M1104 : Introduction aux BD
Partie 1 : SI et Modélisation des données
Abstraction - Modélisation

J. Christian Attiogbé

Université de Nantes - Dpt Info IUT

Septembre 2019

Objectifs

Objectifs du cours (MoDon)

Acquisition de concepts, méthodes et outils permettant une mise en œuvre rigoureuse et maîtrisée 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.

Motivations ? Quels intérêts ?

Pour construire, on a besoin de :

- Outils
- Méthodes
- Techniques
- et plus spécifiquement de langages, techniques, astuces, recettes.

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
- ...

Modélisation

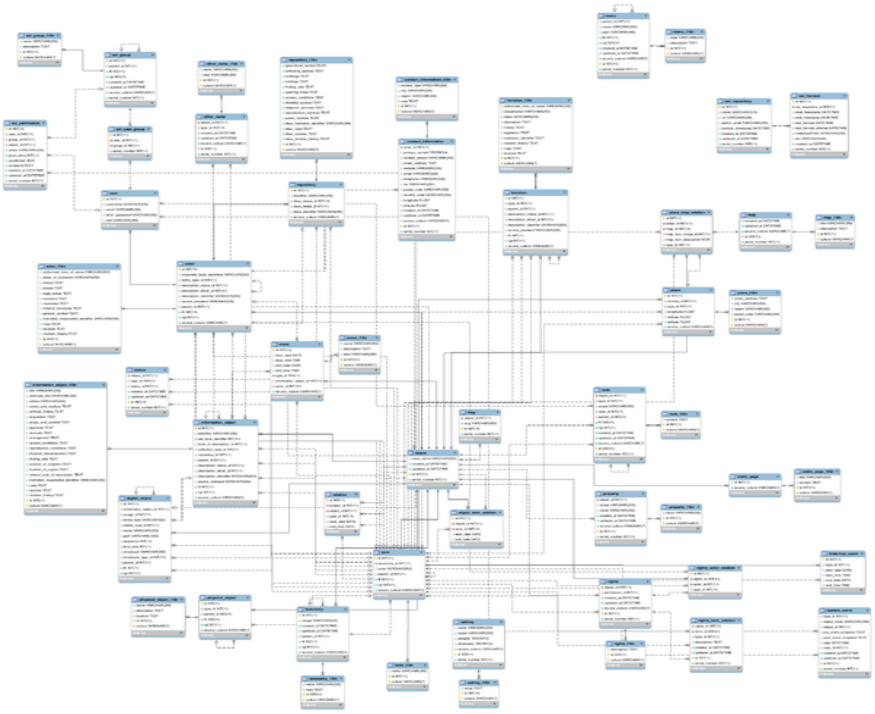


Figure: Exemple de modèle :) :)

Modélisation

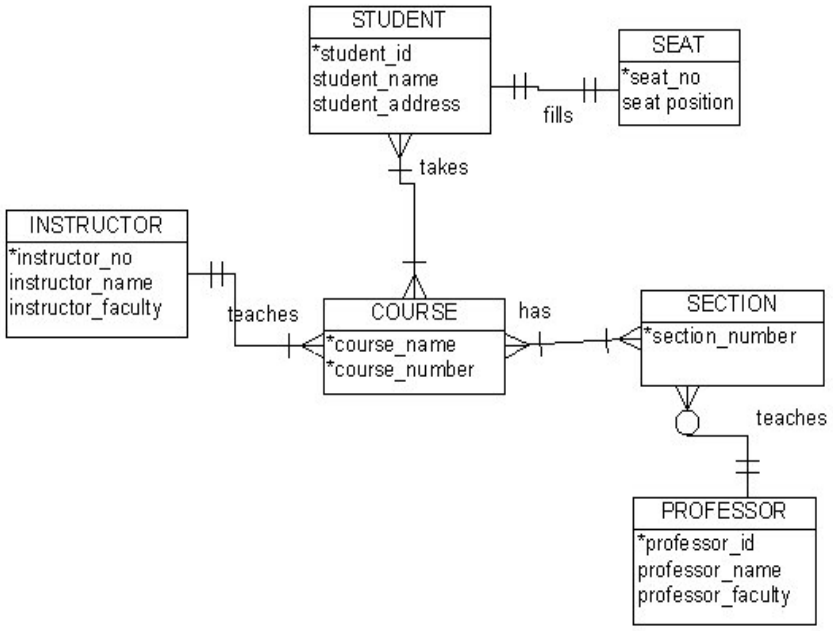


Figure: Exemple de modèle (dans le formalisme *crowfoot*)

Modélisation : où se situe-t-on ?

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

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 ? le modèles des données ? des traitements ?

Modélisation, Système d'information

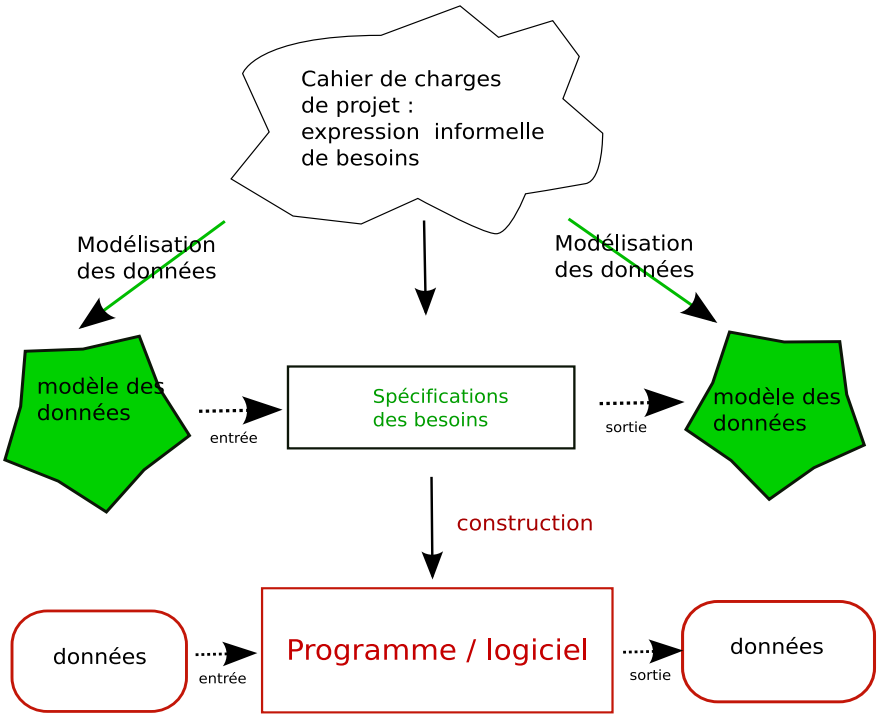


Figure: Place de la modélisation des données

Organisation et volumes horaires (prévision)

- Cours (3 ou 4 séances de 1h20) : C. Attiogbé
- TD : 9h20 (7 séances de 1h20)
- TP : 9h20 (7 séances de 1h20)
  C. Attiogbé, G. Nachouki
- Travail personnel : passionnément
- Contrôles : continu + surprise + à mi-parcours : 15min, 1h, 2h
- Suivi des TDs, compte-rendus, ...
- Devoir surveillé final

Contenu prévisonnel du module (MoDon)

- CM1 - Abstraction - Modélisation et raisonnement logique et ensembliste
  2 TD et 2 TP
- CM2 - Systèmes d'information : paradigmes d'analyse et de conception ; Analyse des besoins, modélisation, propriétés fonctionnelles, non-fonctionnelles
  3 TD et 3 TP
- CM3 - Modèle NIAM et Modèle relationnel de Codd
  3 TD et 3 TP

☛ du travail personnel complémentaire

Si on parlait (encore) de labyrinthe ?

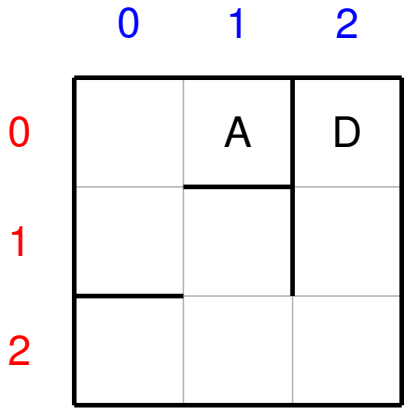


Figure: un labyrinthe

Une façon de voir/abstraire le labyrinthe

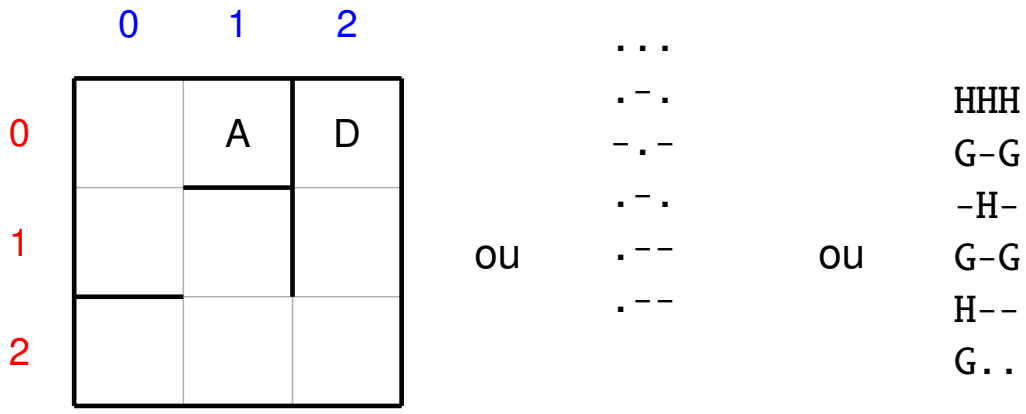


Figure: le labyrinthe

Modélisation de labyrinthe

Qu'est-ce qu'un labyrinthe ?
Un labyrinthe peut être modélisé par :

Labyrinthe

Une suite de **n lignes horizontales** qui se suivent, chaque ligne possède m caractères ; l'ensemble des lignes forme une **grille (nxm)**.
Chaque **ligne** horizontale est caratérisée par deux descriptions :

- l'indication de la présence ou de l'absence d'**un mur horizontal** en haut de chaque case de la ligne
- l'indication de la présence ou de l'absence d'**un mur vertical** à gauche de chaque case de la ligne.

Une position de **départ** ; une position d'**arrivée**.

Est-ce suffisant comme modèle de labyrinthe ?

Modélisation de labyrinthe

Est-ce qu'on peut raisonner (explorer, résoudre) sur le labyrinthe avec ce précédent modèle ?
On peut imaginer :

- une matrice nxm avec des cases,
- une case de départ, une case d'arrivée
- mais comment représenter les murs ? (une case peut être entourée de murs)
- et on doit indiquer pour chaque case, si elle a ou pas des murs (H,B,G,D)

Modélisation de labyrinthe

Un labyrinthe peut être modélisé par :

labyrinthe

- Une matrice (nxm) de cases ;
- pour chaque case on indique la présence ou non des murs
- Une position de départ ; une position d'arrivée.

La case (x,y) a-t-elle un mur en haut ?
Peut-on aller de la case (x,y) à (u,v) ?
Si on peut se servir du modèle pour répondre à toutes les questions qu'on se pose alors c'est un bon modèle.

Modélisation de labyrinthe

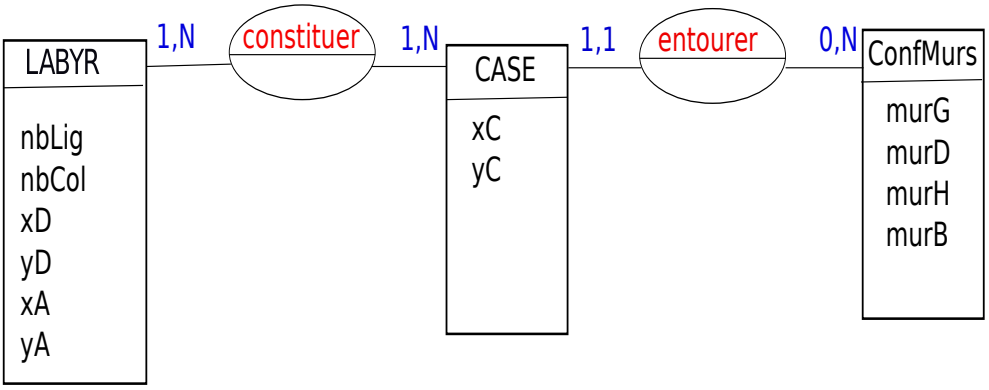


Figure: Un exemple de modèle EAP du labryrinthe

De là, on peut construire des structures de données (conception), ou des bases de données pour traiter les labyrinthes.
De façon générale, on modélise ainsi les systèmes d'information.

Modélisation de labyrinthe

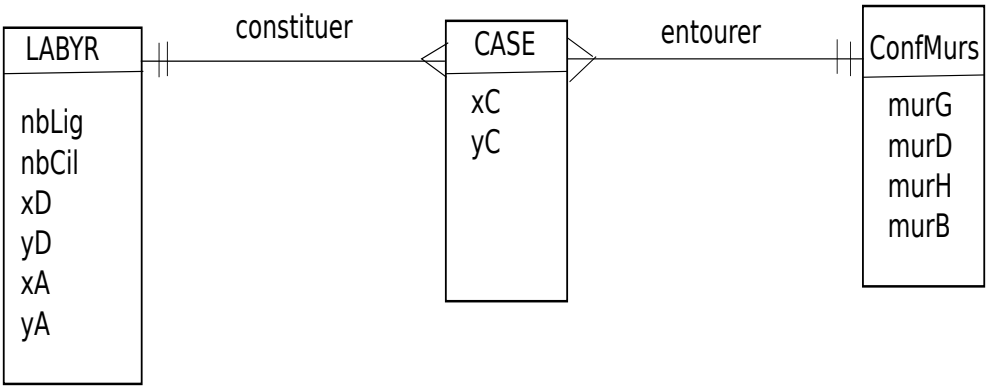


Figure: Un exemple de modèle **Crow Foot** du labryrinthe

Modélisation de système

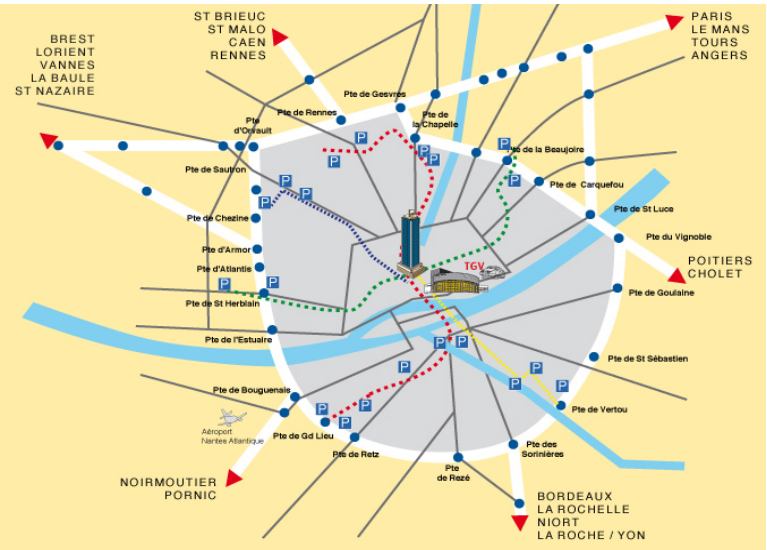


Figure: Périphérique et Parkings Nantes (src : www.lacite-nantes.fr)

Modéliser le réseau de portes et de parking d'un périphérique ; ajouter les restaurants, les centres commerciaux...

Système d'information

Système d'Information (SI)

Un ensemble organisé de ressources permettant de collecter, stocker, structurer, traiter et communiquer des informations dans des entreprises (ou organisations). On distingue

- des systèmes d'information supports d'opérations (traitement de transactions, contrôle de processus industriels, supports d'opérations de communication, etc) et
- des systèmes d'information supports de gestion (aide à la production de rapports, aide à la décision, ...).

Les ressources considérées dans un SI peuvent être : des matériels, logiciels, personnels, données, procédures, etc

Exemples de systèmes d'information

- Système d'information de gestion (dans les entreprises)
- Système d'information personnel (musique, réseaux sociaux, photos, ...)
- Système d'information universitaire (gestion des étudiants, personnels, ...)
- Système d'information hospitalier (gestion hospitalière)
- Système d'information géographique - SIG (entreprises, collectivités locales, régionales, nationales, etc)
- Système d'information de gestion des ressources humaines
- Système d'information maritime, logistique, etc
- Système d'information pour le commandement de forces armées
- ...

Conception des systèmes d'information

Rôle du SI

Le système d'information permet de coordonner les activités de l'entreprise et lui permet ainsi d'atteindre ses objectifs.

Conception de SI

Concevoir un système d'information (et de communication) c'est concevoir comment circule et est stockée l'information de façon efficace et cohérente pour toutes les activités d'une entreprise, d'un réseau d'entreprises, d'une administration publique, des relations entre entreprises, etc

Conception des systèmes d'information

Modélisation des SI

Avant tout, les systèmes d'information reposent sur la modélisation et la structuration des données.

L'informatisation n'est qu'un aspect du système d'information ; elle permet d'élaborer des bases de données pour l'accès aux données, le stockage, le traitement...

☞ Compétences : Analyse, Modélisation et Conception des SI

☞ Métiers : Analyste, ..., Directeur des dystèmes d'information

Composants courants des SI

On inclut souvent dans un système d'information :

- des bases de données de l'entreprise,
- le système informatique comprenant : les ressources et infrastructures réseau (les serveurs d'application ou de données, postes de travail,...)
- des applications spécifiques ou applications métiers,
- des règles et dispositifs de sécurité.

Plan de ce chapitre

Modélisation formelle

En Sciences et Techniques,

on construit à partir de modèles (abstraits, 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

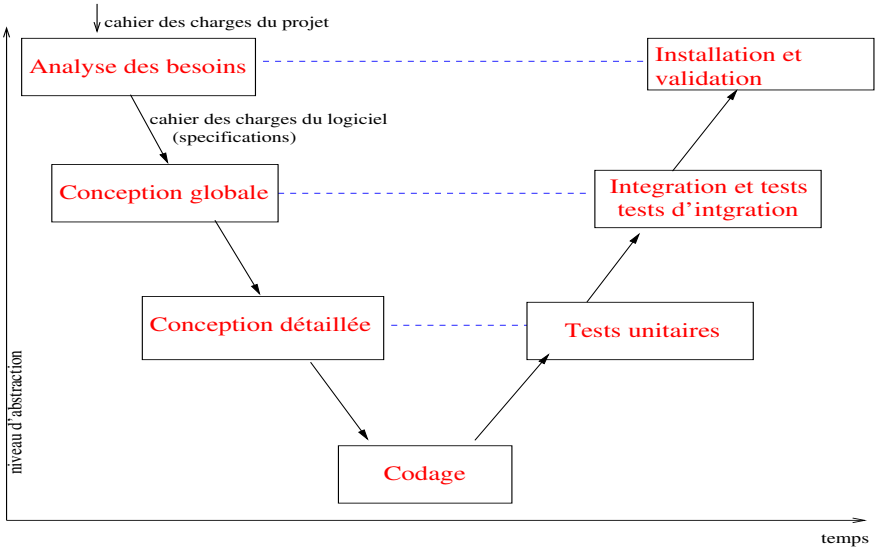
- Expression des besoins
- Analyse des besoins
  (résulte en une spécification, puis à un modèle)
- Conception globale
- Conception détaillée
- Réalisation (développement des produits)
- Mise en oeuvre, tests de conformité aux besoins
- Intégration de différents parties

Modélisation : suite de la phase d'analyse.

Modélisation : où se situe-t-on ?

Informatique > Génie logiciel > construction des logiciels

Un Cycle de vie du logiciel



Modélisation : suite de la phase d'analyse.

Modélisation : 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

ACSI : Analyse et Conception des Systèmes d'information
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 ;

- météorologie (le temps qu'il fera, ...)
- 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 modèle (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

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

$f : Entier \rightarrow Entier$

$x \mapsto y$

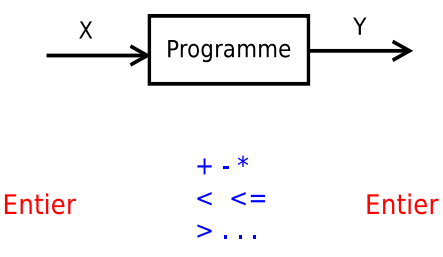


Figure: Idée d'un modèle

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

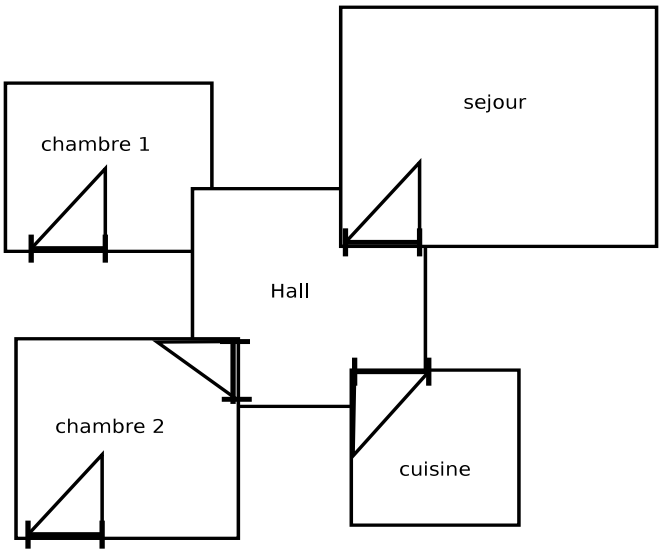


Figure: Plan d'une maison

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

Génie civil

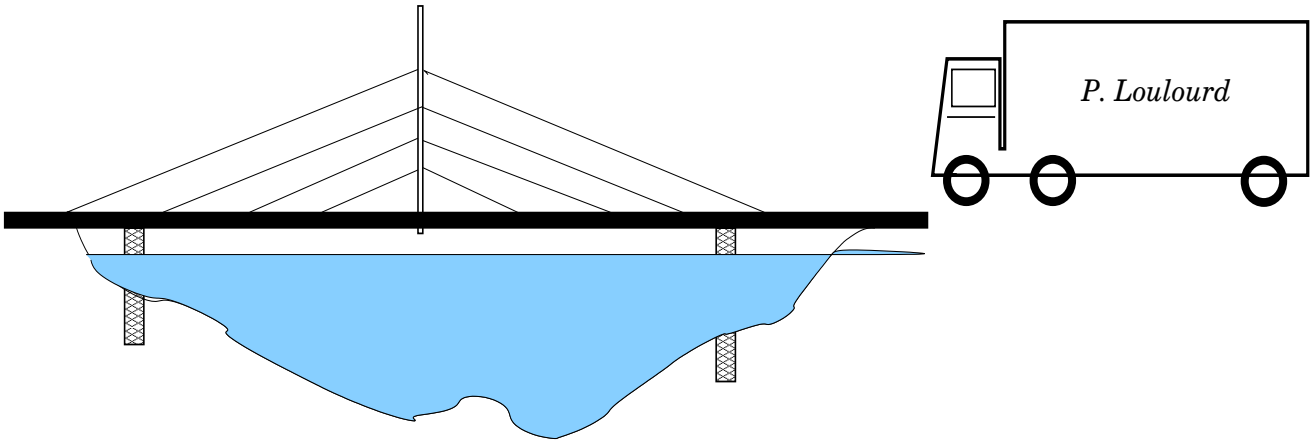


Figure: Schéma/modèle d'un pont

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

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 un labyrinthe dans un programme (logiciel). Comment le modéliser (le représenter) ?
  Qu'est-ce qu'un labyrinthe ?
- Soit à manipuler une chaise dans un programme (logiciel). Comment la modéliser (la représenter) ?
  Qu'est-ce qu'une chaise ?
- Soit à manipuler le dossier d'un patient dans un logiciel (sécu. sociale).
  Qu'est-ce qu'un patient ?
- Soit à manipuler des personnes dans un réseau social
  Qu'est ce qu'une personne ?

☛ Attention à ne pas perdre de l'information lors des abstractions

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 (ensembles, relations, produits cartésiens, etc)** qui représentent l'objet d'étude.

Et aussi : catégorisation, instanciation, composition

Introduction à la Logique (du premier ordre)

La logique

La logique est un formalisme et un outil de raisonnement.
Elle sert à formaliser, modéliser, raisonner.

il existe plusieurs logiques !

- Logigue classique, du premier ordre ✔ suffira ici :
  on manipule propositions et prédicats
- Logique d'ordre supérieur
- Logique non classique
- Logique multivaluée

Les propositions

Proposition

Une proposition est un énoncé élémentaire qui peut être VRAI (1) ou FAUX (0) dans un contexte ou une théorie donnée (par exemple la théorie des ensembles, l'arithmétique, ...).

il fait nuit ; $2 > 7$; $19 \in N$; Cécile est en Info1 ; ...

Une proposition est une expression du langage dont la grammaire (règles de formation) est la suivante où P est un énoncé élémentaire :

$$\begin{array}{lll} prop & ::= & P \\ & | & prop \Rightarrow prop \\ & | & prop \wedge prop \\ & | & \neg prop \end{array}$$

Des parenthèses peuvent être utilisées si nécéssaire, 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.

Sémantique des opérateurs

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

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

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

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.

Prédicat

Un prédicat est un énoncé exprimé à l'aide de variables et dont la valeur de vérité (1 ou 0) dépend des interprétations des variables.

Quelle est la valeur de vérité de l'énoncé suivant (où 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 et j sont des variables.
$i < j$ est un prédicat (on parle aussi de *formule* ou de *relation*).

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

Dans les prédicats on utilise deux sortes de variables :

les variables libres et les variables liées.

Exemple : Que dire de $x + 3 > y$
x et y sont libres (elles ne sont pas liées à un quantificateur)

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 **(** et **)** , parfois les crochets **[** et **]** et la virgule.

Les quantificateurs

- Le prédicat $\forall x.P(x)$ exprime que **tout prédicat obtenu en substituant dans P les occurrences libres de x par une expression** est un théorème.
- Le prédicat $\exists x.P(x)$ exprime qu'**il existe au moins une valeur ou expression** F telle que le prédicat obtenu en substituant dans $P(x)$ les occurrences libres de x par l'expression F est vrai.

Le prédicat $\exists x.P(x)$ est équivalent à $\neg\forall x.\neg P(x)$.

Exemples de prédicats

1. *Tout entier n a un supérieur*

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

2. Les coordonnées du point de départ d'un labyrinthe sont entre les bornes des lignes et des colonnes du labyrinthe.

$$\forall l.(l \in LABYR \Rightarrow ((0 \leq xDep(l) < nbLig(l)) \wedge (0 \leq yDep(l) < nbCol(l))))$$

3. du sens ???

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

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

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

Substitutions en logique

Des exemples :

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

  Attention aux variables libres, liées !

On ne remplace (substitue) que les variables libres.

Introduction à la théorie des ensembles

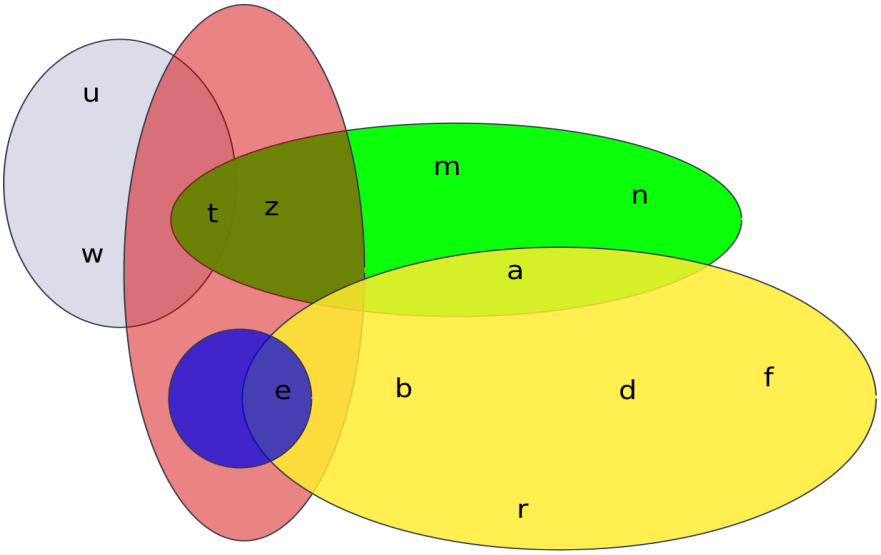


Figure: Exemple de représentation d'ensembles

Introduction à la théorie des ensembles

Théorie des ensembles : un formalisme et un outil de raisonnement.
Elle sert à formaliser, modéliser, raisonner.

Mr Cantor :

- Un ensemble : regroupement d'éléments de même nature ou propriété.

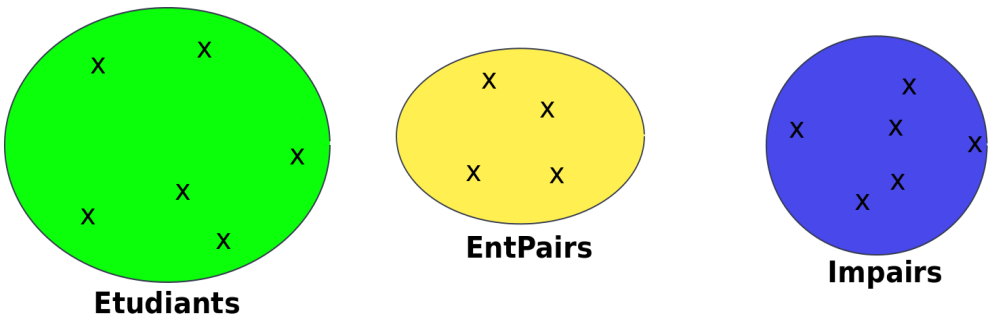


Figure: Des ensembles

Introduction à la théorie des ensembles

- Que pensez-vous de:

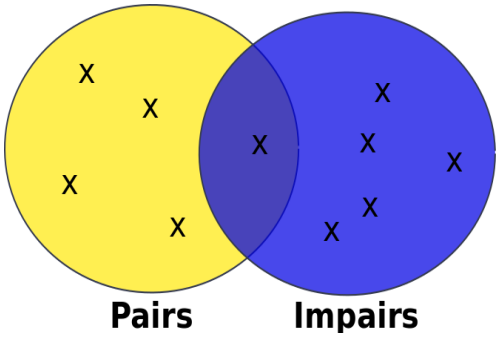


Figure: Une description ensembliste

Introduction à la théorie des ensembles

- Exemples d'ensembles prédéfinis
  entiers naturels (N), NATURAL
  entiers relatifs (Z), INTEGER
  réels (R), REAL
  ...
- On peut définir plusieurs autres ensembles (dans un modèle, ou logiciel) :
  PERSONNE, HOMME, FEMME, ETUDIANT, NUMINSEE, POINT, Abscisses, Ordonnees, Processus, Dossiers, Emoji, etc

Introduction à la théorie des ensembles

On peut définir les ensembles

- abstraits :
  soit S un ensemble ; soit E un ensemble
- en extension :
  Couleur = $\{r, b, v\}$
- en compréhension :
  CarréN4 = $\{n \mid n \in N \wedge n < 5 \,.\, n * n\}$
- en composant d'autres ensembles, avec des opérateurs $\cup$ $\cap$ $\times$:
  Soit $C = A \cup B$, $D = A \cap B$, etc

Introduction à la théorie des ensembles

- Eléments dans un ensemble E ;
  opérateur d'appartenance : $x \in E$ (vrai ou faux)
- Cardinal d'un ensemble E : nombre d'éléments dans E : card(E)
- Opérateurs constructeurs sur les ensembles E, F :

union	$E \cup F, \quad E \cup \{x\}$
intersection	$E \cap F$
différence	$E \setminus F$
produit cartésien	$E \times F$

- Opérateurs relationels sur les ensembles E, F :

inclusion	$E \subseteq F$

Exemple d'ensemble (notation de Venn, Euler

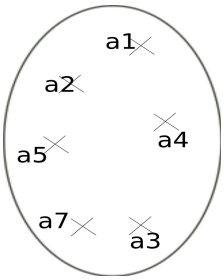


Figure: Exemple d'ensemble (E)

card(E) = 6

Exemples de Produit cartésien

- Soient $E = \{e_1, e_3, e_8\}$ un ensemble d'étudiants,
  $G = \{g_1, g_2\}$ un ensemble de groupes.
  On peut construire
  $E \times G = \{(e_1, g_1), (e_1, g_2), (e_3, g_1), (e_3, g_2), (e_8, g_1), (e_8, g_2)\}$.
- Soit $E = \{e_1, e_3\}$ un ensemble d'étudiants,
  $V = \{v_1, v_3\}$ un ensemble de voitures.
  On peut construire $E \times V = \{(e_1, v_1), (e_1, v_3), (e_3, v_1), (e_3, v_3)\}$.
- Quelles interrogations cela suscitent ?
  quelles compréhensions vous en avez ?
  Quelles idées (interprétations) cela vous suggèrent ?

Relations

Relation

Une relation (notée avec $\leftrightarrow$) est un sous-ensemble du produit cartésien

- Exemple
  $r : ETUDIANT \leftrightarrow GROUPE$
  r est un sous-ensemble de $ETUDIANT \times GROUPE$

Fonction

Les fonctions sont des relations dotées de propriétés particulières.
Un élément de l'ensemble de départ n'a au plus qu'une image.

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

Exemple de relation, Euler, Venn

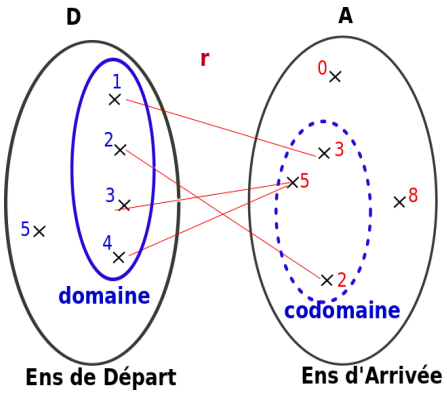


Figure: Exemple de relation - vocabulaire

Soit $r : D \leftrightarrow A$ une relation d'un ensemble D (dit de **Départ**) vers un ensemble A dit d'**Arrivée**
Les éléments de D qui ont une image dans A sont appelés antécédents
Les éléments de A qui ont un antécédent dans D sont appelés images
On appelle domaine de la relation r noté $dom(r)$, l'ensemble des éléments de D qui ont une image dans A.
On appelle codomaine de la relation r noté $ran(r)$, l'ensemble des éléments de A qui sont images des éléments de D.

Exemple de relation, (Euler, Venn)

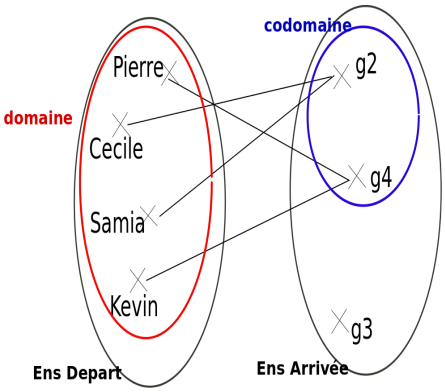


Figure: Exemple de relation

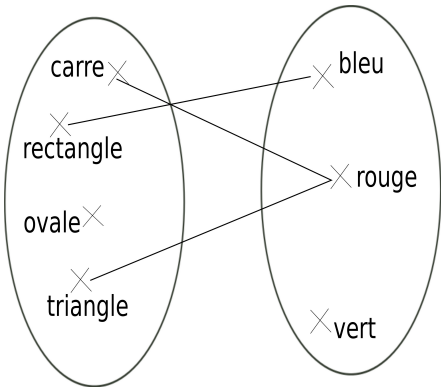


Figure: Exemple de relation

Fonctions

Plusieurs types de fonction

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

fonction injective f (ou une injection)

- une fonction f telle que, pour deux antécédents distincts, on a deux images distinctes ; $x \neq y \Rightarrow f(x) \neq f(y)$

fonction surjective f (ou une surjection)

- une fonction f telle que, tout élément de l'ensemble d'arrivée est image : le codomaine de f égal à l'ensemble d'arrivée ; $ran(f) = A$

fonction bijective f (ou une bijection)

une fonction f telle que f est injective et surjective.

Relations, Fonctions

Lorqu'on a (défini) $r : E \leftrightarrow F$, on peut utiliser aussi son inverse $r^{-1} : F \leftrightarrow E$.

De la même façon on a l'inverse d'une fonction.
Synonymes : inverse, réciproque, transposée

Attention : l'inverse d'une fonction peut être une relation.

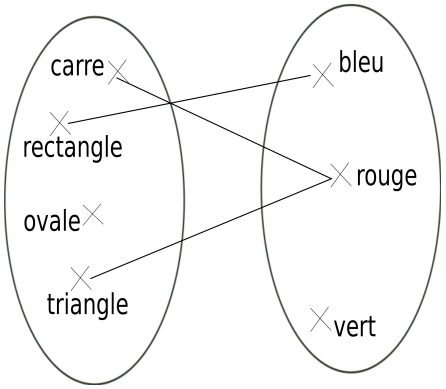


Figure: Exemple de relation (r)

Trouver la relation inverse r^{-1}.

Opérateurs sur les relations

Désignation	Notation	Ascii
relation	$r : S \leftrightarrow T$	r : S <-> T
domaine	$dom(r) \subseteq S$	dom(r) <: S
image	$ran(r) \subseteq T$	ran(r) <: T
composition	$r;s$	r;s
composition r(s)	$r \circ s$	r(s)
identité	$id(S)$	id(S)

Opérateurs sur les relations (suite)

Désignation	Notation	Ascii
restriction domaine	$S \lhd r$	S <| r
restriction codomaine	$r \rhd T$	r |> T
antirestriction domaine	$S \lhd r$	S <<| r
antirestriction codomaine	$r \rhd T$	r |>> T
inverse	$r^{\sim}$	r ~
image relationnelle	$r[S]$	r[S]
écrasement	$r1 \oplus r2$	r1 <+ r2
produit direct de rel.	$r1 \otimes r2$	r1>< r2
fermeture	$closure(r)$	closure(r)
fermeture reflexive trans.	$closure1(r)$	closure1(r)

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, ...

$$
\begin{array}{l}
ETATDOC = \{enEcoute, ...\} \\
titre \in documents \rightarrow TITRE \qquad /* \text{ fonction totale } */ \\
genre \in documents \rightarrow GENRE \\
format \in documents \rightarrow FORMAT \\
etatDocument \in documents \nrightarrow ETATDOC \qquad /* \text{ fonction partielle } */ \\
duree \in documents \nrightarrow ETATDOC \\
auteur \in documents \nrightarrow AUTEUR \\
occupation \in documents \nrightarrow POIDS \\
\ldots
\end{array}
$$

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 ;

$carres \subseteq FIGURE$
$rectangles \subseteq FIGURE$
$triangles \subseteq FIGURE$
$figures \subseteq FIGURE$
$figures = carres \cup rectangles \cup triangles$
$X1 == ...; X2 == ...$
$Y1 == ...; Y2 == ...$
$dim \in FIGURE \times 1..nbCote \nrightarrow N$ $\quad /* \text{dimension d'un cote } */$
$nbcote \in figures \rightarrow N$ $\quad /* \text{nombre de cote d'une figure} */$
$coord \in 1..nbCote \rightarrow X1 * Y1 * X2 * Y2$

Abstractions - propriétés - Illustration (suite)

$$
\begin{aligned}
&\ldots\\
&x \in triangles \Rightarrow nbcote(x) = 3\\
&x \in carres \Rightarrow nbcote(x) = 4\\
&\quad \wedge \forall i, j \in 1..nbcote(x)\ .\\
&\qquad dim(x,i) = dim(x,j)\\
&x \in rectangles \Rightarrow nbcote(x) = 4\\
&\quad \wedge \forall i \in 1..nbcote(x)\ .\ \forall\ j,\ k\ \in 1..nbcote(x)\ |\\
&\qquad j = (i+1)\ mod\ 4 \wedge k = (i+2)\ mod\ 4.\\
&\qquad dim(x,i) \neq dim(x,j) \wedge dim(x,i) = dim(x,k)\\
&\qquad \ldots
\end{aligned}
$$

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]: (xA,yA) ; (xB,yB)
Segment2 [CD]: (xC,yC) ; (xD,yD)

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 (xA,yA) et B (xB, yB) est tel que

$$yA = a1.xA + b1 \quad et \quad yB = a1.xB + b1$$

   pour trouver les valeurs de a1 et b1 il suffit de résoudre le système de 2 équations à deux inconnues (a1 et b1):
   yA=a1.xA + b1 et yB= a1.xB + b1

   Pour le deuxième segment, on fait le même travail
   Equation de la droite support du segment 2.
   Elle passe par les points C (xC, yC) et D (xD, yD)
   Il faut résoudre les 2 équations à deux inconnues a2, b2:
   yC =a2.xC + b2 et yD= a2.xD + b2
   pour trouver la valeur de a2 et b2.

Abstractions - Propriétés - Illustration (suite)

2. Si les 2 droites se croisent alors il existe un x tel que y = y2 ; c'est à dire que l'équation a1.x + b1 = a2.x + b2 a une solution : a1.x - a2.x = b2 - b1
   donc on extrait x=(b2 - b1)/(a1 - a2)
3. En conclusion,
   si x est compris entre xA et xB ou égal à xA ou xB et compris entre xC et xD ou égal à xC ou xD, 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.

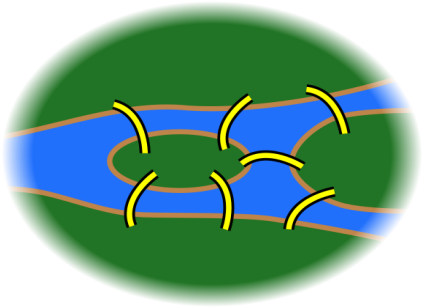


Figure: Ponts de Konisberg

Les ponts de Königsberg

Modélisation des données du problème :

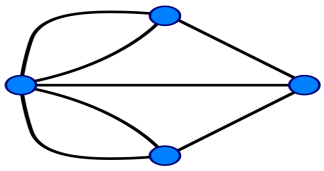


Figure: Graphe des ponts de Konisberg

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 consécutive).

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 Dijskstra

Les fondamentaux sont incontournables

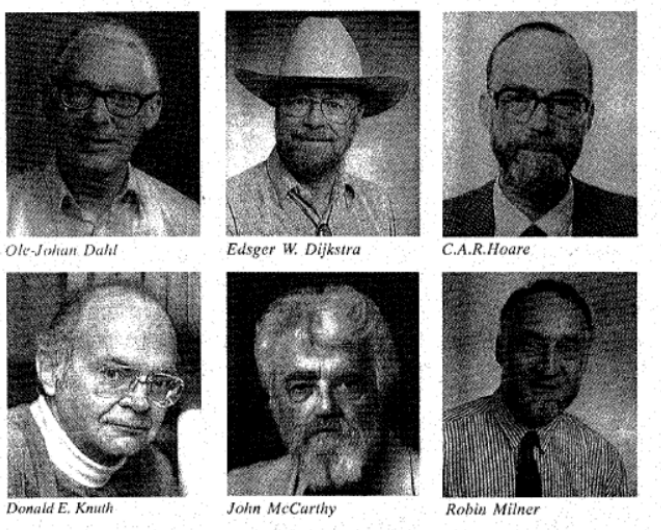


Figure: Des pionniers

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 !)
- ...