La traque des bugs en informatique

Pierre-Louis Curien (CNRS – Paris 7 – INRIA)

Merci à G. Berry, Y.Bertot, G. Gonthier, X. Leroy, J.-J. Lévy, A. Mahboubi pour leurs suggestions,

à Marc Bondiou et Fabienne Clérot pour cette initiative,

et à Li Shanshan pour la virtuosité dans la traduction

28/4/2015, **Café des Sciences, Wuhan**

L'informatique est partout

— world wide web : moteurs de recherche, vente en ligne, réseaux sociaux,...
— ordinateurs de bord, logiciels embarqués
— robotisation : des tâches ménagères, de la médecine

⋮

Mais au fait, c'est quoi, l'informatique ?

— De la physique et de la technologie : électronique, des transistors aux puces et maintenant aux nanotechnologies
— De l'ingénierie : du code, du code et toujours du code
— Une *science*, fille des mathématiques

Et cette science n'est pas ésotérique ! Elle est, telle la prose pour M. Jourdain, présente dans notre vie quotidienne. Quelques illustrations suivent.

Enjeux cognitifs de l'informatique

- s'organiser,
- hiérarchiser des données brutes,
- naviguer intelligemment au milieu d'une mer d'informations,
- identifier ou distinguer les choses,
- les nommer,
- repérer ce qui sert / ne sert pas,
- être contraint à la rigueur et à la précision.

Additions et soustractions

Lorsque l'on écrit (et calcule)

$$3 - (-5 - 7)\ ,$$

Il y a une *surcharge (ambiguïté)* entre l'usage du signe $-$ pour la soustraction ($3-x$) et pour l'opposé ((-5)), et des priorités *implicites* ($(-5)-7$)), ce qui donne

$$3-((-5)-7)$$

$\rightarrow$ Arbres de syntaxe abstraite.

$\rightarrow$ Code machine : par ex., pour additionner deux expressions e et e' :

```
calculer e et sauvegarder la valeur;
calculer e' et sauvegarder la valeur;
appeler l'additionneur
```

Cet exemple illustre la *chaîne de compilation*.

Diverses notions d'égalité

$3 + (5 + 7) = 15$	(le résultat)
$3 + (5 + 7) = 3 + 12 = 15$	(calcul par étapes, ou par réécriture)
$3 + (5 + 7) = (3 + 5) + 7$	(propriété – ici l'associativité)
7 "$=$" 111	(écriture du nombre 7 en binaire)
$4/5 = 0,8$	(calcul effectué / à effectuer)
$1 = 0,99999999999\ldots$	($\rightarrow$ approximation)

De Boole à Google

Les fonctions de recherche avancées de Google, Yahoo, Baidu, ... font intervenir les expressions booléennes.
Manipulations du même genre que (+/-) :

$$\texttt{non}\,(A\ \texttt{et}\ B) = (\texttt{non}\ A)\ \texttt{ou}\ (\texttt{non}\ B)$$

[George Boole (1815-1864), britannique, fut un des fondateurs de la logique mathématique moderne, et notamment de la notion d'algèbre de Boole.]

Leçons de géométrie

Les choses se font dans un certain ordre de **causalité** : il faut deux points pour tracer une droite, une droite et un point pour déterminer une perpendiculaire, etc...

Notion de propriété **invariante** (e.g. par rotation, ou symétrie), cruciale lorsqu'il s'agit de prouver la correction d'un programme.

De l'i-pode aux graphes

Classer l'information de manière arborescente, et y accéder par des voies différentes.

Partage de l'information : tel morceau de musique peut se trouver sous la rubrique d'un chanteur ou d'un genre musical $\rightarrow$ notion de **graphe**.

Analyse de dépendances

• Par le test : on fait varier un paramètre et on voit si cela influe sur le résultat (sur un tableur comme Excel, en changeant la valeur d'une variable, on peut observer quelles cases bougent et quelles cases ne bougent pas).

• Par la **preuve** : calcul de chemins dans des graphes

L'échec d'Ariane du 4 juin 1996 est imputable à des considérations de cet ordre.

Confidentialité, sécurité

Les membres d'un club ou d'une mailing liste veulent pouvoir garder un certain niveau de confidentialité dans les questions dont ils débattent.

Sécurité des transaction d'e-commerce, etc. . ..

$\rightarrow$ Protocoles cryptographiques

Nous parlerons du bug Heartbleed (2014).

Situations de compétition (race conditions)

Une race condition est un défaut dans un système, qui conduit à un résultat erroné parce que l'ordre d'exécution attendu des commandes n'est pas respecté du fait de vitesses différentes d'exécution.

Exemple (Monsieur Boole toujours à l'honneur !) : Soit une porte logique `ET`, dont les deux entrées sont nourries par un signal A sur l'une de ses entrées, et (`NON` A) sur l'autre entrée. En théorie, la sortie (A `ET` (`NON` A)) ne devrait jamais être `VRAI`. **Mais** si A change, disons de `FAUX` à `VRAI`, et si ce changement met plus de temps à être propagé sur la seconde entrée que sur la première, les deux entrées se trouvent `VRAI` pour une brève période et la sortie peut alors devenir `VRAI` si la porte `AND` se dépêche !

Le bug du Therac-25 est imputable à une situation de compétition.

Calculs exacts, calculs arrondis

• calculs exacts : trouver les décimales successives du nombre π est par nature (asymptotiquement) exact

• calculs approchés (ou arrondis) : additionner des approximations x' et y' de deux nombres x et y tend à nous éloigner davantage du résultat exact de l'addition $x + y$.

Le bug du Patriot est imputable à des erreurs d'arrondi cumulées.

`http://en.wikipedia.org/wiki/List_of_software_bugs`
(extraits)

- Médical : Appareil Therac-25 de thérapie par radiations (années 1980)
- Militaire : Patriot (1991)
- Espace : Ariane 501 (1996)
- Date : “bug de l’an 2000”(beaucoup de bruit ! évité à temps)
- Cryptage : Heartbleed (découvert en 2014, code datant de 2012)
- Date : Bug du Zune (la tentative d’Ipod de Microsoft) (31 décembre 2008)
- Algorithme de tri Timsort (Android, Java, Python) (2015)

Mais au fait, qu’est-ce qu’un bug (parfois “francisé” en “bogue”) ? C’est une erreur de programmation non intentionnelle.

Le bug du Therac-25

Cet appareil fonctionait en deux modes très distincts : rayons de faible et forte intensité, respectivement, ce dernier mode étant couplé avec la mise en place sur le parcours du faisceau d'une plaque rotative permettant de répartir le rayonnement. Le passage entre ces deux modes était contrôlé "manuellement" dans les versions précédentes de l'appareil. Avec le modèle Therac 25, le contrôle est devenu (seulement) logiciel.

Les accidents (au moins 6 entre 1985 et 1987, causant de graves irradiations de patients et entraînant trois décès) se sont produits sous le mode "haute intensité" mis en route de manière non voulue, et sans la plaque de répartition mise en place.

Le dysfonctionnement ne se produisait que lorsque l'opérateur rentrait la commandes d'activiation du mode "haute intensité" et se ravisait *aussitôt* pour activer l'autre mode, déclenchant ainsi une "race condition".

Le bug du Patriot

Echec d'un système de missile sol-air (premièrre guerre du Golfe). Bug dû au cumul de petites erreurs d'arrondi qui allaient toutes dans le même sens. Plus précisément, après une première détection radar d'un missile ennemi (Scud), Patriot anticipait des points et horaires de passage de la trajectoire du Scud, jusquà son arrivée sur la zone de portée de ses anti-missiles, et ce sont ces valeurs qui étaient arrondies. En fonctionnement continu, au bout de 100h, l'erreur en temps cumulée atteignait 1/3 sec pour une vitesse 1,7km/s du Scud, qui est passé à plus de 500 m de là où Patriot l'attendait.

le bug du Patriot a provoqué la mort de 28 personnes à la base US de Dhahran en Arabie Saoudite (25 février 1991). Par contre, les israéliens qui utilisaient aussi des Patriot pour neutraliser les Scud irakiens rebootaient tous les matins leurs Patriot, et n'ont jamais eu cette accumulation de petites erreurs.

Le bug d'Ariane

Le 4 juin 1996, la fusée Ariane 5 (vol inaugural) explose 40 secondes après le décollage. Cet échec a été causé par une exception logicielle dans le calculateur interne de la centrale à inertie (SRI), qui a provoqué l'arrêt du SRI, puis des moteurs, et l'explosion. Une opération de conversion (flottants 64 bits $\rightarrow$ entiers 16 bits) n'a pu se faire, car la valeur entière du paramètre (accélération horizontale de la fusée, 5 fois plus forte que pour Ariane 4) excédait ce que peut exprimer un nombre entier à 16 bits. Ces instructions, *contrairement à d'autres, voisines dans le code*, n'étaient pas protégées. Ce module de logiciel calcule des résultats significatifs avant le décollage seulement. Mais surtout, il était *inutile* pour le lanceur Ariane 5.

La détection après coup de l'erreur logicielle précise a mobilisé les techniques les plus avancées d'analyse des programmes développées par la communauté des chercheurs (analyse d'alias, analyse statique).

D’Ariane à Boole

Si cette variable avait été protégée,
ou si le format de données avait été adapté aux valeurs prises par cette variable,
ou si ce composant avait été programmé pour cesser son activité après le décollage,
ou si ce composant avait été tout simplement supprimé,
ou si le second ordinateur (de secours) n’avait pas été programmé pour exécuter exactement la même séquence d’instructions,

alors l’explosion aurait pu être évitée.

Heartbleed en bande dessinée (`https://xkcd.com/1354/`)

Heartbleed est une vulnérabilité logicielle présente dans la bibliothèque de cryptographie OpenSSL qui permet à un “attaquant” de lire la mémoire d’un serveur ou d’un client pour récupérer, par exemple, les clés privées utilisées lors d’une communication via le protocole Transport Layer Security (TLS).

OpenSSL est utilisé par une proportion importante (environ 2/3) de serveurs de vente en ligne, d’impôts en ligne, etc. . .. Deux ans se sont écoulés entre l’introduction (involontaire) de l’erreur et sa découverte. Parmi les dégâts faits par le bug : vol de 900 identifiants de contribuables canadiens sur le site de la Canada Revenue Agency.

La correction de l’erreur a été rapide (quelques jours).

Le bug du Zune (`http://bit-player.org/2009/the-zune-bug`)

Le progamme fautif est le suivant :

```
year := 1980;
while (days > 365)
  if (IsLeapYear(year))
    if (days > 366) days := days − 366; year := year + 1;
  else days := days − 365; year := year + 1;
```

Le but du programme est de convertir un nombre de jours absolu, compté depuis le 1er janvier 1980, en un coupe formé d'une année, et d'un nombre de jours relatif depuis le premier janvier de cette année. Par exemple, pour 380, la réponse est le couple (1981, 14), car

- 1980 était une année bissextile et
- parce que 14 = 380-366

Le programme a buggé le 31 décembre 2008. Pourquoi ?

Le bug du tri Timsort (plus technique) (envisage-project.eu/proving-android-java-and-python-sorting-algorithm-is-broken-and-how-to-fix-it/)

• Algorithme (2002) combinant “insertion” et “merge”, basé sur un invariant. Il délimite des segments successifs s_i où la liste est déjà ordonnée, complétés le cas échéant par “insertion”. Après l’ajout de chaque segment : “merge” jusqu’à rétablissement de l’invariant (local). Lorsque toute la liste a été scannée : “merge” final.

• 2015 : Des chercheurs ont montré que algorithme est incorrect, malgré son look “logique”. L’invariant $P(max)$ n’implique pas l’invariant global $(\forall i \leq max\ P(i))$ (qui garantit que le tableau des longueurs et des positions intitiales des segments reste borné). Ils ont trouvé un contre-exemple provoquant un plantage du programmme (dépassement de borne). Ils ont montré $((P(max)$ et $P(max-1)) \Rightarrow \forall i \leq max\ P(i))$, et conséquemment corrigé l’algorithme.

• Invariant : $P(i)$: $l(s_{i-2}) > l(s_{i-1}) + l(s_i)$ et $l(s_{i-1}) > l(s_i)$ (l désigne la longueur)

• La réaction des développeurs Java a été d’allouer plus d’espace au tableau des longueurs des segments, et non pas d’adopter le code corrigé... (“perhaps the JDK maintainers did not bother to read our report in detail, and therefore don’t trust and understand our fix. After all, Open Java is a community effort, largely driven by volunteers with limited time.”)

Certification

Les informaticiens ont développé des prouveurs (automatisés) et des assistants de preuve (interactifs) comme **Coq**, développé en France par plusieurs équipes de chercheurs (**INRIA**, universités, CNRS) depuis le début des années 90.

• Certification de A à Z en Coq d'un compilateur C, i.e. le code en langage machine exécuté est conforme à la sémantique du programme compilé (Xavier Leroy, INRIA, et collaborateurs, 2013).

Formalisation de preuves mathématiques

• Le même logiciel Coq a été aussi utilisé pour vérifier la correction de **théorèmes**. Certaines constuctions mathématiques, comme la classification des groupes finis simples, sont tellement complexes qu'il est difficile de tout vérifier par soi-même : c'est un consensus collectif qui a permis d'annoncer la classification en 1983 (plusieurs dizaines de milliers de pages, env. 500 articles, plus de 100 auteurs). Une pièce importante de cet édifice a été formalisée, i.e., vérifiée formellement en Coq (Georges Gonthier, Microsoft Reserarch, et collaborateurs, 2012).

Avenir des méthodes formelles

Les méthodes formelles gagnent du terrain !

L'usage industriel systématique des méthodes formelles n'est pas encore gagné :

• Ces outils puissants restent encore relativement difficiles d'accès. On y travaille !

• Les industriels trouvent ces méthodes coûteuses (en temps de travail extrêmement qualifié), et leur préfèrent le plus souvent des jeux de tests. Nous comptons sur les start-up créées par les jeunes chercheurs formés aux assistants de preuve pour faire évoluer les choses.