

Introduction à la modélisation formelle

Abstraction - Modélisation

J. Christian Attiogbé

Université de Nantes

Septembre 2008, maj 2011

Objectifs

Acquisition de **concepts**, **méthodes** et **outils** permettant une mise en œuvre rigoureuse des systèmes informatiques

Cette mise en œuvre va :
de l'**énoncé des besoins** (**cahier de charges**) de l'utilisateur
à une spécification opérationnelle,
en vue d'aboutir à l'installation d'un **logiciel**
conforme à la spécification extraite des besoins.

Modélisation formelle : où se situe-t-on ?

$$Informatique = \begin{cases} \text{Software (Génie logiciel) : construire des logiciels} \\ \text{Hardware : construire des matériels (ordinateurs)} \end{cases}$$

On **modélise ce qu'on va construire**, les données et des traitements

Modélisation : une étape d'abstraction/conceptualisation en amont de la programmation.

Quel peut être le modèle d'un logiciel ?

Quels intérêts pour la modélisation

Pour **construire**, on a besoin de :

- Outils
- Méthodes
- Techniques
- et plus spécifiquement de **langages**,
d'**astuces/recettes/techniques**.

Plus directement, quotidiennement,

- **travail en amont de** l'algorithmique et de la programmation ;
- **structuration** des systèmes d'information, des bases de données ;
- **analyse** des systèmes informatisés
- ...

Plan de ce chapitre

- 1 Modélisation/Spécification formelle
 - Modélisation
 - Exemples de modèles
 - Abstraction - Modélisation
 - Abstraction - Modélisation
- 2 Modélisation logique
 - Exemples de modèle
- 3 Modélisation ensembliste
 - Exemples de modèle
- 4 Logique et raisonnement
- 5 Logique
 - Calcul des propositions
 - Calcul des prédicats
- 6 Opérations en logique
- 7 Théorie des ensembles
 - Ensembles, relations
 - Ensembles, relations, fonctions
- 8 Analyse des besoins - Abstraction, propriétés
- 9 A savoir



Modélisation formelle

En Sciences et Techniques,

on **construit** à partir de **modèles (abstrait, formels,...)** :

Les modèles sont *imaginés/conçus* par les techniciens/ingénieurs pour **représenter le réel** ; mais cela n'en est qu'une **abstraction**.

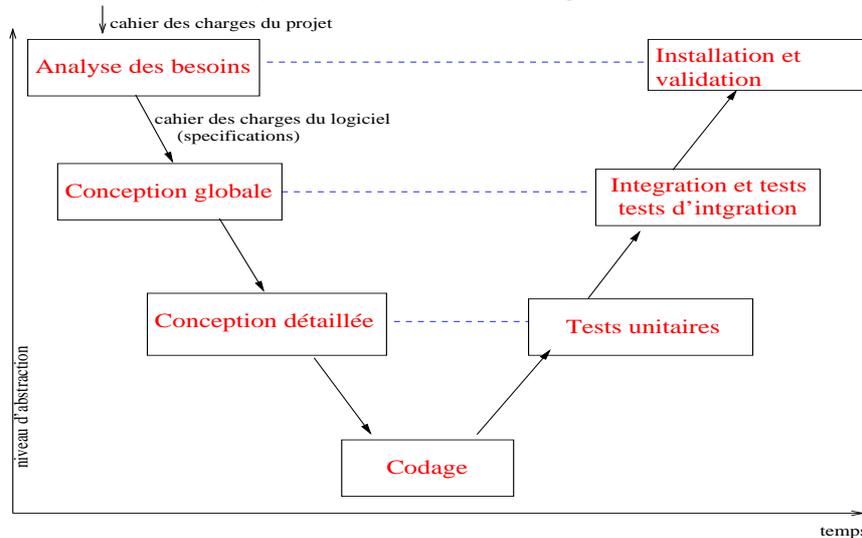
- bâtiment (plans de l'architecte, ...)
- génie civil (plans, spécifications techniques ...)
- génie électrique (plans ...)
- **génie logiciel** (modèles formels)
- ...



Modélisation : où se situe-t-on ?

Informatique > Génie logiciel > construction des logiciels

Cycle de vie du logiciel



Modélisation : suite de la phase d'analyse.

Modélisation formelle: où se situe-t-on ?

- *Hardware* (matériel)
- *Software* (logiciel)
- Système d'exploitation (*Operating system*)
- Ordinateur = matériel + logiciel de base [+ logiciel d'application]
- Système informatique
- Système d'information

Analyse et Conception des Systèmes Informatiques

Analyse des besoins et modélisation : *Requirement Analysis*

Modèle des données, modèles des processus, interaction

Modélisation formelle(suite)

On **raisonne** à partir de **modèles (abstraits, formels,...)** :
soit d'objets existants ou de phénomènes naturels observés,
soit d'objets futurs qu'on veut construire ;

- bâtiment (plans de l'architecte, ...)
- génie civil (plans, spécifications techniques ...)
- génie électrique (plans ...)
- **génie logiciel** (modèles formels)
- ...

☞ Besoin de **concepts** pour **modéliser rigoureusement**

Modélisation

Modélisation :

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

Il y a deux principales notions de modèles (en informatique).

- 1 **Modèle = une approximation de la réalité par une structure mathématique.**
- 2 Un objet O est modèle d'une réalité R , si O permet de répondre aux questions que l'on se pose sur R .

En **Mathématique, Physique, ...**: **systèmes d'équations portant sur des grandeurs (masses, énergie, ...) ou des lois hypothétiques.**

Modélisation

$$f(x) = \begin{cases} x < 0 & y = -x \\ x \geq 0 & y = x + 4 \end{cases}$$

$$f : \text{Entier} \rightarrow \text{Entier}$$

$$x \mapsto y$$

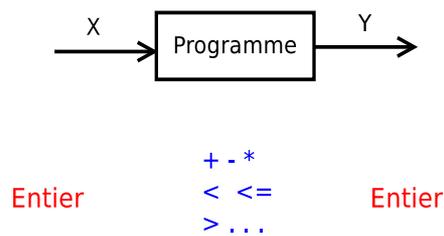


Figure: Idée d'un modèle

Modélisation : exemple

Etat d'occupation d'une salle

Construire un logiciel qui contrôle l'affichage d'un panneau en fonction de l'utilisation des places dans une salle.

Sur le panneau, toutes les places apparaissent avec :

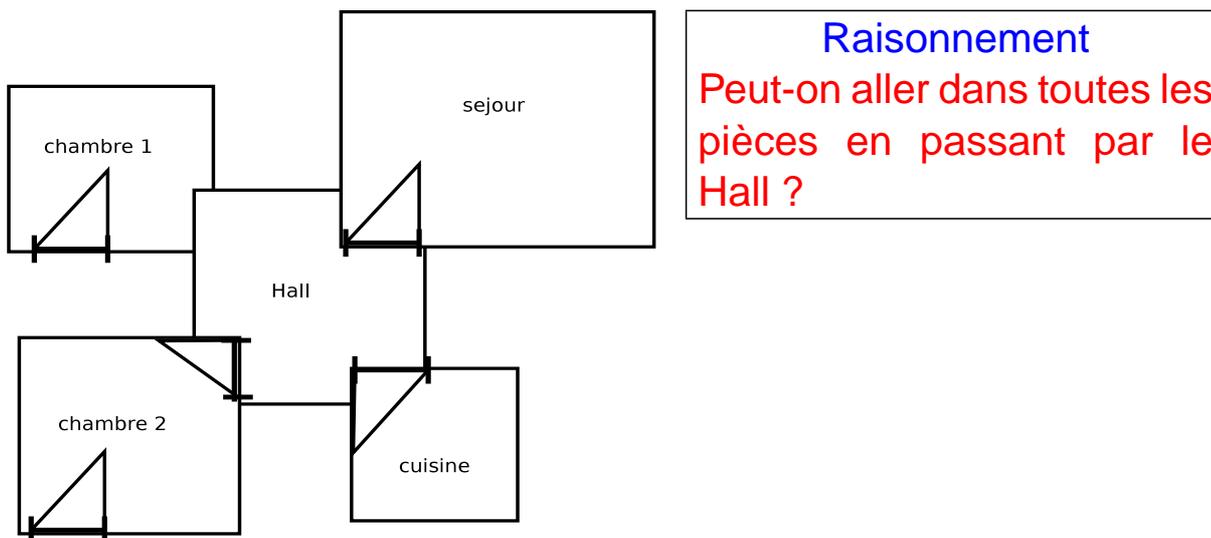
- un point lumineux vert quand la place est libre,
- un point lumineux rouge quand la place est occupée.

Comment faire pour construire ce logiciel ?

Quelles sont les données, les traitements ? les propriétés ?

Analyse du besoin > Abstraction > Conception > Codage > ...

Architecture



Raisonnement
 Peut-on aller dans toutes les pièces en passant par le Hall ?

Figure: Plan d'une maison

Génie Civil

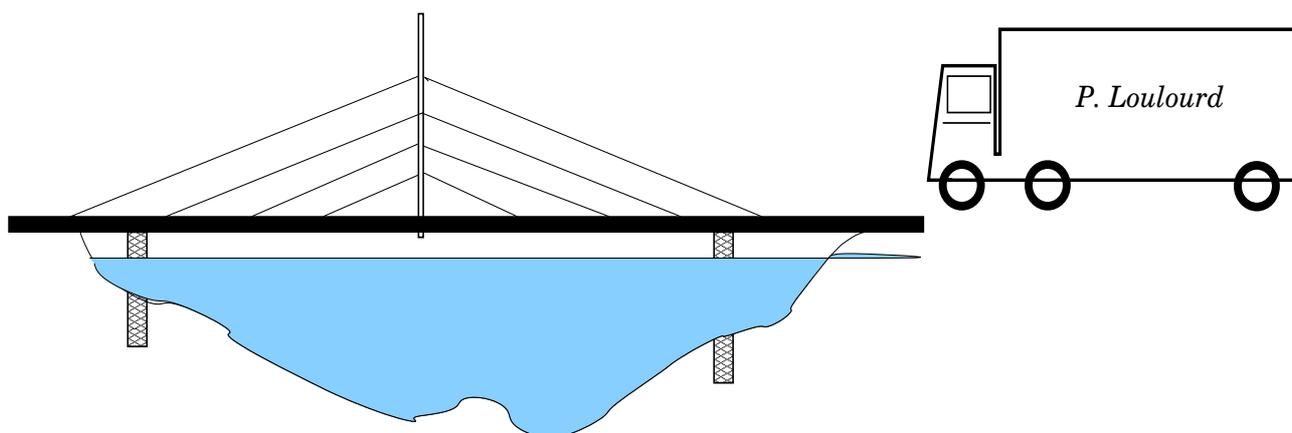


Figure: Schéma/modèle d'un pont

Est-ce que le pont peut supporter le poids des véhicules de plus de 40 tonnes ?

Modélisation à l'aide de graphe

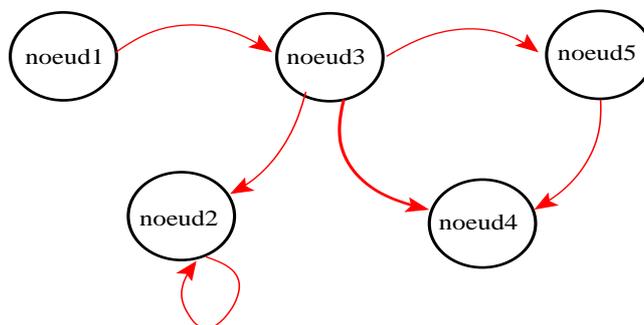


Figure: Modèle sous forme de graphe

Est-ce que le graphe est connexe ?

Modélisation à l'aide de graphe

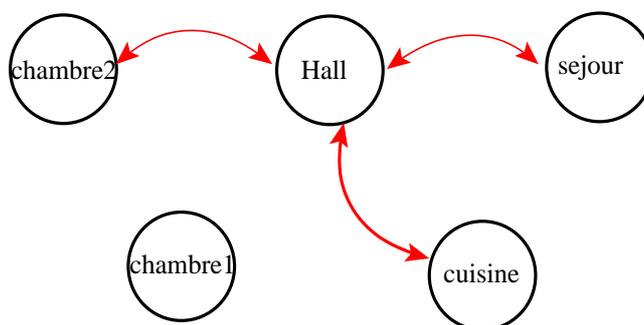


Figure: Modèle d'une maison à l'aide de graphe

Est-ce qu'on peut aller du Hall vers toutes les chambres ?

Construction du logiciel (programmes)

Place de la modélisation (spécification) dans la démarche de construction de programmes/logiciels.

Puis

- les autres étapes de la construction,
- les tâches (ou activités) et les rôles (des acteurs).

Construction du logiciel (programmes)

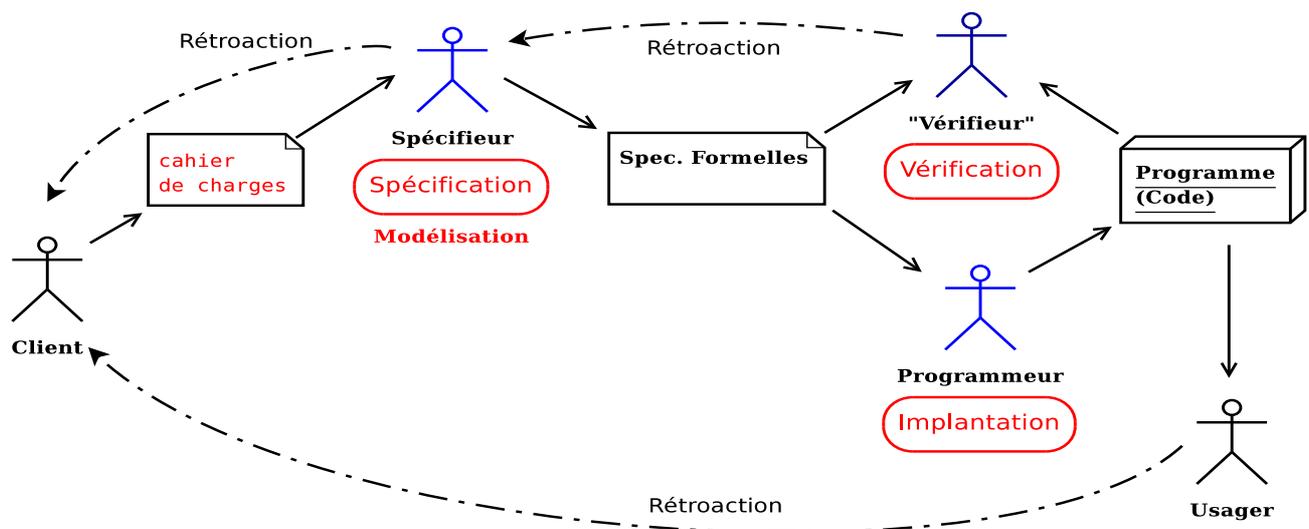


Figure: Aperçu du cycle de construction du logiciel

Abstraire, c'est quoi ?

Abstraire : Oter les détails pour ne retenir que la principale substance/caractéristique/propriété qui donne du sens à ...

Abstraction : la principale façon de construire des modèles
= structurer des données et les contraindre par des propriétés.

Abstraction

- Soit à manipuler une chaise dans un programme (logiciel).
- Comment la modéliser (la représenter) ?
- **Qu'est-ce qu'une chaise ?**

Abstraction

Qu'est-ce que c'est ?

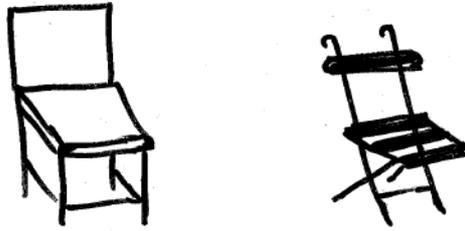


Figure: Dessins (abstrait) de chaises

Abstraction

Qu'est-ce que c'est ?



Figure: Dessins (abstrait) de chaises

Abstraction

Qu'est-ce que c'est ?

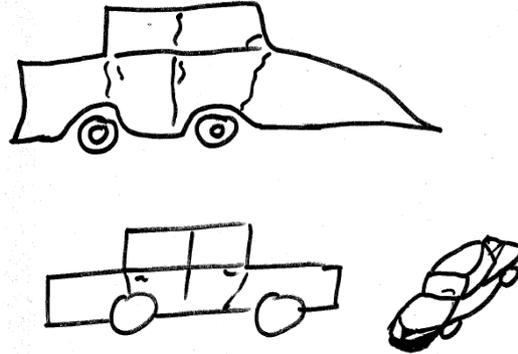


Figure: Dessins (abstrait) de voitures

Abstraction

Qu'est-ce que c'est ?

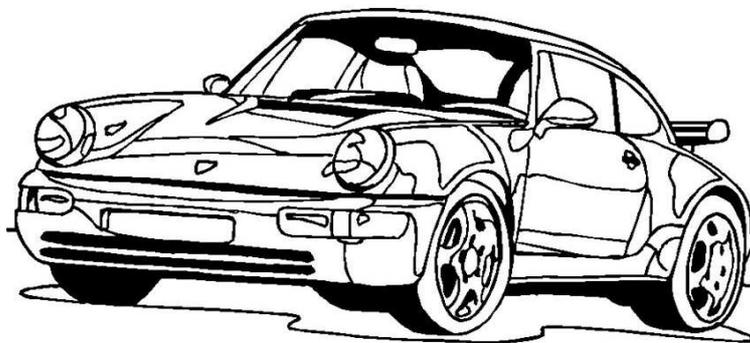


Figure: Dessin de voiture

Abstraction

Attention, il faut néanmoins avoir suffisamment d'information !

Qu'est-ce que c'est ?

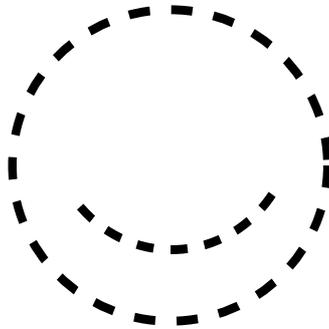


Figure: quoi ?

Abstraction

Attention, il faut néanmoins avoir suffisamment d'information !

Qu'est-ce que c'est ?

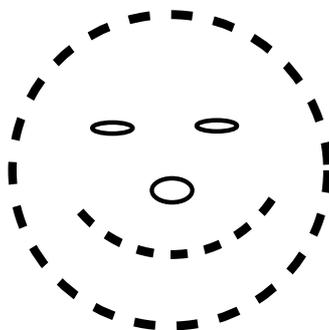


Figure: quoi ?

Différentes techniques d'abstraction

Abstractions orientées Propriétés

Abstraction orientée propriétés : une spécification qui met l'accent sur des propriétés (logiques, algébriques) qui représentent l'objet d'étude.

Abstractions orientées Modèles

Abstraction orientée modèles : une spécification qui met l'accent sur des structures mathématiques (telles que ensembles, relations, produits cartésiens, fonctions, etc) qui représentent l'objet d'étude.

Et aussi : **catégorisation, instanciation, composition**

Modélisation logique

- Soient les propositions :
 - P_1 x n'a pas le forfait F
 - P_2 x a plus de 2 ans d'ancienneté
 - P_3 x a 10 des services de la liste L

Que représente :

$$(P_1 \wedge P_2) \vee P_3 ?$$

Reécrivez sous forme de prédicat (en introduisant des variables).

- Soit l'énoncé suivant : tous les abonnés à 5 ans d'ancienneté sont concernés
 - $\forall x.(\text{abonne}(x) \wedge (\text{anciennete}(x) \geq 5 \Rightarrow \text{concerne}(x)))$

Modélisation ensembliste

Soient les ensembles A, F, S

A : un ensemble d'abonnés (téléphonie)

S : un ensemble de services

F : un ensemble de forfaits

soient les **relations** r_s et r_f suivantes :
les abonnés à des services

$$r_s : A \leftrightarrow S$$

les abonnés à des forfaits

$$r_f : A \leftrightarrow F$$

On peut **rechercher des abonnés à un service particulier**
rechercher des abonnés à un forfait particulier
calculer le nombre d'abonnés au service ss

Modélisation ensembliste (suite)

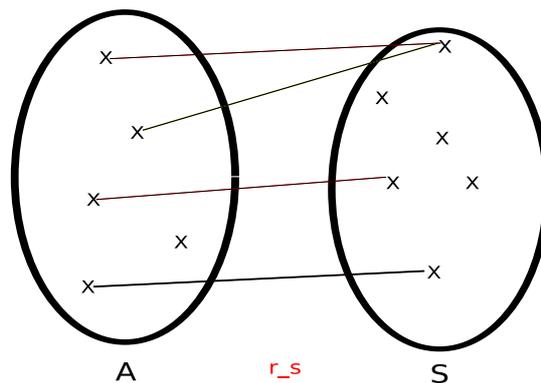


Figure: Exemple de relation (Diagramme de Euler-Venn)

Exercices : Dessiner à l'aide de diagramme de Venn, des fonctions, injections, surjections, bijections

Modélisation ensembliste (suite)

Affichage de l'état des places dans une salle

$PLACE$ / * un ensemble abstrait de places * /

$ETAT = \{occupe, libre\}$

$COULEUR = \{rouge, vert\}$

$etatp : PLACE \rightarrow ETAT$ / * fonction totale * /

$couleurp : ETAT \rightarrow COULEUR$ / * fonction injective * /

...

Affichage(desPlaces) :

Pour chaque place p_i de l'ensemble,

Afficher($couleurp(etatp(p_i))$)

Après la modélisation, le raisonnement

Du **modèle** au **programme** (mais correct !)

Supposons le programme écrit (à partir de la phase de modélisation...)

Est-ce qu'il est correct ?

Comment le montrer ?

☞ **Logique de Hoare** (les fondamentaux sur la correction de programme)

Introduction à la Logique

Les propositions

Appelons P une assertion élémentaire.

C'est un énoncé **VRAI** ou **FAUX** dans un contexte ou une théorie donnée (par exemple la théorie des ensembles).

Une **proposition** est une expression du langage dont la grammaire (règle de formation) est la suivante :

$$\begin{array}{l}
 \textit{prop} ::= P \\
 \quad | \textit{prop} \Rightarrow \textit{prop} \\
 \quad | \textit{prop} \wedge \textit{prop} \\
 \quad | \neg \textit{prop}
 \end{array}$$

Des parenthèses peuvent être utilisées si nécessaire, la grammaire est alors modifiée en conséquence.

Sémantique des opérateurs

La **sémantique des opérateurs ou connecteurs logiques** \Rightarrow , \wedge , \neg est donnée par des **tables de vérité**

P	Q	$\neg P$	$\neg Q$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
0	0					
0	1					
1	0					
1	1					

Exercice : donner les tables de vérité des opérateurs de base

Principe du **tiers exclu** : toute formule est soit VRAI soit FAUX.

Extension du langage à la disjonction et l'équivalence

Le symbole \equiv désigne l'équivalence syntaxique.

Disjonction (\vee) :

$$P \vee Q \equiv \neg P \Rightarrow Q$$

Equivalence (\Leftrightarrow) :

$$p \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$$

On peut compléter la grammaire donnée plus haut avec ces nouveaux opérateurs ; on a ainsi \Rightarrow , \wedge , \neg , \vee , \equiv

Les prédicats : introduction

Le calcul des propositions traite de **vérité absolue**.

Le calcul des prédicats traite de **vérité relative**.

On fait une extension du calcul propositionnel.

Quelle est la valeur de vérité de l'énoncé suivant (i et j sont des entiers) ?

$$i < j$$

Cet énoncé décrit une propriété de deux entiers i et j .

Il est **vrai ou faux** selon les valeurs affectées à i et j .

Les prédicats : introduction

i	j	prédicat $i < j$
0	1	Vrai
2	1	Faux
3	3	Faux

i et j sont des variables.

$i < j$ est un **prédicat** (on parle aussi de *formule* ou de *relation*).

Les prédicats : introduction

Dans les prédicats on utilise deux sortes de variables :

- les **variables libres** et
- les **variables liées**.

Exemples :

Que dire de

$$x + 3 > y$$

Formation des prédicats

Les prédicats sont construits avec :

- des constantes (0, 2, ...),
- des symboles de fonction (f, g, +, *, ...),
- d'autres prédicats (P, Q, R, ...)

et avec des symboles communs à tous les langages du premier ordre :

- les connecteurs logiques \neg , \Rightarrow , \Leftrightarrow , \wedge , \vee de la logique des propositions,
- les **quantificateurs** :
le quantificateur **universel** noté \forall et le quantificateur **existentiel** noté \exists
- les **variables** (x, y, z, ...) qui forment un ensemble dénombrable,
- les parenthèses ouvrantes et fermantes, parfois les crochets [et] et la virgule.

Les quantificateurs

- Le prédicat $\forall x.P$ dit que tout prédicat obtenu en substituant dans P les occurrences libres de x par une expression est un théorème.
- Le prédicat $\exists x.P$ est équivalent à $\neg\forall x.\neg P$.

Il exprime qu'il existe au moins une expression F telle que le prédicat obtenu en substituant dans P les occurrences libres de x par F est un théorème.

Exemples de prédicats

1

$$\neg(x^2 < 0) \vee \forall y.y + 2 = x$$

2

$$\forall n.\exists m \mid m > n$$

Tout entier n a un supérieur

3

...

Substitutions en logique

Une **substitution** est un **remplacement des occurrences libres** d'une variable dans une expression par une autre expression donnée.

Il y a diverses notations ; on trouve souvent :

soit $P[v \setminus x]$: la substitution de v à toutes les occurrences libres de x dans P

soit $[v \setminus x]P$: la substitution de v à toutes les occurrences libres de x dans P .

Substitutions en logique

Des exemples :

- Soit le prédicat $P_1 : x > 2$
la substitution $P_1[3 \setminus x]$ donne $3 > 2$
- Soit le prédicat $P_2 : \forall x.(x > 2) \vee (x = 0)$
Que donne la substitution $P_2[3 \setminus x]$?

Attention aux variables **libres, liées** !

On ne remplace (substitue) que les variables libres.

Introduction à la théorie des ensembles

Introduction à la théorie des ensembles

Cantor :

- **Un ensemble** : regroupement d'éléments de même nature ou propriété.
- Exemples : entiers naturels (N), relatifs (Z), réels (R) , ...
mais aussi HOMME, FEMME, ETUDIANT, NUMINSEE

On peut définir les ensembles

- **abstrait**
- en **extension**
- en **compréhension**
- en **composant** d'autres ensembles (avec des opérateurs)

Introduction à la théorie des ensembles

- Éléments dans un ensemble E ;
opérateur d'appartenance : $x \in E$
- Opérateurs sur les ensembles E, F :

union	$E \cup F$
intersection	$E \cap F$
différence	$E \setminus F$
produit cartésien	$E \times F$
inclusion	$E \subseteq F$
...	...

Relations

- **Relation** = sous-ensemble du produit cartésien
- Exemple
 $r : ETUDIANT \leftrightarrow GROUPE$
 r est un sous-ensemble de $ETUDIANT \times GROUPE$

Les **fonctions** sont des relations dotées de propriétés particulières.

- fonctions : **partielles, totales, injectives, surjectives, bijectives**

Ce sont des concepts fondamentaux en modélisation formelle (du logiciel).

Abstraction, analyse de besoins

Soit à construire un logiciel de **gestion d'une base de documents musicaux** de divers types.

Ici, un document musical est l'œuvre d'un auteur ou d'un compositeur ; **il a un titre, un genre, une durée, une occupation** sur le support de stockage et est enregistré sous un format donné (mp3, mpeg, ...).

Une application logicielle qui fonctionnera sur un lecteur mp3 devra permettre à son utilisateur de **trier** les documents selon leurs auteurs, leurs genres, leur occupation ou leur durée ; **d'écouter** un document, **d'arrêter** une écoute, **effacer un document**, **envoyer à...**, etc

Un document ne peut pas être rangé dans deux genres différents.

Tout document doit avoir un titre, un genre et un format.

Un document ne peut être en même temps en écoute, ...

Analyse des besoins

Données identifiées et caractéristiques :

- document
 - auteur
 - titre
 - genre
 - format de stockage
 - durée
 - occupation

Fonctionnalités voulues :

Pour des documents :

- Trier un **ensemble de documents**

Pour un document :

- écouter
- arrêter l'écoute
- envoyer à
- supprimer

☞ Structuration, spécification fonctionnelle

Analyse, abstraction, modélisation

Prenons des ensembles abstraits : **DOCUMENT, TITRE, GENRE, FORMAT, AUTEUR, ETATDOC, ...**

$ETATDOC = \{enEcoute, \dots\}$
 $titre \in documents \rightarrow TITRE$ / * fonction totale * /
 $genre \in documents \rightarrow GENRE$
 $format \in documents \rightarrow FORMAT$
 $etatDocument \in documents \rightarrow ETATDOC$ / * fonction partielle * /
 $duree \in documents \rightarrow ETATDOC$
 $auteur \in documents \rightarrow AUTEUR$
 $occupation \in documents \rightarrow POIDS$
 ...

Comment définir les fonctionnalités ?

Abstractions - propriétés - Illustration (suite)

Exemple : Une **figure géométrique** polygone a plusieurs côtés qui partagent deux à deux des coordonnées identiques (extrémités avec le suivant et le précédent).

Du fait du nombre de côtés, des dimensions et de leurs coordonnées, certaines **figures** (polygones) sont dites **carré, rectangle, triangle**.

Propriétés :

triangle : 3 côtés

carré : quatre côtés égaux

rectangle : quatre côtés égaux 2 à 2.

Abstractions - propriétés - Illustration (suite)

Soit *FIGURE* un ensemble abstrait donné ;

$carres \subseteq FIGURE$

$rectangles \subseteq FIGURE$

$triangles \subseteq FIGURE$

$figures \subseteq FIGURE$

$figures = carres \cup rectangles \cup triangles$

$X1 == \dots; X2 == \dots$

$Y1 == \dots; Y2 == \dots$

$dim \in FIGURE \times 1..nbCote \rightarrow N$ /* dimension d'un cote */

$nbcote \in figures \rightarrow N$ /* nombre de cote d'une figure */

$coord \in 1..nbCote \rightarrow X1 * Y1 * X2 * Y2$

Abstractions - propriétés - Illustration (suite)

...

$x \in triangles \Rightarrow nbcote(x) = 3$

$x \in carres \Rightarrow nbcote(x) = 4$

$\wedge \forall i, j \in 1..nbcote(x) .$

$dim(x, i) = dim(x, j)$

$x \in rectangles \Rightarrow nbcote(x) = 4$

$\wedge \forall i \in 1..nbcote(x) . \forall j, k \in 1..nbcote(x) |$

$j = (i + 1) \bmod 4 \wedge k = (i + 2) \bmod 4 .$

$dim(x, i) \neq dim(x, j) \wedge dim(x, i) = dim(x, k)$

...

On peut ainsi raisonner sur des figures..., programmer ...

Abstractions - propriétés - Illustration (suite)

Autre exemple : On veut **savoir si un segment donné en coupe un autre**.

On considère d'abord des modélisations :

Modélisation1

Segment1 : ensemble de points (S_1)

Segment2 : ensemble de points (S_2)

Modélisation2

moins abstraite

Segment1 [AB]: (x_A, y_A) ; (x_B, y_B)

Segment2 [CD]: (x_C, y_C) ; (x_D, y_D)

Lequel des deux modèles est plus satisfaisant ?

Abstractions - Propriétés - Illustration (suite)

La question initiale **savoir si un segment coupe un autre** :

Pour la modélisation1 : il suffit de voir

$$S_1 \cap S_2 \neq \{\}$$

Pour la modélisation2 : il faut

- Expliciter les équations des droites qui supportent les segments.
- Déterminer le point de croisement des droites.
- Vérifier que le point de croisement appartient aux deux segments.

Abstractions - Propriétés - Illustration (suite)

- 1 Equation des deux droites.

Rappel : équation de la droite support d'un segment : $y=ax+b$

Le premier segment qui passe par les points A (x_A, y_A) et B (x_B, y_B) est tel que

$$y_A = a_1 \cdot x_A + b_1 \quad \text{et} \quad y_B = a_1 \cdot x_B + b_1$$

pour trouver les valeurs de a_1 et b_1 il suffit de résoudre le système de 2 équations à deux inconnues (a_1 et b_1):

$$y_A = a_1 \cdot x_A + b_1 \quad \text{et} \quad y_B = a_1 \cdot x_B + b_1$$

Pour le deuxième segment, on fait le même travail

Equation de la droite support du segment 2.

Elle passe par les points C (x_C, y_C) et D (x_D, y_D)

Il faut résoudre les 2 équations à deux inconnues a_2, b_2 :

$$y_C = a_2 \cdot x_C + b_2 \quad \text{et} \quad y_D = a_2 \cdot x_D + b_2$$

pour trouver la valeur de a_2 et b_2 .



Abstractions - Propriétés - Illustration (suite)

- 2 Si les 2 droites se croisent alors il existe un x tel que $y = y_2$;
c'est à dire que l'équation $a_1 \cdot x + b_1 = a_2 \cdot x + b_2$ a une solution :
 $a_1 \cdot x - a_2 \cdot x = b_2 - b_1$
donc on extrait $x = (b_2 - b_1) / (a_1 - a_2)$
- 3 En conclusion,
si x est compris entre x_A et x_B ou égal à x_A ou x_B et
compris entre x_C et x_D ou égal à x_C ou x_D , alors les deux
segments se croisent a cette valeur de x .

Quel est le modèle le plus satisfaisant ?



Les 7 ponts de Königsberg

La ville de Königsberg (aujourd'hui Kaliningrad): deux îles ; un pont relie les deux îles ; six ponts relient le continent à l'une ou l'autre des deux îles.

Le problème : déterminer s'il existe (ou non) une promenade dans les rues de Königsberg permettant, à partir d'un point de départ au choix, de passer une et une seule fois par chaque pont, et de revenir au point de départ.

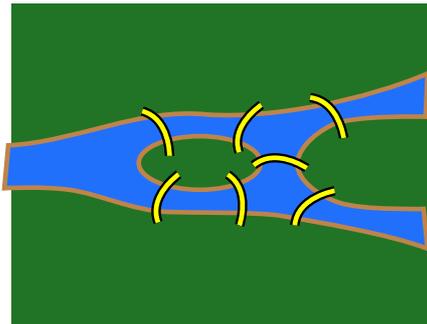


Figure: Ponts de Konisberg

~~Vous devez faire un raisonnement, écrire un programme, ... !~~

Les ponts de Königsberg

Modélisation des données du problème :

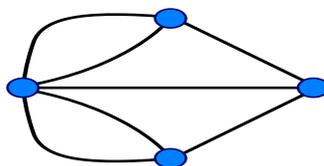


Figure: Graphe des ponts de Konisberg

La promenade demandée n'existe pas ! Solution trouvée par Euler.

Les ponts de Königsberg

EULER a **associé un graphe à la ville** comme dans la figure (**abstraction**).
Pour une solution : il faut ordonner les sept arêtes du graphe de façon à ce que deux arêtes consécutives par rapport à notre ordre soient adjacentes dans le graphe (+ première et dernière consecutive).

Il faut pour cela que tout sommet du graphe soit incident à deux (ou un nombre pair) d'arêtes.

Ce qui n'est pas le cas du graphe (il y a des sommets avec 3 arêtes).

DONC, c'est sûr, il n'y a pas de telle promenade ;

Ce n'est pas la peine d'essayer d'écrire un programme qui en cherche !

Mais on peut écrire un programme qui dit s'il y en a ou pas.

Le barbier

Voir Dijkstra

Les fondamentaux sont incontournables

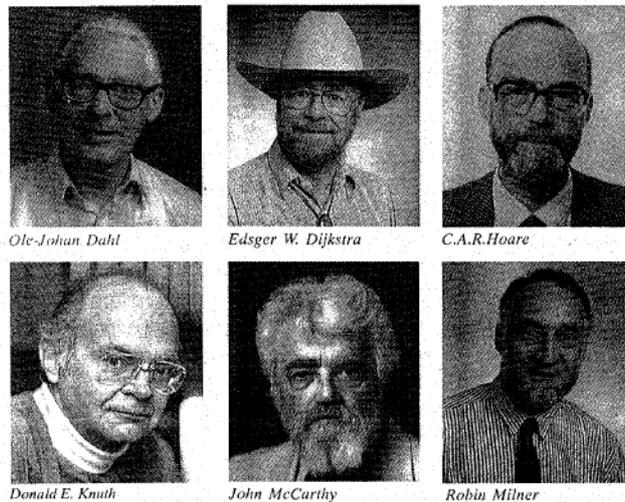


Figure: Des pionniers

Quelques pionniers que vous allez lire, croiser

- A. Turing
- N. Chomsky
- E.F. Codd
- M. Minsky
- K. Gödel
- D. Gries
- R. W. Floyd
- E. W. Dijkstra
- R. Milner
- C. A Hoare
- D. Knuth
- O. Dahl
- J. McCarthy
- D. Parnas
- J. Wing
- ...
- J. Bacckus
- P. Naur
- J. Hopcroft
- L. Lamport
- D. Bjorner
- M. Jackson
- J. Sifakis
(Français, Prix Turing 2007 !)
- ...