Sur la science informatique et son installation à Montpellier

Michel CHEIN
Professeur émérite à l'Université de Montpellier
Académie des Sciences et Lettres de Montpellier

MOTS-CLÉS
Science informatique, algorithme, laboratoire de recherche, Centre de Recherche en Informatique de Montpellier, CRIM, Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, LIRMM

RÉSUMÉ
Dans une première partie on présente rapidement la science informatique que l'on peut définir comme la science du traitement automatique et rationnel de l'information. On présente ses diverses composantes : informations, algorithmes et machines en insistant sur les algorithmes qui sont aujourd'hui présents partout, y compris dans les médias. La deuxième partie est consacrée à une très courte histoire du développement de l'informatique à Montpellier.

Le lecteur peut visionner l'enregistrement vidéo de cette conférence **VIDEO**

En guise d'introduction …

Imaginons un jeune informaticien venant d'être embauché dans le service informatique d'une grande ville. Le maire, un peu mégalomane, lui demande de chercher tous les articles de la presse parlant de lui. « Ça devrait être facile pour vous, tous les journaux sont informatisés maintenant ! » lui dit le maire, qui se targue de promouvoir les technologies HighTech et qui parle souvent de TIC et de numérique. Pour les besoins de cet exemple supposons que le maire s'appelle Jean-Claude Cros (comme le maire de mon village, qui est un ami, qui n'est pas mégalomane et qui ne m'en voudra pas si je prends son nom comme exemple). L'informaticien, qui est jeune, se dit "c'est facile". Je vais parcourir les fichiers des journaux un par un et je vais chercher si on trouve la chaine de caractères « Jean-Claude Cros ». Il pense aussi à considérer des variantes, comme le prénom écrit avec ses seules initiales « J.-C. » ou sans tiret « Jean Claude », l'oubli des majuscules, et, dans le cas d'un journal parisien, il se demande s'il ne devrait pas chercher aussi des « Jean-Claude Crosse ». Continuant sa réflexion, il pense que les articles mentionnant « le maire de La Boissière » devraient être recherchés et aussi les articles parlant des opposants au maire … bref, il généralise son problème, le rend abstrait, le modélise de telle sorte qu'il puisse automatiser sa résolution et finalement il arrive au problème suivant :

Données : une chaine de caractères $c_1c_2 \ldots c_n$ (le fichier informatique d'un journal) et une deuxième chaine de caractères $d_1d_2 \ldots d_m$ (le nom de la personne cherchée).

Problème : est-ce que $d_1d_2 \ldots d_m$ est une sous-chaine de $c_1c_2 \ldots c_n$?

Il trouve tout de suite un algorithme très simple. Celui-ci consiste (Figure 1) à positionner la sous-chaine sous la chaine et à parcourir la sous-chaine et la chaine,

caractère par caractère, en regardant si les caractères se correspondant dans la sous-chaine et la chaine sont identiques, s'il y a correspondance jusqu'au dernier caractère de la sous-chaine on a trouvé une occurrence, sinon on décale la sous-chaine d'une position et on recommence comme schématisé sur la figure ci-dessous.

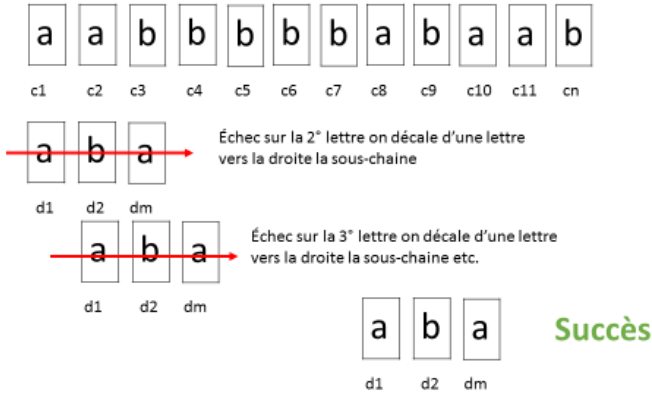

Figure 1 : Une chaîne contient-elle sur sous-chaîne ?

Notre informaticien programme son algorithme et, tout content, il va voir un de ses anciens professeurs, chercheur au LIRMM, qui lui dit : « Ça ne va pas du tout ! Vous risquez d'avoir à utiliser très souvent votre algorithme, surtout si le maire est un peu paranoïaque … L'algorithme naïf que vous avez programmé n'est pas optimal il est en O(mxn), il y a des algorithmes en O(n) … »

Le pauvre malheureux ne se souvient plus de ce que veut dire O(n) … il va voir dans le Cormen, la bible de l'algorithmique pour les étudiants en informatique, et découvre des algorithmes qui utilisent des prétraitements et des structures comme des automates et qui semblent plus rapides que son algorithme naïf… il programme, il teste, il compare. …

Après avoir résolu d'autres questions techniques (panne du serveur du journal en cours de traitement, divers codages des caractères, fichier arrivant comme un flot, etc.), il a envie de montrer son travail au maire qui lui demande alors une démonstration sur le quotidien local du jour.

Effet-démonstration classique : ça ne marche pas ! Il avait testé son algorithme sur « Le journal du hacker », dont le fichier est bien structuré en articles, chaque article est lui-même bien structuré, les photos sont bien identifiées les légendes aussi … ce qui n'est pas le cas du journal local … il se remet au travail, fait des tests sur tous les journaux qu'il a vu empilés sur le bureau du maire, prudent fait des tests sur quelques autres, et retourne le voir.

Le maire aimerait que ça aille très vite pour que le matin dans son bureau il puisse faire lui-même les recherches (« parfait, se dit l'informaticien, j'ai optimisé mon programme »), il souhaiterait aussi un logiciel convivial (« il me faudra un peu de temps mais je devrais y arriver ») et puis le maire lui dit qu'il ne veut que les articles le concernant lui et pas ses homonymes (« Euh !... je vais voir ce que je peux faire, répond notre informaticien un peu inquiet »).

Mais il ne peut pas faire grand-chose. La recherche d'entités nommées, qui peut concerner des personnes, comme dans le cas du maire, mais aussi des institutions, des entreprises etc., est un problème dont il est facile de voir l'importance mais qu'on

ne sait pas résoudre aujourd'hui avec suffisamment de généralité. C'est un problème de recherche actif aujourd'hui mais il n'a pas de solution « sur l'étagère ».

Ce problème, en apparence « simple », nécessite des activités scientifiques dans au moins trois domaines de l'informatique : en algorithmique, en linguistique computationnelle et, puisque le maire veut un logiciel convivial, en communication homme-machine.

Après cet exemple introductif exposant la démarche informatique, nous donnerons une vision très générale de la science informatique, en insistant sur l'algorithmique, et la dernière partie de cet article sera consacrée à une petite histoire du développement de l'informatique à Montpellier.

1. La science informatique

À partir de l'étymologie du mot informatique, construit à partir de *infor*(mation) + (auto)*matique***,** certains font marcher la science informatique sur deux jambes. J'aime bien la faire marcher sur les 3 jambes de la figure 2 en la définissant comme la *science du traitement automatique et rationnel de l'information*, d'autres la font marcher sur 4 jambes, en ajoutant les langages, mais si 2 et 3 ont des charges symboliques plus riches que 4, 3 l'emporte définitivement en ce qui concerne la stabilité (d'une table par exemple …) !

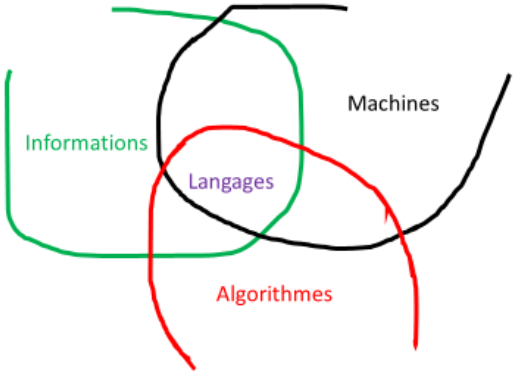

Figure 2 : Les trois piliers de l'informatique

Quelle est la démarche quand on veut résoudre informatiquement un problème ? Il faut modéliser la tâche que l'on veut résoudre, c'est-à-dire représenter les informations que l'on veut communiquer, les données du problème en vue de leur traitement par des procédures de résolution. On peut être amené à devoir transformer ces données, les structurer, les compresser, corriger des erreurs qui peuvent se produire au moment de la transmission. On devra aussi prévoir l'évolution de ces données et résoudre des problèmes de stockage et de protection de ces informations etc.

Ces informations peuvent être de différente nature (des textes, des sons, des images, des nombres, …) et de différentes complexités (une donnée provenant d'un capteur, ou des connaissances provenant d'experts dans un domaine, …) et doivent être discrétisables, c'est-à-dire représentables par une séquence (finie) de symboles.

Un ordinateur est une machine physique universelle pour manipuler des symboles, elle permet de mettre en œuvre n'importe quel algorithme symbolique.

Il ne faut pas se tromper sur le sens de cette universalité. Information est un terme éminemment polysémique et, même si les informations sont discrétisées, tout problème n'est pas nécessairement susceptible d'un traitement informatique.

Voici une anecdote véridique à ce sujet. C'était il y a bien longtemps, lorsque, après avoir quitté le CNRS, j'occupais mon premier poste de professeur (ce n'était pas à Montpellier). J'avais été accueilli très chaleureusement par mes nouveaux collègues de toutes les disciplines et l'un de ceux-ci, un philosophe, était venu me voir dans mon bureau quelques jours après la rentrée. Un ruban perforé à la main, il m'a posé la question suivante : « J'ai codé deux chapitres de la science de la logique de Hegel, pourriez-vous m'aider à comprendre sa logique dialectique sur laquelle je me casse les dents depuis un certain temps ? »). Ah ! … ai-je répondu après un long silence embarrassé.

Ne sachant pas trop comment me sortir de cette situation délicate, quelques jours plus tard, j'ai proposé à l'assemblée des professeurs, ça marchait comme ça à l'époque, un cours d'informatique pour les sciences humaines qui serait axé sur la question suivante : quel type de problèmes peut-on, ou ne peut-on pas, résoudre en informatique ? Mon collègue a suivi ce cours de complexité et calculabilité avec beaucoup d'attention. À la fin du trimestre il est venu me voir et m'a dit : « Tu te souviens de notre première rencontre ? Tu as dû me trouver bien stupide avec ma question sur Hegel ? »

Dans le schéma de la figure ci-dessus les « patates » ne sont pas fermées. Les ouvertures représentent des connexions avec d'autres disciplines : vers les mathématiques pour l'algorithmique, vers toutes les disciplines (agronomie, linguistique, médecine, physique, psychologie, …) pour les informations, vers l'électronique, principalement aujourd'hui, pour les machines. Dans ce schéma, il manque une troisième dimension pour indiquer que ces différentes parties de l'informatique ont toutes des aspects plus ou moins théoriques.

1.1. Les algorithmes

Ce qui suit concerne l'algorithmique. Pour des raisons de place il fallait faire un choix et, à moins d'être très superficiel, il était difficile de considérer les trois jambes de l'informatique. Ce choix a été guidé par l'importance des algorithmes aujourd'hui dans les médias et par le résultat d'un sondage. D'après un sondage réalisé par l'IFOP pour la CNIL en janvier 2017, 83 % des Français ont entendu parler d'algorithmes mais ils sont plus de la moitié à ne pas savoir précisément de quoi il s'agit (52%) et je ne pense pas qu'on puisse en conclure que 48% savent précisément ce qu'est un algorithme … ce que prouve cette liste de titres d'articles publiés en janvier 2017 qui est bien loin d'être exhaustive !

Algorithmes, vont-ils tout décider à notre place ? (Capital)
Un algorithme pour détecter les pensées suicidaires (Le Figaro)
Les algorithmes sont-ils les nouveaux maitres du monde ? (France Inter)
Les radiologues virés par les algorithmes (La Tribune de Genève)
Les algorithmes n'ont pas eu la peau de la télévision (Rue89)
Les algorithmes sont partout, il faut les contrôler ! (France Culture)
L'Etat met les mains dans les algorithmes (Le Monde)
Grâce à cet algorithme, recréez un Picasso ou un Van Gogh en une heure ! (Le Figaro)
Les algorithmes régissent-ils nos vies ? (France Culture)
La propagande des algorithmes (internetactu)

Si vous ne connaissez pas les algorithmes, eux vous connaissent très bien (Télérama)
Les algorithmes peuvent creuser les inégalités et saper la démocratie (Rue89)
Macron est le candidat des algorithmes (France Inter)
Les algorithmes et la disparition du sujet (The Conversation)
Facebook remplace ses éditeurs humains par des algorithmes (Le Figaro)
Le numérique, bouée de secours des zones rurales ? (La Tribune)
Comment les algos nous rendent tous débiles (Rue89)
A quoi rêvent les algorithmes ? (livre de Dominique Cardon)
La chair et l'algorithme (Pièce de théâtre, Théâtre de la Reine Blanche)

Les algorithmes font peur ou font rêver et un effort d'éducation et de transparence est nécessaire pour éviter les peurs infondées mais aussi les usages dangereux ou éthiquement discutables.

Qu'est-ce donc qu'un algorithme ? Voici une première définition.

Un algorithme, c'est une façon de décrire dans ses moindres détails comment un mécanisme automatique doit procéder pour faire quelque chose.

Pour qu'un système informatique soit capable de réaliser une tâche, tout seul, une tâche simple comme regarder si un mot a autant de lettres A que de lettres B, ou une tâche compliquée comme piloter un métro, il faut lui expliquer dans les moindres détails tout ce qu'il doit faire : tout doit être explicité.

On prend souvent l'image d'une recette de cuisine pour donner un exemple d'algorithme. Considérons la recette du potage aux orties que l'on trouve sur le site marmiton à réaliser avec les ingrédients suivants : *5 poignée d'orties, 3 cuillères à soupe d'huile, 1 gros oignon, 4 ou 5 pommes de terre, 15 cl de crème fraîche.*
Cueillez sans vous piquer, pour cela utilisez des gants, quatre ou cinq bonnes poignées d'orties.
Lavez-les soigneusement afin d'en retirer le pouvoir urticant.
Dans une cocote, versez 3 cuillères d'une bonne huile et faites revenir l'oignon émincé, ajoutez les pommes de terre taillées en petits cubes, puis les feuilles d'orties sans les tiges.
Couvrez d'eau, salez, poivrez à votre goût portez à ébullition puis laissez cuire le temps qu'il faut.
Mixez, réchauffez en ajoutant la crème fraîche. Servez sans attendre.

Est-ce un algorithme ?

– Non. Il y a beaucoup de choses imprécises : une bonne poignée, des Rate ou des Bintje, laissez cuire le temps qu'il faut …

– Mais **oui**, si vous avez un algorithme de traitement de la langue naturelle et des connaissances en cuisine (poignée = A gr, cuillère = B cc, 1 gros oignon = C gr, D cc d'eau, temps de cuisson = E minutes, etc.) vous pourrez peut être construire (en ajoutant de la robotique) un automatisme produisant des boites de potage aux orties !

On dit parfois que la pédagogie c'est l'art de répéter plusieurs fois la même chose alors : la notion d'algorithme est un concept qui traduit la notion intuitive de *procédé systématique, applicable mécaniquement, sans réfléchir, menant au résultat voulu en suivant simplement un mode d'emploi précis.*

Un algorithme doit fournir un résultat, il doit donc s'arrêter, un algorithme décrit un processus fini.

Un dernier mot : alors que les hommes utilisent des algorithmes depuis des milliers d'années, leur irruption dans l'espace public provient des deux autres aspects de l'informatique : ordinateur (pour tous) et informations.

1.2. Un deuxième exemple

Après la recherche d'un mot dans un texte, nous allons considérer le problème du voyageur de commerce. Un voyageur de commerce doit parcourir un certain nombre de villes reliées par des routes en ne passant qu'une fois et une seule par une ville. Dans la figure 3 on considère 10 villes, notées a, b, …, j et on met une croix dans une case lorsqu'il existe une route entre la ville de la ligne et la ville de la colonne de cette case. Ainsi, on a une route entre a et b, une route entre a et f etc. et, comme on suppose qu'il n'y a pas de sens unique, on a aussi une route entre b et a, et une entre f et a etc. ceci est représenté par le tableau de gauche.

C'est un problème en apparence simple, un puzzle, puisqu'il s'agit de trouver si on peut extraire de ce tableau un motif composé d'une unique croix par ligne et par colonne. Un tel motif est représenté par les croix en gras à gauche et par le tableau de droite obtenu en changeant l'ordre des lignes et des colonnes pour mettre en évidence la solution trouvée : chiafdbgjec.

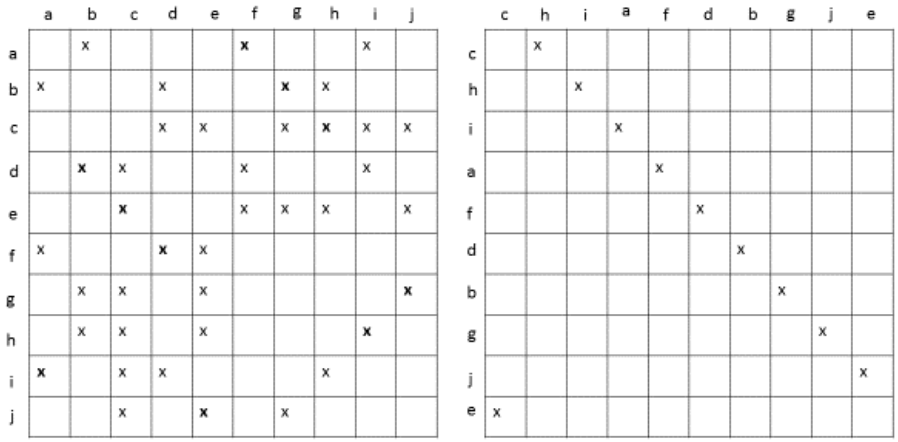

Figure 3 : Problème du voyageur de commerce

On peut construire, un peu plus difficilement que dans le premier exemple, un algorithme naïf qui consiste à considérer tous les parcours possibles entre n villes puis regarder pour un tel parcours s'il existe une ligne directe entre deux villes consécutives.

On s'aperçoit rapidement que le temps de calcul augmente très rapidement en fonction du nombre de villes. Avec 70 villes le nombre de parcours à tester (environ 10^{100}) dépasse le nombre d'atomes de l'univers observable (environ 10^{80}).

On ne connait pas d'algorithme général, fonctionnant quelque que soit la configuration des routes entre les villes, qui soit fondamentalement plus rapide.

Tous les algorithmes connus pour résoudre ce problème peuvent prendre, dans certains cas, un temps exponentiel en le nombre de villes et beaucoup de chercheurs pensent qu'on ne trouvera pas mieux !

Si ce problème est difficile à résoudre il est par contre facile de vérifier si un parcours donné satisfait la condition de passer une fois et une seule par chaque ville.

Pour vérifier que, dans notre exemple, afdbgjechia est bien une solution, il suffit de parcourir une fois cette chaine de caractères en vérifiant qu'on a bien une route entre deux villes consécutives et que toutes sont atteintes.

Ainsi nous venons de voir des exemples appartenant à deux classes de problèmes :

P = les problèmes « faciles » à résoudre (comme notre premier exemple)

NP = les problèmes « faciles » à vérifier (comme notre deuxième exemple)

Savoir si P=NP ou si P≠ NP, est l'un des 7 problèmes dotés par l'Institut Clay d'un million de dollars.

Parmi les problèmes de NP certains, appelés NP-complet, sont aussi difficiles que n'importe quel problème de NP. Si vous trouvez un algorithme « efficace » pour résoudre un problème NP-complet, n'importe lequel, par exemple le problème du voyageur de commerce que nous venons de voir (il existe des centaines de problèmes NP-complets), alors on a un algorithme efficace pour n'importe quel problème de NP … donc vous avez démontré que P=NP et vous devenez millionnaire !

Une large majorité de chercheurs pensent que P≠ NP.

Lorsque j'étais à Paris VI j'avais invité Jack Edmonds pour faire une conférence sur ce sujet. Nous avions déjeuné au Buisson Ardent, il aimait bien le Cahors et il fit une conférence mémorable. Il expliqua qu'embauché comme ingénieur pour résoudre le problème du voyageur de commerce il n'arrivait pas à trouver de solution efficace. Au bout de quelques semaines, il alla donc voir son patron et lui expliqua que s'il n'avait pas trouvé de solution efficace c'était parce qu'il n'y en avait pas, parce que le problème était intrinsèquement difficile … il fit sa conférence puis il conclut en disant qu'en réalité il n'avait pas été assez malin pour trouver un algorithme efficace mais il avait été assez malin pour garder son poste ! Je ne sais pas si ce n'était qu'une histoire mais depuis elle fait partie du folklore en algorithmique.

Après cette conférence, un certain nombre d'étudiants avaient pris au mot Jack Edmonds et s'étaient mis à chercher des algorithmes efficaces pour des problèmes NP-complets et nous avons perdu pas mal de temps à analyser leurs algorithmes pour trouver leurs erreurs !

1.3. Un troisième exemple

Non seulement il existe de très nombreux problèmes pour lesquels on ne connait pas d'algorithme efficace, non seulement il existe des problèmes pour lesquels on ne sait pas vérifier facilement une solution, mais la situation est encore pire, il existe des problèmes pour lesquels on peut prouver qu'il n'existe pas d'algorithme pour les résoudre et ce ne sont pas des problèmes insolites. Par exemple, savoir si un programme s'arrête est un problème algorithmiquement indécidable : il n'existe pas d'algorithme qui étant donné un programme P et une donnée D s'arrête en concluant que soit le programme P s'arrête pour D soit il ne s'arrête pas, et ceci quels que soient P et D.

Car si un algorithme doit s'arrêter, un programme ne s'arrête pas nécessairement !

Je n'ai illustré très rapidement que l'algorithmique, qui concerne des structures abstraites de toute sorte, composées de symboles et pas seulement de nombres. Il aurait fallu autant de temps pour présenter quelques concepts de chacun des autres aspects de la science informatique.

1.4. Le mode de pensée informatique

Pour terminer ce survol de la science informatique je dirai un mot de ce que Jeannette Wing appelle le mode de pensée informatique.

Toutes les disciplines scientifiques procèdent par abstraction, le propre de la science informatique est qu'une **modélisation informatique** doit être **exécutable par une machine physique** : mécanique, hydraulique, électrique, quantique, biologique, …

Voici quelques exemples d'abstractions informatiques.

Dans certains jeux de cartes un talon est constitué d'un paquet de cartes sur lequel on ne peut faire que deux opérations : prendre la carte au sommet de la pile ou poser une carte sur le sommet de la pile.

Une *pile* informatique est une abstraction d'une structure de données dans laquelle on ne peut que supprimer la donnée au sommet de la pile ou qu'ajouter une donnée au sommet de la pile (et aussi vérifier si une pile est vide).

Une *file* informatique est une autre structure de données qui modélise une file d'attente : le premier arrivé dans la file est le premier servi alors que dans une pile c'est le dernier arrivé qui est le premier servi. Ce sont deux exemples très simples de structures de données et il en existe de nombreuses autres : tableaux, tables de hachage, listes clé-valeur, graphes, etc.

Nous avons vu quelques exemples d'algorithmes, un *algorithme* est une abstraction qui décrit les étapes qu'il faut effectuer pour qu'à partir d'une donnée on en déduise un résultat. Le fait que l'algorithme doive être automatisé nécessite de s'intéresser aux cas limites (une pile est vide et on demande de retirer un élément), aux incidents comme une coupure dans le réseau, à la taille des mémoires, au temps de calcul, etc. Les algorithmes sont des objets qu'on peut manipuler : on peut combiner des algorithmes entre eux ou faire des opérations sur un algorithme par exemple le paralléliser.

Un concept, que j'ai effleuré en parlant des problèmes NP-complets et qui est utilisé pour faire des preuves d'indécidabilité ou pour construire des programmes, est celui de la *réductibilité*, qui est l'utilisation d'un algorithme résolvant un problème B pour résoudre un problème A, ce qu'explique la figure 4 ci-dessous :

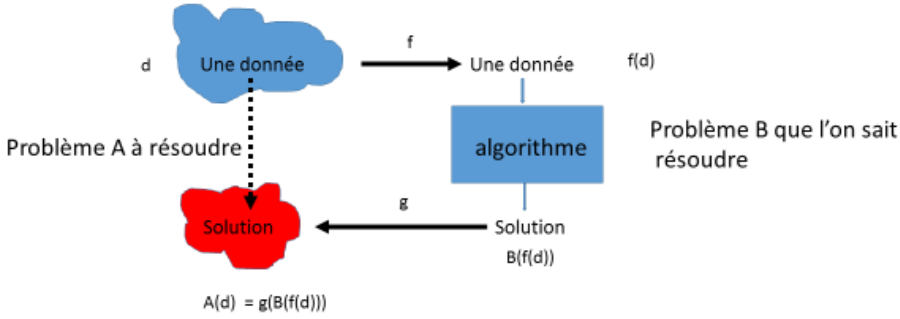

Figure 4 : Réductibilité d'un problème

Et il y a de très nombreuses manières de décrire un algorithme.

Une *procédure* en informatique est une manière particulière de décrire un algorithme dans certains langages. Les entrées et les sorties, les données et les résultats sont structurés.

Un *module* est un ensemble de procédures traitant des données communes. L'interface d'un module décrit les éléments du module, constantes, variables, types, fonctions, accessibles de l'extérieur du module, et qui sont utilisables dans un autre programme.

Une *classe* est une spécialisation de la notion de module. Les données et les procédures traitant les données sont réunies ensemble dans une classe qui constitue un nouveau type. On peut ainsi définir une classe Pile et un élément de cette classe est un *objet* qui est une pile. Et un objet informatique est une abstraction composée de données (les attributs de l'objet) et de méthodes pour les manipuler.

Un *agent* est une entité computationnelle active, avec une identité persistante, qui peut percevoir, raisonner et agir dans son environnement et qui peut communiquer notamment avec d'autres agents.

Ces abstractions sont les briques de base qui permettent de construire différents langages de programmation que l'on peut regrouper suivant certains paradigmes de programmation.

La *programmation procédurale* est une abstraction, qui consiste à concevoir un programme comme composé essentiellement de procédures. La *programmation par objets* consiste à concevoir un programme comme composé d'objets informatiques. Il existe d'autres paradigmes de programmation qui sont des abstractions : la programmation logique ou par contraintes ou fonctionnelle.

Ce n'est pas fini, si les programmes construits suivant ces différents paradigmes sont complexes, ils doivent être munis d'une architecture de même que les systèmes qui les mettent en œuvre. Une architecture de système est elle-même une abstraction, une architecture peut être centralisée ou répartie, une architecture multi-agents est composé d'un ensemble d'agents qui communiquent entre eux suivant certains protocoles, et un agent peut-être lui-même composé d'un ensemble d'agents, plus simples, coopérant, etc.

Un processus d'abstraction conduit naturellement à la notion de décomposition hiérarchique (une procédure est composée de procédures qui elles-mêmes sont composées de procédures, …) et à la notion de décomposition en couches.

Par exemple, quand on navigue sur le web, un navigateur utilise un protocole (HTTP ou HTTPS) pour envoyer ou recevoir des messages, le serveur de messagerie utilise le protocole SMTP, un autre protocole FTP permet de transférer des fichiers etc. Dans le modèle TCP/IP, Internet est structuré en 4 couches : accès (physique + liaison données), internet (réseau), transport (transport), application (session, présentation, application). Tout ceci est organisé par des protocoles qui permettent de gérer les relations entre les couches.

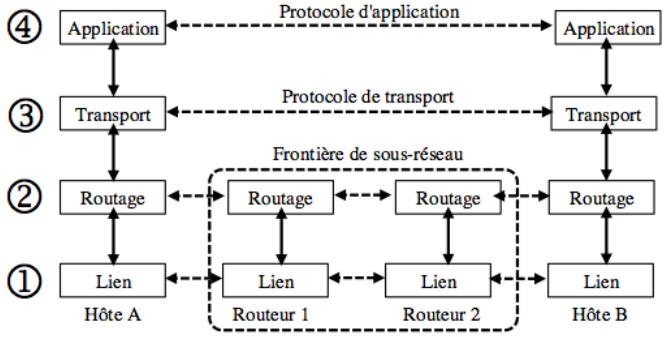

Figure 4 : Protocole TCP/IP internet

Les abstractions informatiques dont nous venons de donner quelques exemples doivent, si on veut les utiliser pour modéliser un système que l'on veut simuler ou un problème que l'on veut résoudre, pouvoir être interprétées par une machine physique : mécanique, électrique, quantique, biologique, …

1.5. Un espoir ?

Pratiquement toutes les sciences ont une activité de modélisation computationnelle et, plus généralement, ce mode de pensée informatique est présent partout, il est donc nécessaire de comprendre en quoi consiste cette modélisation informatique qui met en œuvre une pensée informatique.

Des scientifiques (prix Nobel, professeurs au Collège de France, Académiciens,…) ont décidé de poser des questions aux candidats à l'élection présidentielle. La liste des questions et la liste des réponses sont sur le site de l'ENS http://science-et-technologie.ens.fr/

Voici une question concernant l'informatique :

III.1 Un enseignement de spécialité Informatique et sciences numériques a été introduit en 2012 en terminale scientifique. En 2015, le gouvernement a décidé sa généralisation et introduit de nouveaux programmes de science informatique du primaire au baccalauréat. Néanmoins il n'existe ni CAPES ni agrégation spécifiques. Comment comptez-vous développer la formation des élèves, le recrutement et la formation permanente de leurs enseignants dans ce domaine ?

Cette question de l'enseignement de la science informatique pour tous est posée depuis plus de 50 ans par des informaticiens. En 1978, le rapport Simon sur l'informatisation de la société demandait la création d'un CAPES d'Informatique, mais nous ne voyons rien venir : ceci est un peu désespérant !

2. L'informatique à Montpellier

Établir une nouvelle discipline scientifique dans une université, est un processus long, complexe et difficile. Ce processus est encore plus difficile lorsqu'il s'agit d'une discipline jeune comme l'informatique que certains, heureusement de moins en moins nombreux, refusent de reconnaitre comme une science bien que l'informatique soit la science au cœur du numérique : *sans informatique pas de numérique !*

Ce processus implique de créer des cursus, de trouver des machines, de trouver de la place dans des bâtiments existants, ou d'en faire construire de nouveaux, de créer un laboratoire de recherche et de le faire reconnaitre et l'essentiel, qu'il y ait des personnes : des universitaires, des administratifs, des techniciens, des chercheurs pour mettre en place les structures et assurer leur fonctionnement. Il faut donc des postes, et la lutte (et c'est un euphémisme) pour les postes structure en grande partie la vie d'une université.

On peut distinguer trois étapes dans l'établissement de l'informatique à Montpellier. L'étape des pionniers, une vingtaine d'années entre 1960 et 1980. L'étape de la consolidation qui est celle de la création d'un cursus complet d'informatique à la faculté des sciences et de la création du CRIM (Centre de Recherche en Informatique de Montpellier), laboratoire associé au CNRS. La 3° étape commencée en 1992 par la création du LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier), résultat de la fusion de deux laboratoires le CRIM et le LAMM (Laboratoire d'Automatique et de Microélectronique de Montpellier), ne sera pas détaillée car il est facile de trouver des informations sur cette dernière période.

2.1. Les pionniers 1960-1980

En simplifiant on peut distinguer quatre types de pionniers en informatique.

Des physiciens ou des chimistes ayant des besoins de calcul, ce fut le cas à Montpellier de Jean Falgueirettes, cristallographe.

Une deuxième catégorie est constituée de mathématiciens attirés par tout ce qui concerne le calcul comme d'autres sont intéressés par des problèmes de physique ou de biologie, ou par des mathématiciens ayant des calculs importants à faire, ce fut le cas à Montpellier d'Yves Escoufier, statisticien.

Une troisième catégorie de pionniers est composée d'universitaires d'une autre discipline pensant que l'informatique est une discipline tellement importante qu'une faculté des sciences doit avoir un département informatique et à Montpellier ce furent un mécanicien Olivier Maisonneuve et un mathématicien Bernard Charles.

Il y eut aussi, à Montpellier, parce que l'informatique a démarré plus tard que dans d'autres universités, un quatrième type de pionniers : des informaticiens, comme Jean Bosmorin ou Marc Nanard, ayant passé une thèse ailleurs et se retrouvant isolés.

1960. Tout commence cette année-là, avec la création d'un centre de calcul, rue de l'Université, équipé d'un IBM 1620, pour les travaux de Jean Falgueirettes sur les structures cristallines qui nécessitaient des calculs importants. Bernard Filliatre, qui s'occupera quelques années plus tard de la MIAGE (Maitrise d'Informatique Appliquée à la Gestion) est le premier technicien embauché dans le centre de calcul.

1966. C'est l'année de la création de l'IUT avec pour seul département celui d'Informatique. Jean Falgueirettes assume les fonctions de directeur de l'IUT et du département informatique pendant deux années scolaires (plus tard J. Falgueirettes créera une filière informatique dans le centre associé du CNAM). L'IUT est hébergé Rue du Cardinal de Cabrières. L'arrivée du 360/40 au centre de calcul et le transfert du 1620 à l'IUT ont été concomitants à l'automne 1966.

1968. Yves Escoufier (un statisticien qui deviendra président de l'Université) et Robert Reix (un gestionnaire qui créera l'IAE) co-dirigent le département informatique.

1969. Création de la MIAGE (20 étudiants) avec B. Charles pour directeur et J. Falgueirettes comme professeur d'informatique. Intégrée à l'ISIM (Institut des Sciences de l'Ingénieur de Montpellier) en 1970, cette filière deviendra la filière IG de Polytech en 2003 (avec deux ingénieurs en 71, 13 en 72, 24 en 73, … 40 maintenant). C'est aussi l'année du recrutement de Jean Bosmorin en 1969 comme Maitre-Assistant à l'IUT.

1970. Marc Nanard, qui était assistant à Lille, rejoint l'IUT, et Jocelyne Nanard l'ISIM.

1971. Yves Césari, un informaticien théoricien, est le premier professeur d'informatique recruté à Montpellier (il quittera quelques années plus tard l'IUT pour Paul Valéry). O. Maisonneuve, dont le rôle sera important y compris au niveau national puisqu'il sera président de la Commission Pédagogique Nationale des départements d'informatique, est recruté lui aussi comme professeur. Il crée un DEA d'informatique, Diplôme d'Études Approfondies, qui est nécessaire pour s'inscrire en thèse. Ce DEA, qui fonctionna un an, permit aux nombreux assistants, non informaticiens mais « convertis » à l'informatique dans le cadre d'un stage national organisé à Montpellier (H. Bétaille, J. Boyat, Danièle Hérin, A.-M. Massotte, J.-P. Magnier, …), de préparer une thèse. B. Filliâtre soutient une thèse d'Etat et est recruté comme professeur à l'ISIM.

1974. Création du CRIG, un laboratoire universitaire composé de 3 équipes, Informatique (J. Falgueirettes et « sa » secrétaire Renée Bessière qui deviendra la secrétaire du CRIM), Statistiques (Y. Escoufier, retour de Montréal en octobre date à

laquelle il crée une option analyse des données en DEA de maths et qui créera un laboratoire de statistiques à l'ENSAM avant de devenir président de l'UM2) et Gestion (R. Reix qui créera l'Institut d'Administration des Entreprises). Le CRIG, qui est plus une structure administrative qu'un laboratoire de recherche sauf en gestion, dispose d'une salle dans le département info de l'IUT et d'un SOLAR 16 (qui n'a pas que très peu servi).

1975-80. Cette période voit une accélération du développement de l'informatique dans les deux pôles que sont l'IUT et l'ISIM. Deux professeurs d'informatique, spécialistes des systèmes, sont recrutés, Jean Ferrié à l'ISIM et Claude Boksenbaum à l'IUT. Marc Nanard soutient une thèse d'Etat. L'IUT s'est équipé d'un SIEMENS 4004 (360/30) et le centre de calcul d'un 360/65 installé dans un bâtiment spécialement construit sur le campus de la Colombière.

Résumons la situation à la fin des années 70. Il y a une douzaine d'informaticiens qui sont dispersés thématiquement, géographiquement (campus Triolet, Saint Priest, La Colombière et UPV) et institutionnellement (UM2 et UPV).

Les deux places fortes sont : le département informatique de l'IUT (avec C. Boksenbaum, J. Bosmorin, M. Nanard, J.-M. Boë, H. Bétaille, J. Boyat, …) et la filière FIG de l'ISIM (avec J. Ferrié, B. Filliâtre, J. Nanard, …), il y a aussi quelques personnes autour de Y. Césari à l'UPV.

Il n'existe pas de cursus d'informatique à la Faculté des Sciences et pas de laboratoire de recherche en informatique.

2.2. La consolidation 1980-1992

Ces années sont marquées par la création et le développement d'un laboratoire de recherche, le CRIM, et d'un cursus complet à la faculté des sciences. Pour accélérer le développement de certaines disciplines dans des universités de province, la mission à la recherche du Ministère, dirigée par François Davoine, suscita le transfert d'équipes de recherche d'universités pléthoriques (en fait, parisiennes …) vers des universités de province. Voulant quitter Paris VI, une partie de mon équipe a sauté sur l'occasion. Cependant, si Paris VI était pléthorique elle ne l'était pas en informatique … Paris VI ne voulant pas perdre un poste de professeur, trois postes d'assistants et un poste de chercheur au CNRS et ayant de plus peur que notre départ suscite un appel d'air (ce qui d'ailleurs se produisit) … notre transfert fut plus compliqué que prévu ! Notre mission était de créer un cursus complet d'informatique à la faculté des sciences (licence, maitrise, DEA, plus tard complété par un IUP et une formation doctorale) et un laboratoire d'informatique reconnu par le CNRS.

Le CRIM, Centre de Recherches en Informatique de Montpellier, fut créé en 1982. Il était composé de trois équipes : Algorithmique et graphes (J.-P. Bordat, A. Cazes, M. Chein, O. Cogis, M.-C. Vilarem), Combinatoire des langages formels (J.-M. Boë, Y. Césari, E. Lochard, C. Mallol, M. Vincent) et Logiciel de base (C. Boksenbaum, J. Ferrié, J. Nanard, M. Nanard). Le CRIM s'est développé rapidement, en particulier grâce à la création du GSDIALR (Groupement Scientifique pour le Développement de l'Intelligence Artificielle en Languedoc-Roussillon) qui permit le transfert, entre autres, de J. Sallantin, d'O. Gascuel, et J. Quinqueton, la création du GLIM (Groupe de Recherche en Linguistique, Informatique et Médecine) avec C. Baylon et P. Dujols associant les trois universités montpelliéraines, et la création de l'École doctorale SPI (Sciences Pour l'Ingénieur, devenue I2S Information Structures Systèmes).

Cette période se termine avec la fusion du CRIM et du LAMM, créé et dirigé par C. Durante, pour créer le LIRMM en 1991 (comme laboratoire universitaire) et officiellement en 1992 comme unité mixte de l'UM2 et du CNRS. G. Cambon, son premier directeur, sut gérer avec doigté la difficile fusion de deux laboratoires ayant des cultures différentes, et depuis le LIRMM s'est développé pour devenir l'un des plus importants laboratoires français en informatique, robotique et microélectronique, dont le rayonnement est international.

2.3. Aujourd'hui

Pour terminer, un mot sur la situation présente.

Enseignement : IUT (DUT ~130, licence pro ~80), Polytech (IG ~40 par promotion), Fac des Sciences : licence 1,2,3 ~ 400 étudiants, Master (avec 6 parcours et un Cursus Master en Ingénierie) plus de 300 étudiants.

Recherche : 15 équipes de recherche dans le département d'informatique du LIRMM, 200 chercheurs et doctorants.

Les domaines de recherche sont indiqués sur la figure 5 suivante en notant une composante importante concernant les interfaces à d'autres disciplines : agronomie, biologie, environnement, santé et sciences humaines.

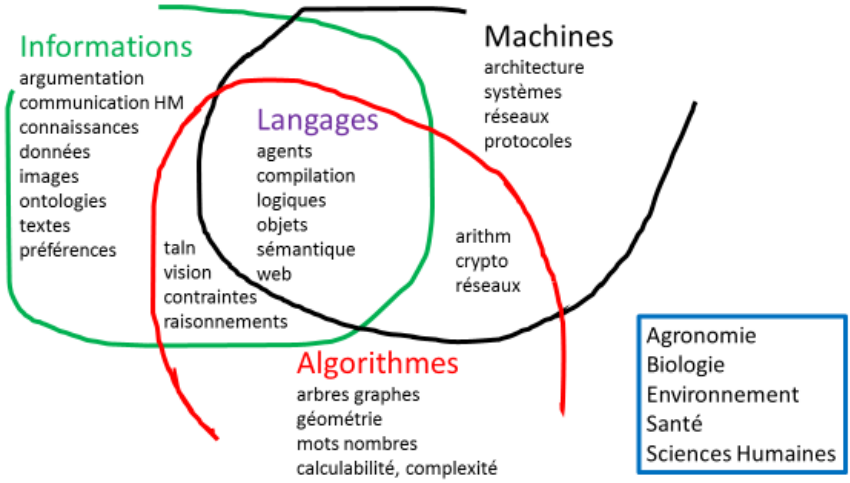

Figure 5 : Domaines de recherche du LIRMM

En guise de conclusion

Je n'ai pas utilisé souvent le terme « numérique » parce que l'informatique consiste à manipuler des symboles et que les nombres n'en sont qu'un cas particulier. Un ordinateur est une machine universelle pour manipuler des symboles. Mais aussi parce que je partage le point de vue de Maurice Nivat, un des pionniers de la science informatique, qui a dénoncé avec force dans une tribune libre du journal en ligne EPINET, l'usage du terme numérique : « Le mot numérique, écrit-il, est un cas typique de ce que Éric Hazan appelle « l'évitement des mots du litige » consistant, en changeant de vocabulaire, à faire disparaître des sujets qui fâchent, des sujets trop polémiques, des sujets dont on ne sait pas bien comment se dépêtrer. Nous n'avons

cessé depuis que je m'occupe d'informatique en 1967, écrit-il, de perdre du terrain, immédiatement occupé par les Américains… Dans ce tour de passe-passe, hélas, ce n'est pas seulement l'informatique qui perd c'est toute la science : quelques brillantes idées de start-up, que l'on dorlote et chouchoute tant qu'on peut, quelques licornes de plus, et c'est toute la France, nous dit-on, qui sortira de son marasme, qui oubliera le cauchemar du chômage et de la désindustrialisation, de sa dette et de son déficit pour retrouver sa place, au premier rang dans le monde. » Ce cri de colère de Maurice Nivat a sans doute joué un rôle dans la reconnaissance, et ceci jusqu'au ministère de l'éducation nationale, de l'enseignement supérieur et de la recherche, du fait que ***l'informatique est la science au cœur du numérique*** **!**

Contrairement à d'autres disciplines, l'Informatique à Montpellier n'a pas un long passé prestigieux … elle a plutôt un court passé, laborieux, mais aussi quelque peu glorieux, car il a fallu mener de nombreux combats pour installer l'informatique dans le paysage scientifique montpelliérain. Son présent, dans les différentes structures, me semble heureux et, comme nous avons parcouru le chemin permettant aux informaticiens de Montpellier de tenir fermement les deux bouts de la chaine de la recherche à l'enseignement, on peut être serein sur son avenir !

Pour finir, je vous conseille de lire le petit livre de Serge Abiteboul et Gilles Dowek « Le temps des algorithmes » (Le Pommier, 2017), il est lisible par tout le monde !