hoare : $\{P_1\}\ i\ \{P_2\}$
  - i est une instruction
  - P_1 est un prédicat appelé pré-condition
  - P_2 est un prédicat appelé post-condition

Problèmes

- D'où viennent P_1 et P_2 ? Invariant et clause **PRE**
- Comment calculer si la propriété décrite est vraie ?
  - $P_1[i]$: calcul de la plus forte post-condition
    ou
  - $[i]P_2$: calcul de la plus faible pré-condition

Weakest Precondition VS Strongest Postcondition ?

Est il vrai que :

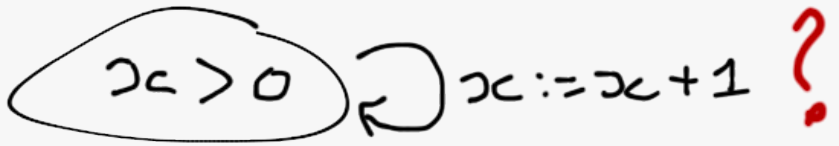


SP (en avant $\rightarrow$ dur $\rightarrow$ à éviter)

- Si $x > 0$ toute exécution de $x := x + 1$ établit elle $x > 0$?
- Se décompose en $(P_1[S] \Rightarrow P_2)$:
  - Sachant $x > 0$, quel état atteint on par $x := x + 1$?
  - $x > 0$ l'inclue-t-il ?

Weakest Precondition VS Strongest Postcondition ?

Est il vrai que :

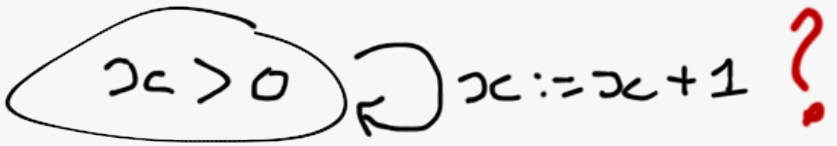


SP (en avant → dur → à éviter)

- Si $x > 0$ toute exécution de $x := x + 1$ établit elle $x > 0$?
- Se décompose en $(P_1[S] \Rightarrow P_2)$:
  - Sachant $x > 0$, quel état atteint on par $x := x + 1$?
  - $x > 0$ l'inclue-t-il ?
- Définition de la SP de l'affectation :
  $$P[x := E] \quad \widehat{=} \quad \exists x' \cdot (P(x'/x) \ \wedge \ x = E(x'/x))$$
- À démontrer : $\exists x' \cdot (x' > 0 \ \wedge \ x = x' + 1) \Rightarrow (x > 0)$

Weakest Precondition VS Strongest Postcondition ?

Est il vrai que :

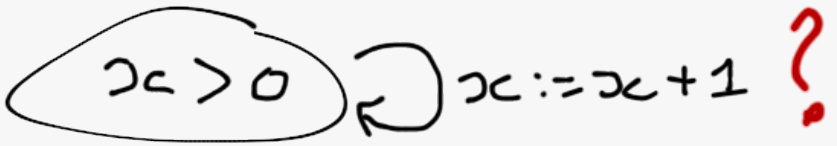


WP (en arrière → facile → à privilégier)

- Si $x > 0$ toute exécution de $x := x + 1$ établit elle $x > 0$?
- Se décompose en $(P_1 \Rightarrow [S]P_2)$:
  - Depuis quel état $x := x + 1$ atteint toujours $x > 0$?
  - Cela inclue-t-il $x > 0$?

Weakest Precondition VS Strongest Postcondition ?

Est il vrai que :



WP (en arrière → facile → à privilégier)

- Si $x > 0$ toute exécution de $x := x + 1$ établit elle $x > 0$?
- Se décompose en $(P_1 \Rightarrow [S]P_2)$:
  - Depuis quel état $x := x + 1$ atteint toujours $x > 0$?
  - Cela inclue-t-il $x > 0$?
- Définition du WP de l'affectation :
  $$[x := E]P \quad \hat{=} \quad P(E/x)$$
- À démontrer : $(x > 0) \Rightarrow (x + 1 > 0)$

Sémantique des substitutions : calcul du WP

Sémantique axiomatique : plus faible pré-condition

- Weakest Precondition (WP) : Dijkstra, 1976
- Notation : $[S]P$

Definition (Calcul du WP)

$[S]P$ est la plus faible pré-condition telle que, si elle est vérifiée, alors l'exécution de S termine et établie la propriété P

Terminaison

Cas possibles de non-terminaison :

- Pré-condition non respectée lors d'un appel d'opération
- Boucle infinie (à l'implémentation uniquement).

Calcul du WP 1/3 : cas de l'affectation

Axiome du calcul du WP

- $[x := E]P$ est le remplacement de chaque occurrence libre de x par E dans P

Dans une formule, une variable x est

libre s'il existe une occurrence de x non quantifiée

liée si toute occurrence de x est sous un quantificateur

non-libre si x y est lié ou n'y apparait pas.

Example

- $[x := t]x > 0 \quad \equiv \quad t > 0$ *x libre dans* $x > 0$
- $[x := t]\exists x \cdot (x > y) \quad \equiv \quad \exists x \cdot (x > y)$ *x lié dans* $\exists x \cdot (x > t)$

Calcul du WP 2/3 : substitutions généralisées

[S]P		Réduction
[**skip**]P	$\equiv$	P
$[x := E]P$	$\equiv$	P *où les occurrences libres de x sont remplacées par E*
[**PRE** C **THEN** S **END**]P	$\equiv$	$C \wedge [S]P$
[**ANY** x **WHERE** C **THEN** S **END**]P	$\equiv$	$\forall x \cdot (C \Rightarrow [S]P)$ *si x non libre dans P*
[**CHOICE** S_1 **OR** S_2 **END**]P	$\equiv$	$[S_1]P \wedge [S_2]P$
[**IF** C **THEN** S_1 **ELSE** S_2 **END**]P	$\equiv$	$(C \Rightarrow [S_1]P) \wedge (\neg C \Rightarrow [S_2]P)$
$[x :\in E]P$	$\equiv$	$\forall y \cdot (y \in E \Longrightarrow [x := y]P)$

Calcul du WP 3/3 : connecteurs

Substitution séquence $S_1 \ ; \ S_2$

- $[S_1 \ ; \ S_2]P \mathrel{\hat{=}} [S_1]([S_2]P)$
- $[S_1]([S_2]P) \equiv [S_1][S_2]P$

Substitution simultanée $S_1 \ || \ S_2$

- S_1 et S_2 ne doivent pas modifier les mêmes variables
- $[x := E_1 \ || \ y := E_2]P \mathrel{\hat{=}} [z := E_2][x := E_1][y := z]P$

Si z non libre dans E_1, E_2 et P.

Exemple d'une obligation de preuve

OP : Est ce qu'une action préserve un invariant ?

- Soit $S \mathrel{\hat{=}} t := x + 1; x := t + 1$ l'action à étudier
- Soit $I \mathrel{\hat{=}} x > 0$ notre invariant
- Montrons que I est une condition suffisante pour que l'exécution de S garantisse I

Est il vrai que $x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0$?

Exemple d'une obligation de preuve

OP : Est ce qu'une action préserve un invariant ?

- Soit $S \mathrel{\hat{=}} t := x + 1; x := t + 1$ l'action à étudier
- Soit $I \mathrel{\hat{=}} x > 0$ notre invariant
- Montrons que I est une condition suffisante pour que l'exécution de S garantisse I

Est il vrai que $x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0$?

$$x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0$$

Exemple d'une obligation de preuve

OP : Est ce qu'une action préserve un invariant ?

- Soit $S \mathrel{\hat{=}} t := x + 1; x := t + 1$ l'action à étudier
- Soit $I \mathrel{\hat{=}} x > 0$ notre invariant
- Montrons que I est une condition suffisante pour que l'exécution de S garantisse I

Est il vrai que $x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0$?

$$x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0$$
$$\equiv x > 0 \Rightarrow [t := x + 1][x := t + 1]x > 0$$

Exemple d'une obligation de preuve

OP : Est ce qu'une action préserve un invariant ?

- Soit $S \mathrel{\hat{=}} t := x + 1; x := t + 1$ l'action à étudier
- Soit $I \mathrel{\hat{=}} x > 0$ notre invariant
- Montrons que I est une condition suffisante pour que l'exécution de S garantisse I

Est il vrai que $x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0$?

$$
\begin{aligned}
& x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0 \\
\equiv\ & x > 0 \Rightarrow [t := x + 1][x := t + 1]x > 0 \\
\equiv\ & x > 0 \Rightarrow [t := x + 1]t + 1 > 0
\end{aligned}
$$

Exemple d'une obligation de preuve

OP : Est ce qu'une action préserve un invariant ?

- Soit $S \mathrel{\hat{=}} t := x + 1; x := t + 1$ l'action à étudier
- Soit $I \mathrel{\hat{=}} x > 0$ notre invariant
- Montrons que I est une condition suffisante pour que l'exécution de S garantisse I

Est il vrai que $x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0$?

$$
\begin{aligned}
& x > 0 \Rightarrow [t := x + 1; x := t + 1]x > 0 \\
\equiv\ & x > 0 \Rightarrow [t := x + 1][x := t + 1]x > 0 \\
\equiv\ & x > 0 \Rightarrow [t := x + 1]t + 1 > 0 \\
\equiv\ & x > 0 \Rightarrow x + 2 > 0 \\
\equiv\ & \text{true}
\end{aligned}
$$

Mise en pratique

Fiche TD, exercice 3

Modèles B : description et génération d'obligations de preuves

Structure d'un modèle abstrait B

MACHINE M /* *Nom du modèle* */

SETS T /* *Ensembles donnés* */
CONSTANTS C /* *Noms des constantes* */
PROPERTIES P_C /* *Propriété des constantes* */

VARIABLES V /* *Noms des variables* */
INVARIANT I /* *Propriété des variables* */
INITIALISATION init /* *Initialisation des variables* */

OPERATIONS /* *Ensemble des opérations* */
$Res \longleftarrow Op_1(Params) \mathrel{\hat{=}}$ **PRE** C_{Op_1} **THEN** S_1 **END**
...
$Res \longleftarrow Op_n(Params) \mathrel{\hat{=}}$ **PRE** C_{Op_n} **THEN** S_n **END**
END

Conditions de validité d'un système B

L'initialisation établit l'invariant

$$T \wedge P_C \Rightarrow [init]I$$

Chaque opération préserve l'invariant

- Étant donnée une opération de la forme :

  $Op \mathrel{\hat{=}}$ **PRE** C_{Op} **THEN** S **END**

- Il suffit de vérifier que :

$$T \wedge P_C \wedge I \wedge C_{Op} \Rightarrow [S]I$$

Mise en pratique

Fiche TD, exercices 4, 5 et 6

Le raffinement B

Le raffinement : kesako ?

Description de plus en plus fine du système

- Exemple :

Abstraction	Opération *O* modifie 1 tableau *T*
Raffinement 1	Le tableau modifié par *O* est trié
Raffinement 2	*T* contient les mêmes valeurs avant et après modification
Raffinement 3	Description d'un algorithme de tri

- **Remplacer une opération par son raffinement ne doit pas être perceptible de l'extérieur**

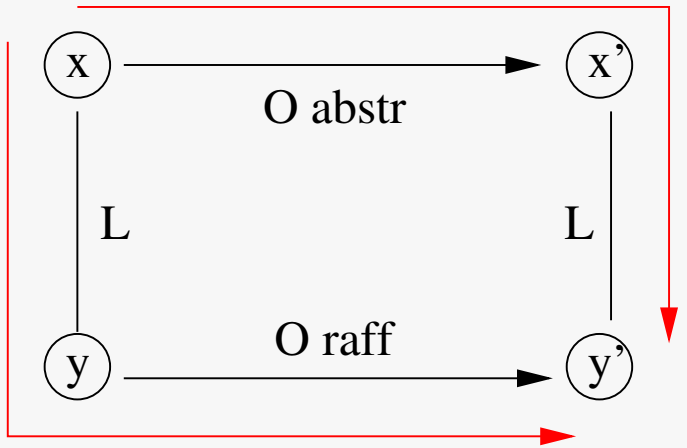
Le raffinement : kesako ?

Description de plus en plus fine du système

- Les paramètres acceptés par l'abstraction sont acceptés par le raffinement
  $(pre_{\text{abstr}} \Rightarrow pre_{\text{raff}})$
- Les modifications faites par le raffinement sont prévues dans l'abstraction
  $(post_{\text{raff}} \Rightarrow post_{\text{abstr}})$

Raffinement en B

Raffinement en B : L-Simulation



$$\forall x, y, y' \cdot \left(\begin{array}{l} (x, y) \in L \quad \wedge \quad (y, y') \in O_{raff} \\ \Rightarrow \exists x' \cdot ((x, x') \in O_{abstr} \quad \wedge \quad (x', y') \in L) \end{array} \right)$$

En B : $L \mathrel{\widehat{=}} I_{raff} \wedge I_{abstr}$

Raffinement d'une substitution (1/2)

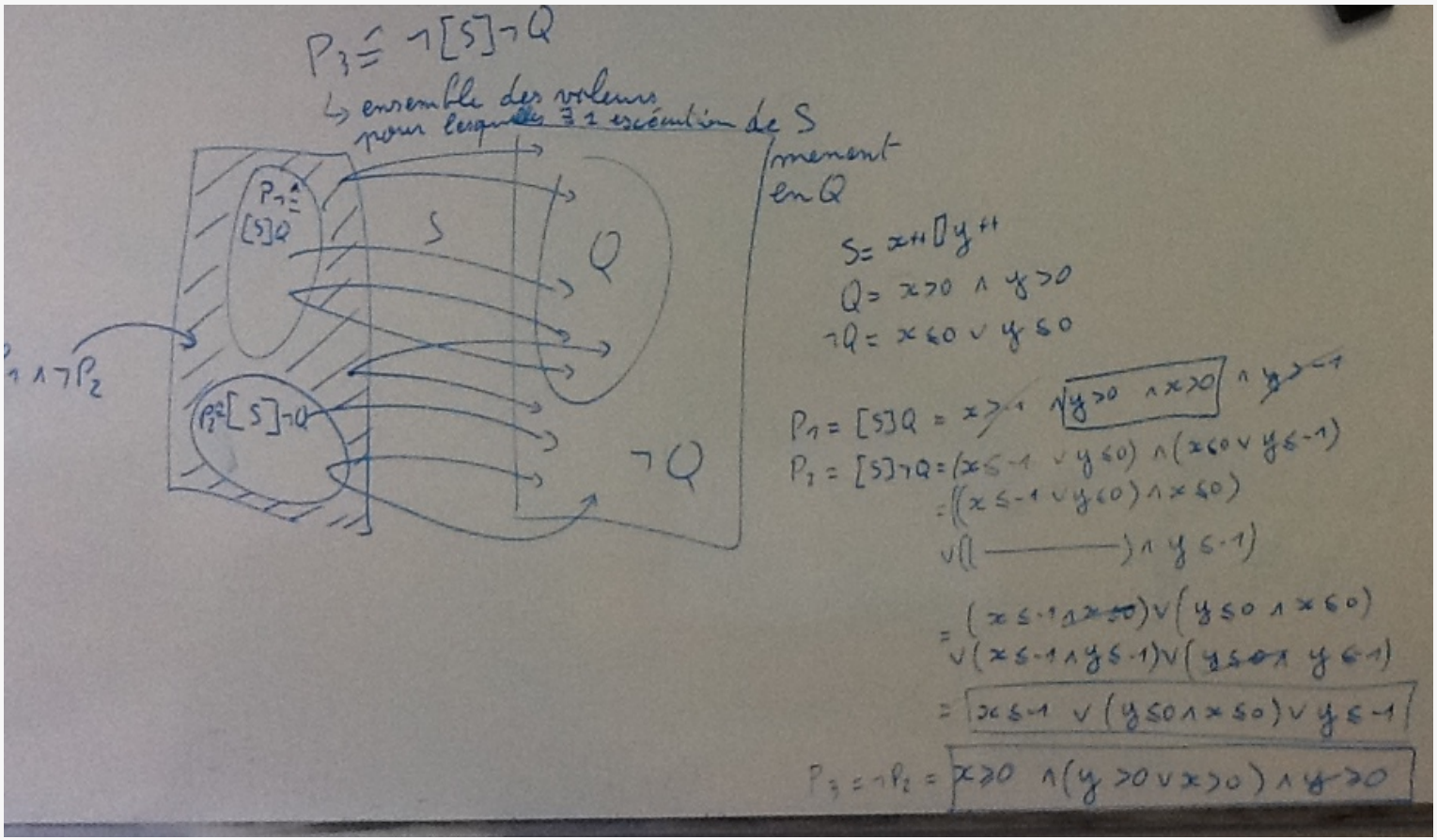
Pour une substitution

- S_R raffine S_A ssi :

$$L \Rightarrow [S_R]\neg[S_A]\neg L$$

- avec :
  - S_R et S_A portant sur des espaces de variables différents
  - L l'invariant de liaison

Raffinement d'une substitution (2/2)



S_R raffine S_A ssi : $L \Rightarrow [S_R]\neg[S_A]\neg L$

Manipulation

Exercice 7.1 de la feuille de TD

Structure d'un raffinement B

REFINEMENT N /* Nom du raffinement */
REFINES M /* Nom du modèle raffiné */

SETS T_R /* Ensembles donnés */
CONSTANTS C_R /* Noms des constantes */
PROPERTIES P_{C_R} /* Propriété des constantes */

VARIABLES V_R /* Noms des variables */
INVARIANT I_R /* Propriété des variables */
INITIALISATION $init_R$ /* Initialisation des variables */

OPERATIONS /* Ensemble des opérations */

$Res \longleftarrow Op_1(Params) \mathrel{\hat{=}}$ **PRE** $C_{Op_{1,R}}$ **THEN** S_{1_R} **END**

...

$Res \longleftarrow Op_n(Params) \mathrel{\hat{=}}$ **PRE** $G_{Op_{n,R}}$ **THEN** S_{n_R} **END**

END

Obligations de preuve du raffinement en B

Pour un module

Initialisation	$P_C \wedge P_{C_R} \Rightarrow [Init_R]\neg[Init_A]\neg I_R$
Pré-condition	$P_C \wedge P_{C_R} \wedge L \wedge C_{Op} \Rightarrow C_{Op_R}$
Opé. sans résultat	$P_C \wedge P_{C_R} \wedge L \wedge C_{Op} \Rightarrow [S_R]\neg[S_A]\neg I_R$
Opé. avec résultat	$P_{C_R} \wedge L \wedge C_{Op} \Rightarrow [[Res := Res_R]S_R]\neg[S_A]\neg(I_R \wedge Res = Res_R)$ *(Si Res_R non libre dans Res, S_R, S_A, I_R)*

Exemple de raffinement

Cf fiche d'accompagnement

Mise en pratique

Fiche TD, exercice 7.2

Bilan

Bilan sur la méthode B

- Permet l'écriture de bibliothèques sûres
- Propriétés exprimables : invariants (ex : pas de temporel)
- Usages principaux :
  - Développement correct par construction
  - Oracle pour la génération de cas de tests
  - Support de réflexion avant développement
- Très coûteux en temps et en expérience
  - SAT : problème NP-Complet
  - Expérience : nécessaire pour formuler plus efficacement les modèles

Bilan sur les méthodes formelles

- Méthodes de vérification : toujours très coûteuses
  - Preuve : problème SAT
  - Interprétation abstraite : choix de la connexion de Gallois
  - Model Checking : choix de la factorisation de l'espace d'état

  Mais les garanties obtenues à la hauteur de leur coût.
- À tout moment dans le cycle de développement
  - Définition des besoins
  - Développement
  - Vérification
- Méthodes formelles :
  - vous les voulez *(pour votre CB, votre voiture, votre avion,...)*
  - feront parti de votre vie d'ingénieur
  - nécessitent d'y être préparé

That's all folks !

Annexe : Complexité de la preuve de programme ?

SAT : problème NP-complet

- Principe :
  - Soit une formule φ en CNF *(forme normale conjonctive)*.
  - φ est satisfaisable s'il existe une valuation des variables qui la rend vraie.
  - ordre de φ : *Nombre de clauses par disjonction*
  - Déterminer si une formule CNF d'ordre *n* est satisfaisable est appelé problème SAT d'ordre n (n-SAT).
- Exemple de formule 3-SAT :

$$\varphi = (v_1 \vee v_2 \vee \neg v_3) \wedge (v_1 \vee \neg v_4 \vee v_5)$$

- Problème NP-Complet :
  - NP : *On peut résoudre le problème en temps polynomial sur une machine non-déterministe*
  - Complet : *Tout problème NP est réductible en ce problème.*