Comment faire des choses intelligentes avec des machines stupides

Gérard Berry

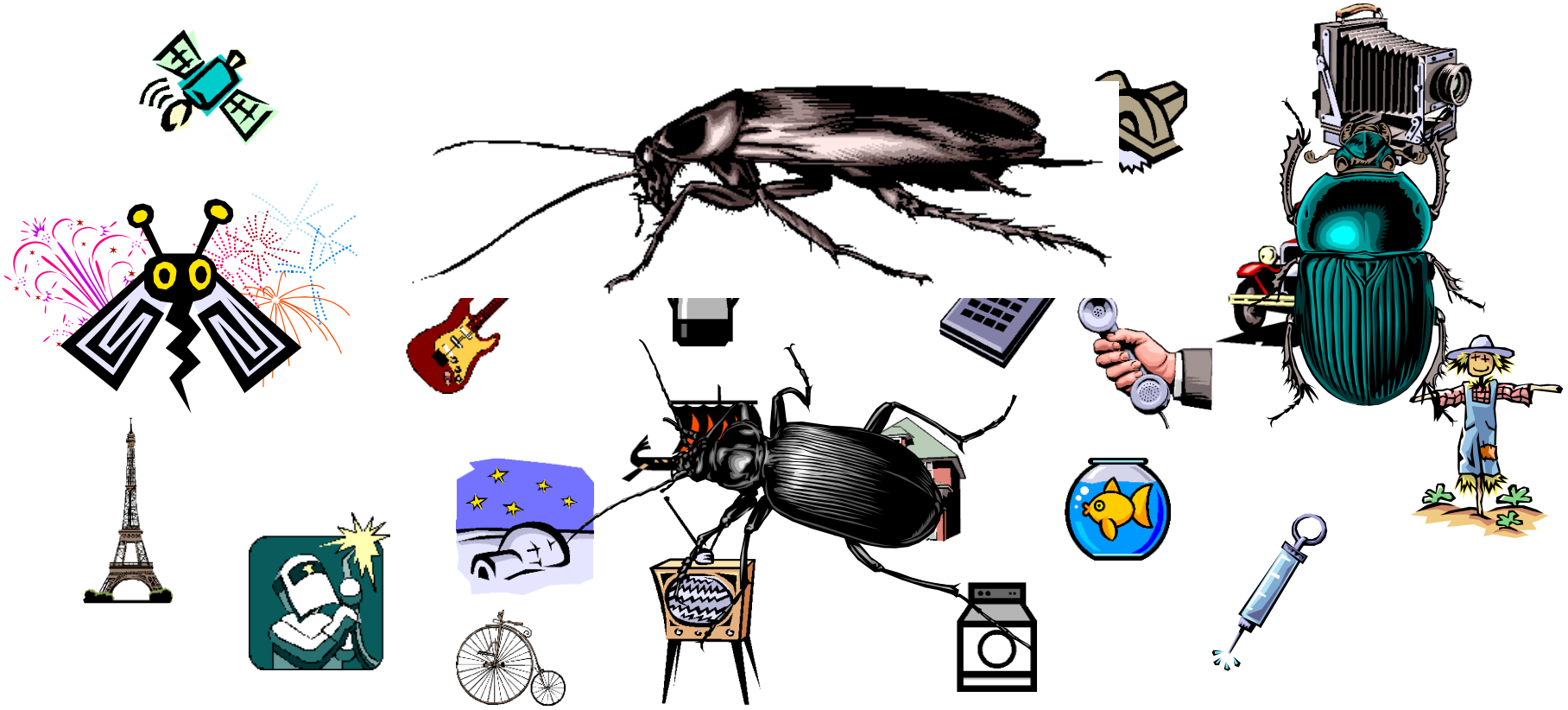
Collège de France
Chaire Algorithmes, machines et langages

http://www.college-de-france.fr/site/gerard-berry
gerard.berry@college-de-france.fr

SystemX, 12 mars 2015

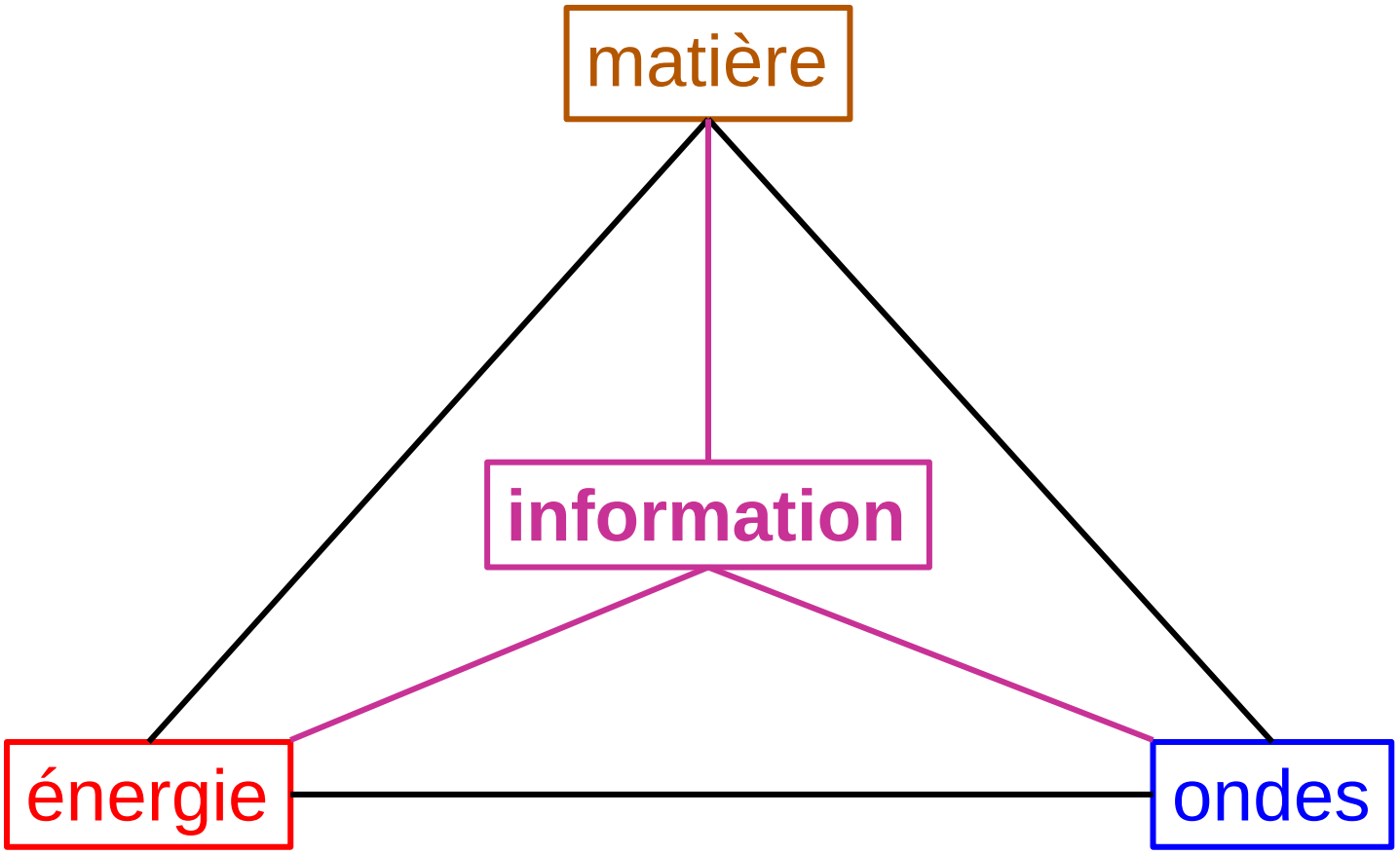
COLLÈGE
DE FRANCE
1530

Alerte aux pucerons



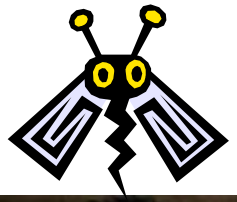
Infestation massive par pucerons enfouis partout
et branchés sur le Web ...
... mais de plus en plus d'applications critiques

L'irruption d'un intrus



L'information, ça ne pèse pas, ne sent pas, ne brûle pas, ça se conserve, se recopie et se transmet parfaitement,...

Un gouffre à combler

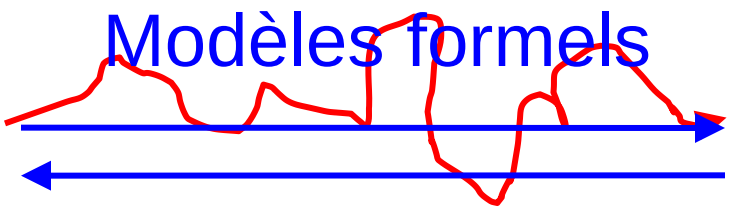


Intuition

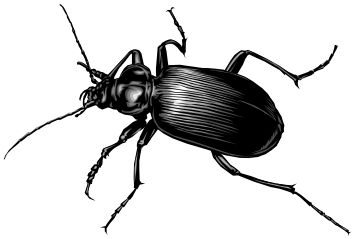
Rigueur

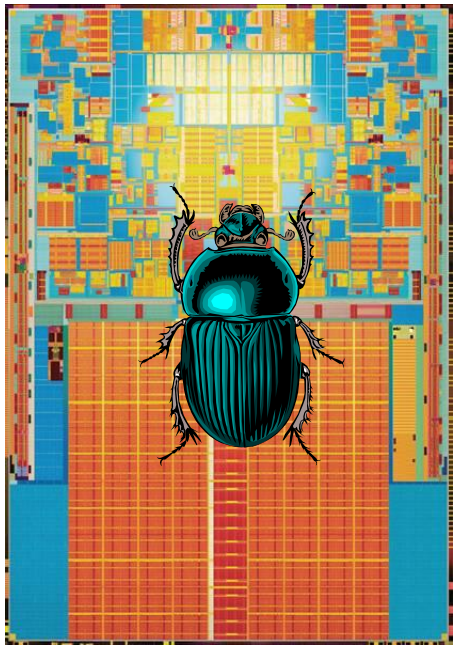
Lenteur

Modèles formels



Maîtrise ?



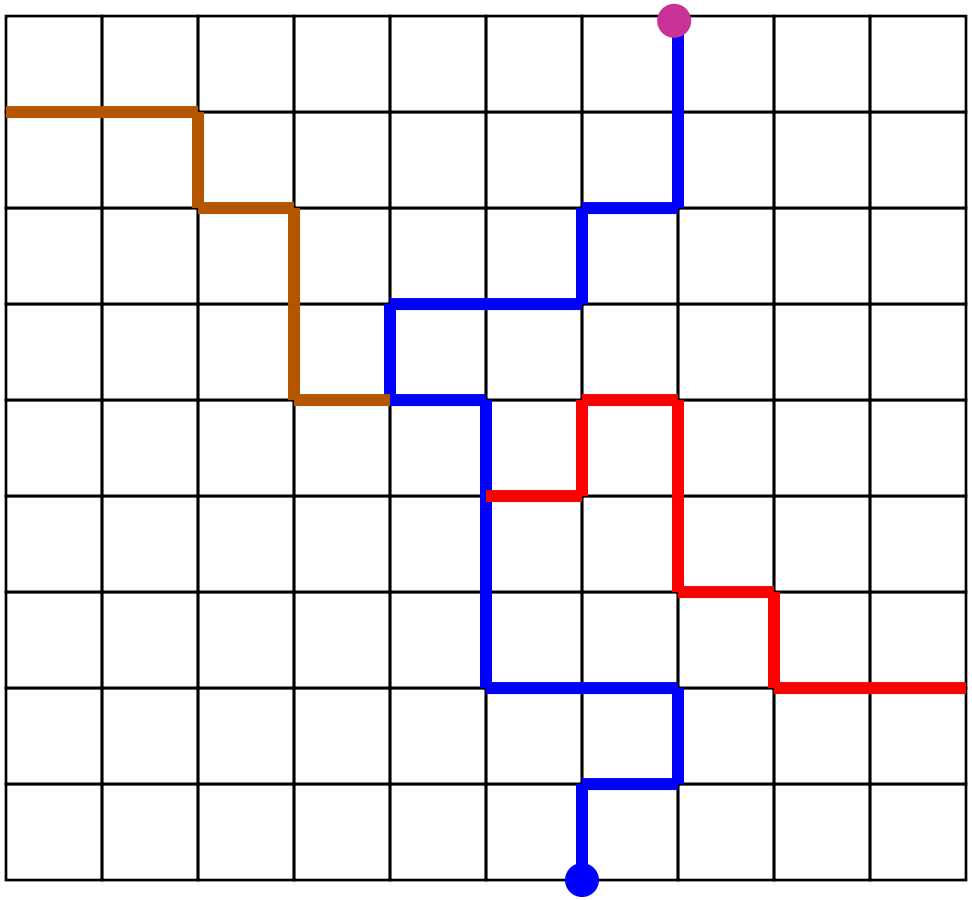


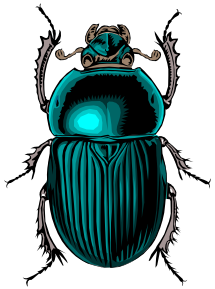
Rapidité

Exactitude

Stupidité

Ordinateur = amplificateur d'erreurs !

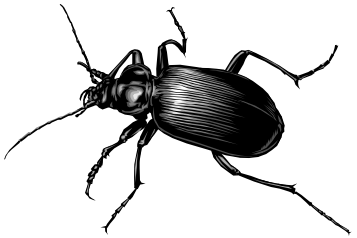




TDGGTDTTGDDTGDGT

TDGGTDTDGDDTGDGT

TDGGTDTTGTDTGDGT



Parallélisme : deadlock

Lise et Laure

Parallélisme : famine

Lise, Laure et Manon

Exemples de bugs ou mauvais designs

- Spatial : Ariane 501, Pathfinder, ...
- Automobile : ouvrants (Mercedes), freins (Volvo),
  contrôle moteur (Toyota), contact (Citroën), etc.
- Instrumentation de la ville : horaires bus (RATP)
- Systèmes de réservation de trains ou avions (très) mal foutus
- ...

Une question de responsabilité : si on a un accident à cause d'un bug informatique, qui est responsable ? Le conducteur ou le constructeur ? cf Toyota, Volvo, etc.

Applications de méthodes formelles

1990+ : RER et métros (B → Event-B, Abrial)

1995- : Esterel / SCADE 6 en avionique, ferroviaire, etc.

2004- : seL4, micronoyau formellement vérifié

200x : protocoles CMM7 (Trusted Logic)

2008- : CompCert (X. Leroy), compilateur C prouvé en Coq

seul à résister aux tests aléatoires de CSmith

2014 : vérification et génération du noyau Minix (Prove&Run)

Vérification formelle d'algos distribués,
de protocoles de sécurité (SSL, etc.)

Model-checking dans toute la chaine CAO de circuits

Le test peut trouver les bugs mais ne montre pas leur absence
Les méthodes formelles sont efficaces et rentables
si appliquées tôt avec des ingénieurs formés

Sûreté et sécurité

Beaucoup de problèmes de sécurité viennent de tout petits bugs (même non fonctionnels) des logiciels

- Les protocoles de sécurité ne sont pas humainement vérifiables

- Les systèmes modernes ne pourront être sûrs que s'ils sont fabriqués à partir de composants garantis sûrs
  - vrais processus de qualifications
  - méthodes formelles partout où c'est approprié pour la sûreté et la sécurité

Le rôle central du temps

- L'informatique distribuée traditionnelle est souvent asynchrone, mais c'est souvent une erreur !
  - ex. : distribution dans les systèmes cyber physiques
- Distribuer le temps est plus facile que distribuer les algorithmes
  - PTP / NTP : synchro des ordinateurs à qq millisecondes ($\rightarrow \mu s$)
  - Spanner, base de donnée répliquée Google
- Programmer synchrone remplace la complexité du non-déterminisme par la simplicité du déterminisme
  - langage synchrones : Esterel / SCADE / Signal $\rightarrow$ SCADE 6
  - distribué synchrone : Ptolemy II, PTides (E. Lee, Berkeley)
- Synchrone et asynchrone peuvent collaborer
- Mais les modeleurs EDO ne comprennent pas bien le temps...

cf http://www.college-de-france.fr/site/gerard-berry/course-2014-03-19-16h00.htm

Conclusion

- Les difficultés intrinsèques de l'informatique distribuée et temporelle sont largement sous-estimées
- Or, son rôle est critique dans tous les systèmes modernes
- Les méthodes traditionnelles n'arriveront pas à garantir sûreté et sécurité même au niveau le plus élémentaire
- Mais les alternatives formelles deviennent disponibles et efficaces
- Comme d'habitude, la clef est dans la formation des ingénieurs et surtout la prise de conscience des décideurs