

Où en est-on de la dissipation du calcul ?

Retour à Bennett

What about the "dissipation of computation" question ?

A return to Bennett

Philippe Matherat

GET - ENST - Comelec / CNRS - LTCI - UMR 5141

<http://perso.enst.fr/~matherat>

Version publiée dans la revue *Annales des télécommunications*

Volume 62, n° 5-6, mai-juin 2007, pages 690 à 713

<http://at.revuesonline.com/article.jsp?articleId=10174>

Résumé

Pour un circuit électronique, la totalité de l'énergie électrique consommée est transformée en chaleur (*dissipation*). Un résultat de Bennett de 1973 montre que l'énergie minimale dissipée ne dépend pas de la complexité du calcul en temps, mais de la taille du résultat, qui peut être très faible. Pourtant, les puces actuelles dissipent proportionnellement à la complexité des calculs, et ce paramètre est celui qui freine le plus la miniaturisation et l'augmentation de performances des appareils mobiles. Pourquoi vivons-nous cette contradiction ? Nous proposons de relire le papier de Bennett pour y chercher un renouvellement des questions fondamentales qu'il peut soulever. Nous suggérons un lien avec la nature du temps et la synchronisation dans les systèmes distribués. Ce texte est une transcription de l'exposé donné au séminaire MIR (Mathématiques pour l'Informatique et les Réseaux) de l'ENST, le 9 mars 2006.

Abstract : In an electronic computing circuit, all the energy delivered by the electric power supply is converted into heat (dissipation). A result from Bennett in 1973 showed that the minimal amount of dissipated energy is not linked with the time-complexity of the computation, but rather with the size of the result, which can be very low. However, real chips are dissipating proportionally to the time-complexity, and this is the major problem challenging the increase of performance and the miniaturization of mobile devices. Why are we confronted to this contradiction ? We return to the reading of Bennett's paper in order to look for a renewal of fundamental questions it can rise. We suggest a link with the nature of time and with synchronization in distributed systems. This text is a transcript of the talk given at the MIR seminar (Mathématiques pour l'Informatique et les Réseaux) in ENST, the march 9th 2006.

Où en est-on aujourd'hui de la question de la *dissipation du calcul*? Pourquoi les machines à calculer devraient-elles dissiper, c'est-à-dire transformer de l'énergie électrique en chaleur? Nous sommes en fait dans une situation curieuse :

- D'une part, depuis un papier de Bennett de 1973 [1], la communauté scientifique s'accorde généralement pour dire qu'il n'est pas nécessaire de dissiper pour calculer, et qu'on peut réaliser des machines à calculer réversibles. (En fait, elle doivent dissiper proportionnellement à la taille du résultat, ce qui est en général très faible.)
- D'autre part, les puces électroniques, qui jouent un rôle de plus en plus important dans notre vie quotidienne et dans nos économies modernes, dissipent proportionnellement à leur activité, c'est-à-dire à la complexité des calculs, ce qui apparaît comme contradictoire avec ce résultat de Bennett.

Or, il est clair que cette question est liée à un enjeu considérable du point de vue économique et industriel, car en effet la dissipation est un inconvénient majeur des appareils électroniques, qui contraint fortement les possibilités de ces technologies :

- Elle entraîne des problèmes d'évacuation de la chaleur (radiateurs, ventilateurs, volume, poids, bruit, etc.).
- Elle entraîne des problèmes d'approvisionnement en énergie, qui a un coût non négligeable, qui est associé à ses conséquences pour l'environnement, et qui est associé à des inconforts (encombrement des adaptateurs-secteur, des fils, des piles).
- Les appareils mobiles souffrent d'une autonomie restreinte et/ou de l'excès de poids des piles ou des batteries. En fait, toute conception d'appareil mobile est un compromis entre les deux contraintes opposées que sont la grande autonomie et le faible poids.
- Les piles et les batteries posent des problèmes de recyclage, à cause de la pollution chimique potentielle.
- La dissipation est réellement aujourd'hui ce qui limite la performance en terme de puissance de calcul. Par exemple pour les ordinateurs portables, les ingénieurs sont obligés de restreindre les fréquences d'horloge, afin de ne pas être obligés d'ajouter des ventilateurs trop encombrants, et des batteries trop lourdes.
- Dans les récents dispositifs sans contact et sans source d'énergie (cartes à puce sans contact, *tags RFID*), les puces sont alimentées par le champ électromagnétique qui sert au dialogue avec l'appareil lecteur. La consommation électrique de ces puces doit être extrêmement faible, alors qu'on leur demande de faire des calculs de plus en plus complexes, par exemple lorsqu'on leur ajoute des capacités cryptographiques afin de pouvoir les authentifier. Plus généralement, toute conception électronique devient un compromis sévère entre dissipation et complexité des calculs.

Alors, comment se fait-il que, devant une question aussi cruciale, on puisse se contenter de réponses aussi discordantes (d'une part celle de Bennett, d'autre part celle de la pratique industrielle)? Il faut croire que quelque chose résiste, qu'on n'a pas tout compris!

Nous ne prétendons pas pouvoir proposer ici des pistes pour trouver *ce qu'on n'aurait pas encore compris*, mais nous allons présenter une relecture du papier de Bennett, afin de

montrer ses subtilités, et afin d'essayer de mettre en lumière ce qu'il nous dit et ce qu'il ne peut pas nous dire. Nous tenterons d'éclairer ce que pourrait être une recherche visant à sortir de cette contradiction, en nous laissant guider par exemple par les trois questions suivantes :

- Est-il possible de réaliser des calculs non-dissipatifs (c'est-à-dire de mettre concrètement en oeuvre la construction de Bennett) ?
- Est-il possible de prouver que, au contraire, une telle réalisation restera toujours impossible en raison d'une obstruction de nature fondamentale ?
- Est-ce indécidable ? Ou bien est-ce une question mal posée ? Quelle autre question poser à la place ?

Le texte qui suit comporte 3 parties. La deuxième est une présentation du papier de Bennett, vulgarisée et illustrée par des schémas. Elle est encadrée d'une introduction et d'une conclusion de notre cru.

I. Introduction : dissipation du calcul

I. 1. Pourquoi faudrait-il dissiper pour calculer ?

Les ordinateurs chauffent. Pourquoi ? Si nous étudions le bilan énergétique d'une machine à calculer, nous constatons qu'elle consomme de l'énergie électrique et produit de la chaleur (figure 1).

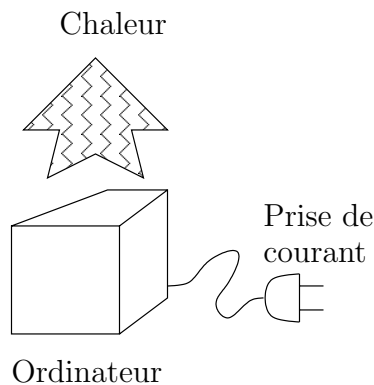


FIG. 1 – Bilan énergétique d'un ordinateur / Balance of energy in a computer

La totalité de l'énergie électrique consommée est convertie en chaleur. De ce point de vue, un ordinateur se comporte comme un radiateur électrique. Dans les termes de la thermodynamique, cela s'appelle une *dégradation d'énergie* ou *dissipation*.

Mais la fonction de l'objet ordinateur est de *calculer*. Or, le calcul est un objet mathématique

qui n'est pas *a priori* relié à l'énergie. Un ordinateur n'est ni un moteur thermique, ni une pompe à chaleur, et le second principe ne peut pas nous offrir un résultat sous la forme d'un rendement maximal lié à une température. Il se contente de nous dire que la dissipation est positive ou nulle. Mais pourquoi serait-elle non-nulle ?

Y aurait-il une raison fondamentale (indépendante de la technologie de réalisation des ordinateurs) qui contraint à transformer, lors d'un calcul, de l'énergie ordonnée (mécanique, électrique, ...) en énergie désordonnée (chaleur) ? Ne serait-ce que par une insuffisance de savoir-faire que les ingénieurs conçoivent des ordinateurs qui chauffent ?

On sait facilement augmenter la dissipation, plus difficilement la diminuer. On peut rajouter des fuites (rajouter un chauffage inutile), ou des transitions inutiles liées à des calculs inutiles. Cela ressemble beaucoup à l'allongement du temps de calcul des programmes, ou à l'allongement des preuves en mathématiques. Il est toujours facile d'allonger une preuve ou un programme. Mais pour les raccourcir, il faut montrer de la créativité. Cette remarque n'aurait-elle pas un rapport avec notre question ? Mais ce serait déjà vouloir relier la dissipation à une complexité algorithmique.

Avec l'évolution des technologies, on apprend progressivement à identifier des *raisons* ou des *lieux* de dissipation et à les supprimer. À chaque fois qu'on a ainsi supprimé une cause de dissipation, on la qualifie "après coup" de *inutile* ou *parasite*. Mais avant de l'avoir identifiée, on la croit *nécessaire*. Y aurait-il une *limite inférieure* ? Liée à la technologie ? (Si oui, comment trouver la technologie *la meilleure* ?) Liée à la complexité des calculs ? (en *temps* ? en *longueur du programme* ? en *taille des données* ? en *taille du résultat* ?)

Il est généralement reconnu, depuis le papier de Bennett de 1973 que nous allons détailler, que la limite inférieure est liée uniquement à la taille du *résultat* et non à la complexité du calcul en temps.

Matherat et Jaekel en 1996 [2] ont montré que, malgré ce résultat de Bennett, des arguments concernant la structure *logique* du *design* de l'implémentation conduisent à une dissipation proportionnelle au temps de calcul. Ces arguments ne peuvent prétendre être définitifs : on ne peut jamais prétendre qu'on n'inventera pas de nouveaux dispositifs insoupçonnés à ce jour. Mais ces arguments sont-ils pour autant à négliger ? Ils nous interrogent sur des limitations imposées, aux dispositifs physiques, par nos formulations logiques des lois de la nature.

On peut par exemple s'interroger sur les relations entre cette question de la dissipation et les modèles de temps utilisés d'une part dans les modèles de calcul et d'autre part dans les théories physiques. Une telle perspective débouche sur un éventail très large de questions, liées aux systèmes distribués et à la *synchronisation* [3]. Nous indiquerons que nous sommes tentés par cette voie, mais nous n'aborderons pas ces questions ici.

I. 2. Le point de vue des électroniciens dans les technologies contemporaines

Dans les technologies utilisées aujourd’hui pour les puces électroniques (principalement CMOS), une variable booléenne est représentée par une tension électrique V (figure 2).

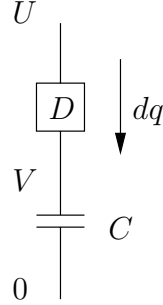


FIG. 2 – Montée de l’équipotentielle V de 0 à U / Rise of the V node from 0 to U

0 et U sont les tensions constantes aux bornes de l’alimentation. Considérons un dispositif D qui fait varier la tension V du point V . Ce point est simplement un conducteur, une équipotentielle, qui représente une variable booléenne. La capacité C n’est pas un composant ajouté mais simplement la capacité (dite *parasite*) du conducteur par rapport à son environnement. D peut être une résistance R , un transistor MOS, ou un assemblage de transistors, ou n’importe quel dispositif qui ne comporte pas de dérivation de charges électriques. Considérons un front montant de V , dont la tension monte de 0 à U (à cause du dispositif D). L’énergie E_d dissipée par le dispositif D lorsqu’il fait passer la charge $Q = CU$ depuis l’équipotentielle U vers la capacité C est :

$$\begin{aligned}
 E_d &= \int_0^{CU} (U - V) dq \\
 &= U \int_0^{CU} dq - C \int_0^U V dV \\
 &= CU^2 - \frac{1}{2}CU^2 \\
 E_d &= \frac{1}{2}CU^2
 \end{aligned}$$

Le premier terme CU^2 est l’énergie fournie par l’alimentation, la moitié $\frac{1}{2}CU^2$ est l’énergie stockée dans le condensateur, leur différence est l’énergie E_d dissipée dans le dispositif D .

Si l’on considérait maintenant la décharge de la capacité C , à travers un autre dispositif D' vers l’équipotentielle 0, c’est-à-dire le front descendant de V , ce serait l’énergie stockée dans le condensateur qui serait dissipée dans D' , et ce serait donc encore $\frac{1}{2}CU^2$. Nous en concluons que chaque changement d’une variable booléenne est associé à une dissipation $\frac{1}{2}CU^2$.

Cette présentation est tout-à-fait générale, n'est pas liée à la nature des dispositifs D et D' , et n'est pas liée à une technologie particulière. Elle n'est pas non plus spécifiquement électronique, car ce raisonnement et ce calcul peuvent être repris tels quels dès lors qu'on souhaite associer un degré de liberté *mécanique* (au sens large) à une variable booléenne. On aura un terme de même forme si on code sur un courant, ou sur une vitesse, ou sur une altitude, ou sur une pression, etc...

Ceci conduit à une dissipation proportionnelle au temps de calcul. La puissance moyenne dissipée est la moyenne de l'énergie dissipée en une seconde de calcul :

$$P_d = f \times N \times \tau \times \frac{CU^2}{2}$$

- f : fréquence d'horloge.
- N : nombre d'équipotentiels (*ie* de variables booléennes),
- τ probabilité de transition pour une équipotentielle, pendant la durée d'une période d'horloge.

On peut considérer que :

- le terme CU^2 est un facteur technologique,
- le terme $f \times N \times \tau$ est lié à une complexité algorithmique.

La puissance dissipée est indépendante du dispositif D , et en particulier de la valeur R d'une résistance. Il s'agit pourtant bien de l'*effet Joule*. L'utilisation de supraconducteurs n'y changerait rien. On peut comprendre cet apparent paradoxe en considérant le courant limite dans les supraconducteurs : si on impose une différence de potentiel non nulle aux bornes d'un supraconducteur, il devient résistif afin d'assurer une dissipation $\frac{1}{2}CU^2$. D'une certaine manière, cette loi de dissipation est prépondérante sur les lois particulières des dispositifs.

D'autres méthodes ont été proposées, comme celle qui consiste à coder une variable booléenne sur deux degrés de liberté (au lieu d'un seul), par exemple en faisant un échange entre une tension et un courant. Mais aucune réalisation n'a démontré le bien-fondé de ces possibilités. Il a même été argumenté sur l'impossibilité d'y parvenir [4]. Comment diminuer cette dissipation ?

1. On peut diminuer U , et diminuer C . Grâce à la *loi de Moore*, qui contrôle l'évolution de la miniaturisation des puces électroniques, on a ainsi gagné un facteur 250 depuis 15 ans sur le terme CU^2 . C'est considérable, mais cela a été fait sans progresser dans la compréhension des mécanismes fondamentaux de la dissipation. Dans le même temps, ce facteur 250 a été compensé par le fait qu'on en a profité pour réaliser des circuits 250 fois plus complexes, qui dissipent donc autant, en faisant 250 fois plus de calculs. Et la dissipation pose toujours les mêmes problèmes.
2. On peut essayer de diminuer le produit $f \times N \times \tau$? En diminuant la complexité algorithmique ?

Le deuxième point conduit à essayer de comprendre mieux ce qu'est *calculer* ! Cela justifie une approche plus formelle : pourquoi avons nous besoin de modifier ces variables ?

Pouvons-nous faire autrement ? Comment est-ce lié à un modèle de calcul ? Peut-être un autre modèle de calcul nous permettrait de ne pas voir le calcul comme une modification de variables booléennes ? C'est le point fort du papier de Bennett, qui ignore les variables booléennes pour raisonner sur les graphes d'automates.

I. 3. Graphes d'automates : dissipation et oubli

Nous allons présenter maintenant une approche qui ne s'appuie pas sur un modèle de calcul lié à la modification de variables booléennes, mais fondé sur des parcours dans des graphes d'automates.

Le résultat de Bennett se situe dans ce cadre et conduit à dire qu'on peut *annuler le terme algorithmique*. Ainsi, le terme technologique sera multiplié par *zero*. La dissipation pourra alors être nulle, quelquesoit la technologie (plus précisément, elle sera liée uniquement à la longueur du résultat, qui peut être très court, voire un bit dans le cas d'une décision). Bennett présente de la façon suivante la genèse de ces idées [5].

Maxwell, dans une annexe de sa *Théorie de la chaleur* (1871) [6], remarque les difficultés liées à l'existence éventuelle d'un "être aux sens assez aiguisés pour trier les molécules en fonction de leurs positions ou leurs vitesses". C'est la naissance du *démon de Maxwell* [7] [8], qui a déclenché de nombreux débats sur les relations entre l'intelligence (ou la vie) et le second principe de la thermodynamique. À l'origine, on n'a pas considéré le démon comme lié au calcul (c'était 65 ans avant le papier fondateur de Turing). Mais aujourd'hui, on a envie de le considérer comme une procédure algorithmique.

Szilard (en 1929) [9] considère qu'il n'y a pas de raison pour ne pas appliquer le second principe au démon lui-même. Ainsi, ce dernier devra dissiper pour compenser les diminutions d'entropie dont il est capable. Cette attitude permet à Szilard de quantifier la dissipation de cet objet "qui fait une mesure et qui l'utilise après" : il doit dissiper au moins $kT \ln 2$ par bit mesuré. Le mot *après* suggère la fonction *mémoire*, donc *automate*. Là encore, ce n'est pas ce qui en a été retenu à l'époque.

Par la suite, on a été conduit à associer une dissipation à une *acquisition* d'information (voir par exemple Gabor (1951) [10], Brillouin (1959) [11]) : il s'agit d'une *divergence* dans un graphe, associée à l'idée de *mesure*. L'information est calculée suivant Shannon (1949) [12], et la dissipation est celle de Szilard.

Landauer (1961) [13] associe la dissipation de Szilard, non à l'*acquisition*, mais à la *perte* d'information (irréversibilité logique) : une *convergence* dans un graphe est une perte d'information, un effacement, un oubli. Cet effacement doit s'accompagner d'une dissipation. On

dira qu'il s'agit là de l'énoncé du *principe de Landauer*.

Bennett (1973) [1] montre qu'on peut trouver un graphe sans convergence pour chaque calcul. C'est le papier que nous allons détailler. Le calcul pourrait ainsi échapper à la dissipation de Szilard, par un *design* soigneux de la machine à calculer.

À la suite de ces considérations, Matherat et Jaekel (1996) [2] ont introduit la notion de *dissipation logique*, qui est définie comme la quantité d'information perdue dans les convergences, quantité qui est indépendante de la technologie de réalisation, et qui devra être ensuite multipliée par le facteur propre à la technologie choisie pour une réalisation particulière. La dissipation logique est donc la quantité dont Bennett suppose implicitement l'existence et dont il démontre la possibilité d'annulation. Ces auteurs montrent quand à eux que le *design logique*, nécessairement modulaire, d'une machine à calculer, impose une dissipation *logique* proportionnelle au temps, ce qui conduit à une dissipation physique également proportionnelle au temps.

Revenons à la démarche de Bennett. Son raisonnement repose sur la considération des convergences dans les graphes, sur lesquelles Landauer avait mis l'accent par son papier de 1961. Considérons une convergence dans un graphe d'automate (figure 3).

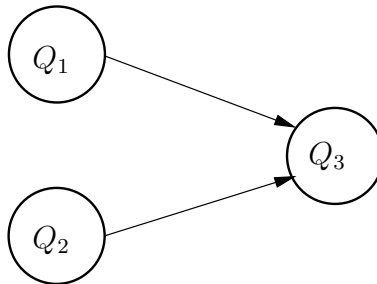


FIG. 3 – Une convergence / A convergence

Quand l'automate se trouve dans l'état Q_3 , il a *oublié* s'il vient de Q_1 ou Q_2 , c'est-à-dire qu'il a perdu l'information qui lui permettrait de *revenir sur ses pas*. Landauer focalise l'attention sur ces *irréversibilités logiques* : un dispositif n'assurant pas de réversibilité logique ne pourra prétendre à la réversibilité physique, donc à la non-dissipation.

Pour essayer d'avoir une compréhension *plus électronique* de ce point, nous pouvons revenir à la discussion précédente sur la charge d'un condensateur (figure 2). En fait, il est tout-à-fait possible de charger un condensateur en dissipant moins que $\frac{1}{2}CU^2$, par exemple avec une tension d'alimentation U qui ne serait pas constante, mais qui serait croissante suivant une rampe lente. Dans ce cas, la différence de potentiel aux bornes du dispositif D pourrait être maintenue faible, et l'énergie dissipée pourrait être beaucoup plus faible. Mais si on applique cette procédure de charge alors que le condensateur est *déjà chargé* (tension U à ses bornes), alors au début de la rampe le dispositif D aura une différence de tension U à ses bornes, et il commencera donc par *décharger* le condensateur en dissipant $\frac{1}{2}CU^2$ avant de le charger à nouveau par la rampe lente montante. Dans ce cas, ce serait le fait de passer de

l'état *chargé* à l'état *chargé* qui serait dissipatif! Ce qu'on ne sait pas faire, c'est appliquer la même procédure de charge dans les deux cas d'état initial du condensateur, et que ces deux cas conduisent tous les deux à une non-dissipation. Il s'agit bien d'une illustration de la convergence de la figure 3 : on veut appliquer la même action pour passer dans l'état Q_3 (chargé) sans connaître l'état initial (chargé ou déchargé).

Appliquons cela au modèle de calcul de la *Machine de Turing*. Considérons que l'on peut voir une machine de Turing comme deux automates qui communiquent (figure 4) : la tête et la bande. Chacun des deux automates peut être représenté par un graphe qui comporte des boucles, des convergences et des divergences.

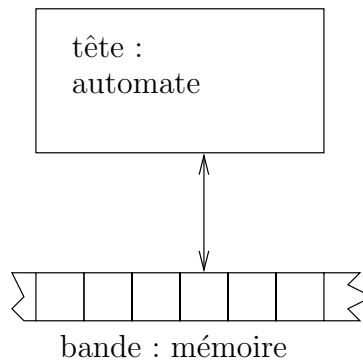


FIG. 4 – Machine de Turing : 2 automates qui communiquent / A Turing machine : 2 communicating automata

En général, les mathématiciens ne considèrent pas que la bande est un automate car c'est son caractère infini et régulier qui importe. Mais dans ce qui suit, nous utilisons ce modèle pour l'appliquer aux ordinateurs concrets, et il est important pour nous de considérer la mémoire comme un automate fini. Certes, nous sortons ainsi du cadre de Turing puisque les résultats sur l'indécidabilité sont liés au caractère infini de la bande. Mais nous nous limiterons à considérer les calculs qui s'arrêtent (donc en fait des automates finis de grande taille en nombre d'états : dans un ordinateur moderne qui possède un disque dur de 100 Goctets, le nombre d'états est 2^{800G} ! Sachant que le nombre de particules de l'univers est de l'ordre de 2^{200} , nous avons à rajouter un *exposant* de 4 milliards).

Nous allons montrer que le produit de la communication entre deux automates peut être un graphe sans convergence, alors même que chacun possède des convergences dans son graphe. Donnons tout de suite un exemple simple (de notre cru) pour imaginer comment cela est possible. Considérons une mémoire 1 bit, une *bascule* T , et son graphe (figure 5).

Il y a deux états. De chaque état partent deux flèches (car l'entrée T peut prendre deux valeurs). De même, sur chaque état arrivent deux flèches. Il y a donc, à chaque coup d'horloge, à la fois acquisition d'un bit, et effacement d'un bit, donc *dissipation logique d'un bit* (cette terminologie est issue de [2]). Considérons deux bascules T synchronisées par un signal commun d'horloge (figure 6).

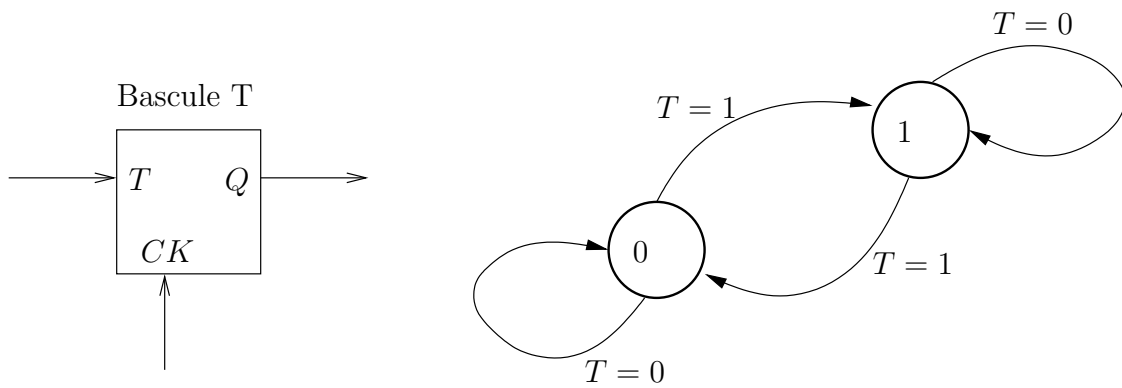


FIG. 5 – Deux flèches convergent sur chaque état : dissipation / Two arrows converging on each state : dissipation

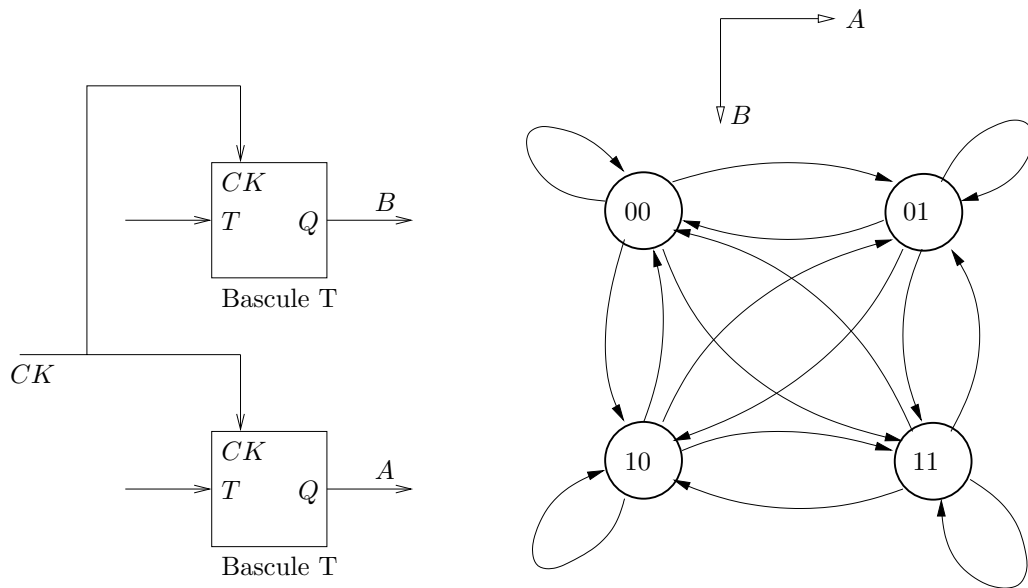


FIG. 6 – Produit cartésien de graphes : quatre flèches convergent sur chaque état / Cartesian product of automata : 4 arrows converging on each state

Il y a quatre états. De chacun partent 4 flèches, sur chacun arrivent 4 flèches, donc dissipation de deux bits à chaque coup d'horloge.

Un calcul particulier est une contrainte permettant de choisir un parcours dans ce graphe. Connectons des fils aux entrées T des bascules (figure 7) afin d'effectuer un tel calcul.

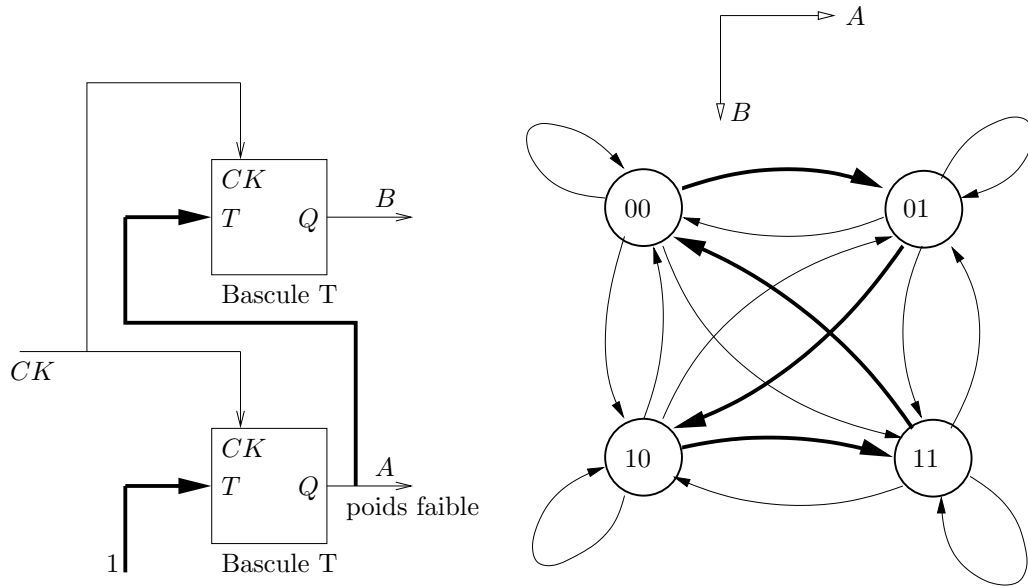


FIG. 7 – Un calcul : parcours contraint dans le graphe / A computation : a constrained path in the graph

Le fait que ces deux automates communiquent entre eux a pour effet de restreindre les possibilités de transition. Les seules transitions autorisées sont les flèches en gras de la figure 7. Le graphe se résume donc à celui de la figure 8 (compteur à 4 états).

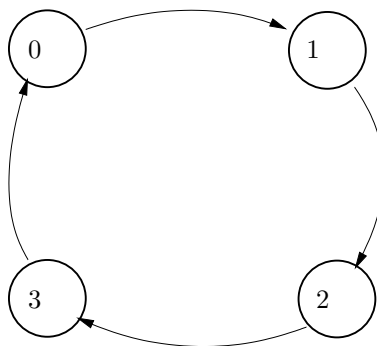


FIG. 8 – Compteur modulo 4 / Modulo 4 counter

Conclusion : *en faisant communiquer des graphes qui comportent des convergences, on peut obtenir un graphe sans convergence.*

L'idée de Bennett est d'appliquer la même procédure avec les deux parties d'une machine de Turing : *tête* et *bande*. Sur une machine de Turing quelconque, *un calcul qui s'arrête* est

un graphe linéaire comme celui de la figure 9 qui ne comporte pas de divergences (car il n’y a pas d’entrée pour la machine globale, la bande se trouvant à *l’intérieur* de la machine).

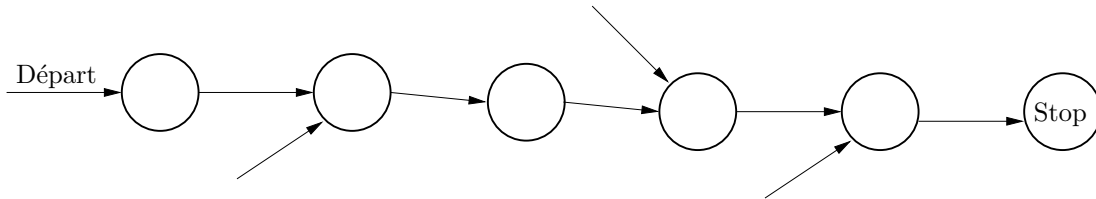


FIG. 9 – Graphe produit pour un calcul qui s’arrête / Product graph for a computation which stops

Il n’y a pas non plus de boucle car on a supposé que le calcul s’arrête. (En effet, s’il y avait une boucle, le calcul ne s’arrêterait pas car on ne sortirait pas de la boucle puisqu’il n’y a pas de divergence.)

Mais ce graphe comporte en général des convergences car plusieurs calculs peuvent conduire au même état *stop* (ie avec la même bande résultat). Il s’agit donc d’un arbre, dont l’état *stop* est la racine, et dont nous n’avons représenté qu’un seul parcours sur la figure 9.

Ainsi, une machine de Turing, en général, efface de l’information en calculant. Elle ne pourrait pas faire le calcul à l’envers. Avancer dans le calcul revient en général à perdre de l’information. Le résultat final contient moins d’information que la configuration initiale. Cela peut paraître paradoxal, mais ça ne l’est pas si on considère que *moins d’information* peut être lié à *information plus pertinente*.

Pour pouvoir faire le calcul à l’envers, on peut imaginer écrire un historique du calcul (les variables intermédiaires) sur une deuxième bande afin de retrouver par quels états le calcul est passé. Si nous souhaitons alors faire un bilan global de l’information effacée, on doit considérer un cycle complet qui revient à l’état initial après avoir fait le calcul. Ce cycle devra donc se terminer par l’effacement de l’historique (réinitialisation), ce qui conduira à une dissipation proportionnelle à la longueur de l’historique, c’est-à-dire proportionnelle à la complexité du calcul en temps.

L’idée ingénieuse de Bennett consiste à imaginer de faire cet effacement de façon *réversible* : dans une première phase du calcul, on écrit l’historique. Puis, au lieu de s’arrêter, la machine continue dans une autre phase dont le rôle est d’effacer l’historique, mais, grâce à une symétrisation complète, cet effacement est fait de façon réversible. Ce que nous allons détailler ci-dessous.

II. Le papier de Bennett

II. 1. Introduction

Nous reprendrons ici l'énoncé de Bennett, auquel nous ajouterons une illustration graphique des transitions, afin d'aider à visualiser les graphes évoqués.

Commençons par présenter la structure générale de la démonstration. Nous allons partir d'une machine de Turing déterministe quelconque (à une bande), que nous appellerons S (figure 10). Nous allons construire une machine R réversible (figure 10) qui effectue le même calcul que S sur le même programme (même bande initiale et même évolution de cette bande, appelée bande de travail). Nous lui adjoindrons une 2ème bande pour enregistrer l'historique, et une 3ème bande pour sauvegarder le résultat. Nous appelons \mathcal{A} l'automate de tête de la machine S et \mathcal{A}' l'automate de tête de la machine R . L'automate \mathcal{A}' sera fonction uniquement de \mathcal{A} , et il sera fini.

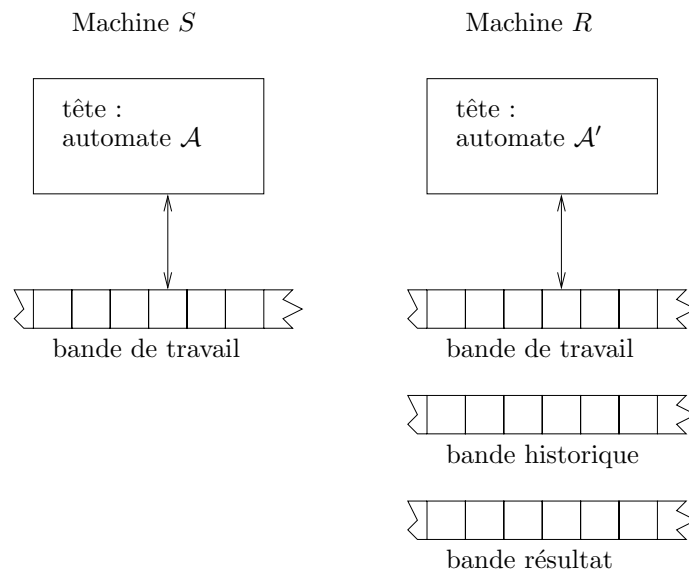


FIG. 10 – Transformation de la machine de Turing S en R / Transformation of the S Turing machine in R machine

La seule différence entre l'état de la machine R à la fin du calcul et son état initial sera la présence du résultat sur la 3ème bande (figure 11). Si nous souhaitons faire un bilan global de l'information effacée, on doit considérer un cycle complet qui revient à l'état initial après avoir fait le calcul. Il nous faut donc ajouter l'effacement de la 3ème bande. Comme ce sera la seule phase dissipative dans le calcul complet, la dissipation logique sera liée à la longueur du résultat (bande output).

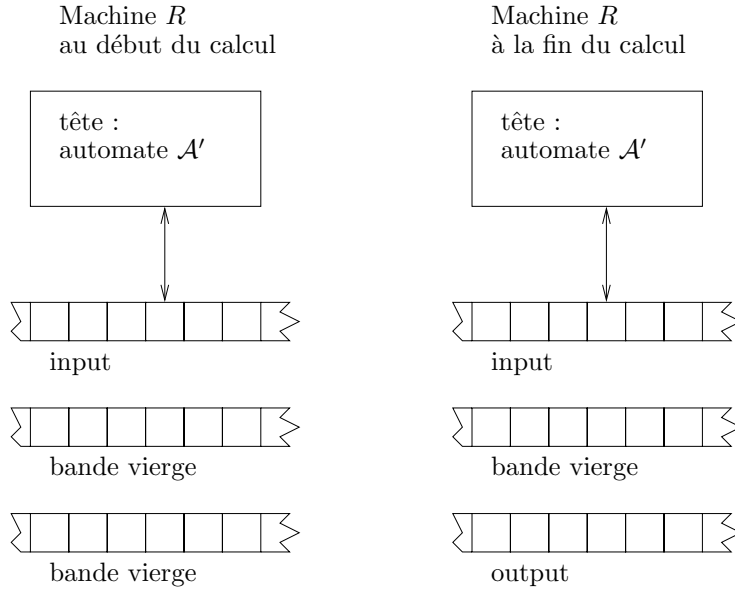


FIG. 11 – Bilan de *dissipation logique* pour la machine R / Balance of *logical dissipation* for the R machine

II. 2. Détaillons les transitions

Une machine de Turing est un ensemble de quintuples qui représentent toutes les transitions possibles. Avant une transition, l'automate de tête est dans l'état A_j et le symbole pointé sur la bande est T . L'effet de la transition sera de remplacer T par T' , de changer d'état (qui devient A_k), puis de décaler éventuellement la bande d'une case, à droite ou à gauche. Nous pouvons noter cette opération de transition de la machine S par un quintuple tel que :

$$A_j T \rightarrow T' \sigma A_k$$

- A_j, A_k états de l'automate de tête \mathcal{A}
- T, T' symboles sur la bande de travail
- σ un déplacement de la tête ($-$, $+$ ou 0)

Nous pouvons aussi la représenter par une flèche dans un schéma (figure 12). À chaque étape, la machine commence par modifier le symbole de la case pointée (T en T') *puis ensuite* effectue le déplacement σ . Ce n'est pas apparent sur la figure 12, mais c'est l'ordre temporel indispensable pour une machine concrète.

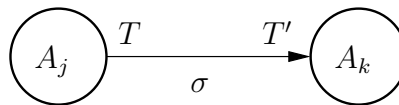


FIG. 12 – Une transition / A transition

L'hypothèse (faite dans l'énoncé initial) que la machine est *déterministe* (figure 13) signifie qu'il n'existe pas 2 quintuples avec le même membre gauche (A_jT).

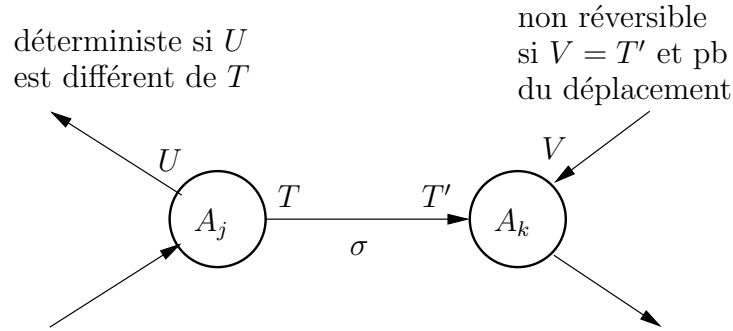


FIG. 13 – Déterminisme et réversibilité de la machine S / Determinism and reversibility of the S machine

Que signifierait que la machine soit *réversible* (figure 13)? S'agirait-il du fait qu'il n'existe pas 2 quintuples avec le même membre droit ($T'\sigma A_k$)? Nous ne pouvons pas raisonner ainsi car les opérations d'écriture sur la bande et de déplacement ne commutent pas, et nous ne pouvons donc simplement changer le sens des flèches pour faire le calcul à l'envers.

II. 3. Quadruples et nouveaux états

Pour rendre envisageable la discussion sur la réversibilité, chaque quintuple :

$$AT \rightarrow T'\sigma A'$$

sera découpé en 2 quadruples :

1. $AT \rightarrow T'A''$ (transcodage seul)
2. $A''/ \rightarrow \sigma A'$ (déplacement seul). Le symbole $/$ indique que pour ce quadruple et cette bande, la lecture de la bande n'a pas d'effet. Le déplacement σ à effectuer est déjà connu par la connaissance de l'état A'' .

Cela nous conduit à introduire un nouvel état A'' pour chaque quintuple transformé. Nous envisagerons alors les quadruples symétriques :

1. $A'/ \rightarrow -\sigma A''$ (déplacement opposé, mais pas de symétrie graphique)
2. $A''T' \rightarrow TA$ (symétrie graphique)

Présentons les notations pour une machine à 2 bandes (2 symboles dans un crochet). Le quadruple suivant (déplacement σ sur la première bande, transcodage $t \rightarrow t'$ sur la deuxième bande, les bandes se déplacent indépendamment les unes des autres) :

$$A[/t] \rightarrow [\sigma t']A'$$

a pour symétrique :

$$A'[/t'] \rightarrow [-\sigma t]A$$

Nous pouvons alors définir déterminisme et réversibilité.

Déterminisme : la machine est déterministe si il n'existe pas 2 quadruples avec à gauche le même état et une lecture qui pourrait avoir la même conséquence. Par exemple, $A[/t]$ et $A[st]$ à gauche forcent le non-déterminisme si les membres de droite sont différents.

Réversibilité : la machine est réversible si l'ensemble des quadruples symétriques définissent une machine déterministe.

Nous pouvons maintenant détailler la transformation d'une machine S quelconque en une machine R réversible effectuant le même calcul. On commence par énumérer les quintuples de l'automate \mathcal{A} de S :

$$\mathbf{1.} \quad A_1 b \rightarrow b + A_2$$

...

$$\mathbf{m.} \quad A_j T \rightarrow T' \sigma A_k$$

...

$$\mathbf{N.} \quad A_{f-1} b \rightarrow b 0 A_f$$

Le symbole b est utilisé comme délimiteur de la zone occupée sur la bande de travail. Les transitions **1** et **N** expriment une normalisation dans la forme des transitions initiale (start) et finale (stop).

Puis on remplace chaque quintuple par un couple de quadruples. Le quintuple d'indice m devient :

$$\begin{aligned} \mathbf{m.} \quad & A_j T \rightarrow T' A'_m \\ & A'_m / \rightarrow \sigma A_k \end{aligned}$$

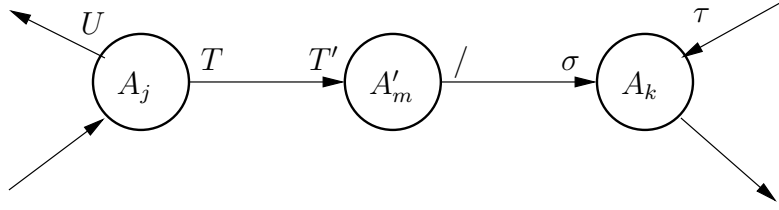


FIG. 14 – Quintuple dédoublé / Split quintuple

Nous avons ainsi ajouté N nouveaux états (les A'_m). L'état A'_m contient entre autre l'information σ . Nous noterons graphiquement les deux transitions ainsi (figure 14).

On ajoute alors la deuxième bande, pour enregistrer l'historique (que nous noterons en dessous des flèches, sur la figure 15).

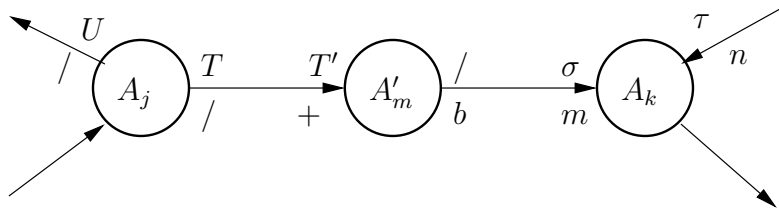


FIG. 15 – Deux bandes / Two tapes

La bande historique enregistre l'indice m de l'état intermédiaire (alphabet de N symboles, N étant le nombre de quintuples de \mathcal{A}). Sur cette bande historique, le mouvement est toujours $+$ car on inscrit les indices m à la suite, en pile.

II. 4. Quadruples pour effacer l'historique

Nous pouvons alors construire les quadruples symétriques, qui nous serviront pour effacer l'historique lorsque nous ferons le calcul à l'envers dans la dernière phase de l'exécution. Nous noterons par la lettre C les nouveaux états symétriques des états A , avec l'indice qui correspond par la symétrie (figure 16). En partant de C_k , la lecture de m sur la bande historique permet de retrouver $-\sigma$ et C'_m . Puis C'_m permet de connaître la transition vers C_j . Sur ces quadruples, déterminisme et réversibilité se correspondent par symétrie.

La totalité du graphe de l'automate de tête de la machine R peut se représenter comme sur la figure 17, en 3 parties. La phase 1 est celle qui mime le calcul sur S . La phase 3 est l'image miroir de la phase 1, elle sert à faire le calcul à l'envers pour effacer l'historique. Entre ces deux phases, la phase 2 sert à recopier le résultat, de la bande de travail vers la bande *Output*.

Les mouvements des têtes sur les 3 bandes et l'état des bandes peuvent se représenter comme

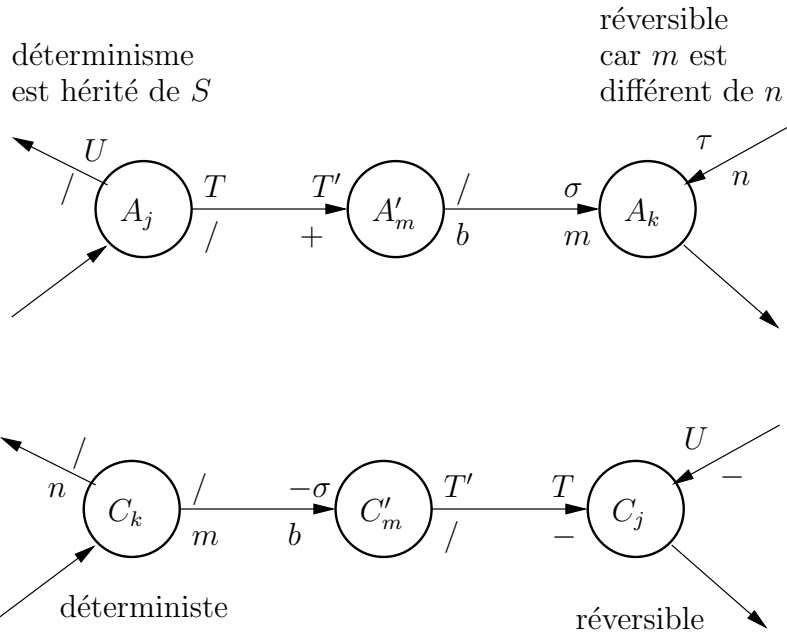


FIG. 16 – Transitions symétriques de la machine R / Symetric transitions for the R machine

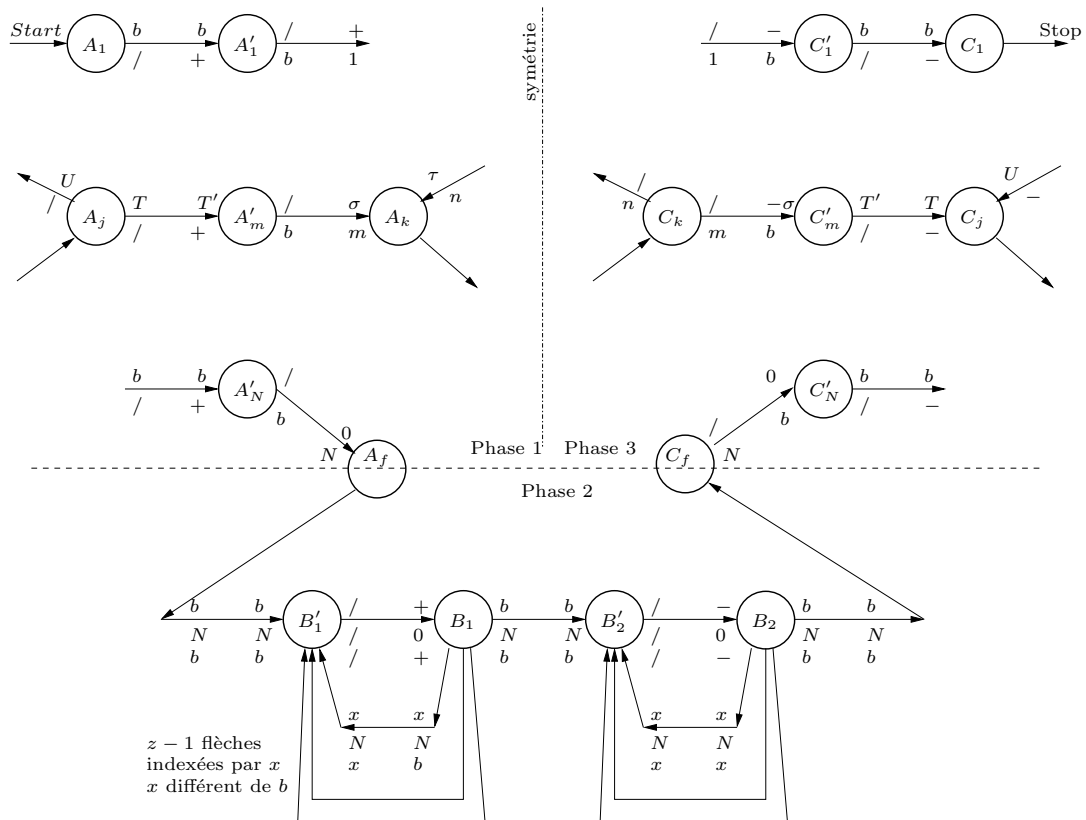


FIG. 17 – Graphe de l'automate \mathcal{A}' , tête de la machine R / Graph of the \mathcal{A}' automaton, head of the R machine

sur la figure 18 (le souligné indique la position de la tête).

<i>Phase</i>	<i>Bande de travail</i>	<i>Bande historique</i>	<i>Bande résultat</i>
1 : calcul	<u>_INPUT</u>	-	-
	<u>_OUTPUT</u>	<u>HISTORY</u>	-
2 : recopie du résultat	<u>_OUTPUT</u>	<u>HISTORY</u>	<u>_OUTPUT</u>
	<u>_INPUT</u>	-	<u>_OUTPUT</u>

FIG. 18 – Positions des têtes sur les bandes (d’après Bennett) / Positions of the heads on the tapes (from Bennett)

II. 5. Bilan

Si la machine S ne s’arrête pas, R ne s’arrête pas non plus, et ne sort pas de la phase 1. Nous ne nous intéresserons pas à ce cas. Considérons au contraire un calcul de S qui s’arrête :

- après ν étapes
- en utilisant s cases
- et en produisant un résultat de longueur λ .

Nous aurons, pour la machine R :

Phase 1 : Calcul classique (mime la machine S)

- f états (copiés sur ceux de S) auxquels s’ajoute N nouveaux états A'_m
- $2N$ transitions (les quadruples)
- 2ν étapes (2 quadruples à la place de chaque quintuple),
- $s, \nu + 1, 0$ cases utilisées (respectivement) sur les 3 bandes.

À la fin de cette phase, la tête de la bande de travail est positionnée à l’extrémité gauche de la zone écrite (c’est-à-dire à gauche de la zone différente de b).

Phase 2 : On effectue alors la recopie du résultat sur la 3ème bande (recopie de la bande 1 sur la bande 3) grâce aux états B_i et aux transitions spéciales.

Au début, les têtes des bandes 1 et 3 sont à gauche (figure 18). On scanne vers la droite en recopiant, puis quand on rencontre le caractère b qui indique l’extrémité droite du résultat,

on ramène les deux têtes à gauche.

- 4 nouveaux états (B_1, B'_1, B_2, B'_2)
- $2z + 3$ transitions (si z est le nombre de symboles différents y compris le blanc)
- $4\lambda + 5$ étapes
- $0, 0, \lambda + 2$ cases supplémentaires utilisées

Phase 3 : On poursuit le calcul par une phase symétrique de la phase 1, grâce aux états C_j et aux quadruples symétriques de ceux de la 1ère phase.

La tête de la bande 2 ne se déplace alors que vers la gauche, en effaçant chaque case de façon réversible.

- $f + N$ états
- $2N$ transitions
- 2ν étapes
- $0, 0, 0$ cases supplémentaires utilisées

Nous obtenons le *bilan total* qui suit.

- $2f + 2N + 4$ états
- $4N + 2z + 3$ transitions
- $4\nu + 4\lambda + 5$ étapes
- $s, \nu + 1, \lambda + 2$ cases utilisées

La machine R est **déterministe**. Elle est aussi **réversible**, et son graphe est linéaire, sans aucun embranchement : ni divergence, ni convergence (figure 19).

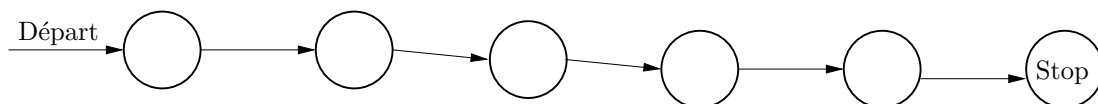


FIG. 19 – Graphe global du calcul, sans embranchement / Global graph of the computation, with no branch

S peut être une machine de Turing universelle, auquel cas R est une machine de Turing universelle réversible.

Si R s'arrête, elle ne dissipe que pour l'effacement du résultat lors de sa réinitialisation.

III. Discussion-conclusion

III. 1. Additivité de la dissipation des parties ?

Ce n'est pas le graphe de la tête \mathcal{A}' qui est sans convergence, c'est le graphe du calcul particulier, le graphe global. De même, le graphe de la bande seule n'est pas sans convergence.

On peut comprendre cela en pensant que l'information de l'historique (les indices m) est échangée entre la tête et la bande, elle se trouve suivant les moments soit dans la tête (sous forme d'états A'_m ou C'_m) soit dans la bande *historique* (symboles m écrits). Elle est donc *effacée de façon irréversible dans chaque partie*, alors que globalement elle ne se trouve effacée que de façon réversible.

La question essentielle est donc la suivante : « Le graphe d'une partie n'est pas non-dissipatif mais le graphe complet pour chaque calcul est non-dissipatif. Dans une réalisation, quel graphe doit-on retenir pour la dissipation ? ». La dissipation d'une partie isolée de machine est-elle modifiée lorsque cette partie communique et engendre la non-dissipation du calcul global ? En d'autres termes, la dissipation d'un assemblage de composants est-elle la somme des dissipations des composants ?

Dans le calcul quantique (voir par exemple [14]), ce sont les interférences entre les parties qui règlent cette question. La machine dissiperait si, à chaque étape, l'échange d'information entre les deux parties consiste en ce que la partie qui lit (*ie* qui échantillonne) la sortie de l'autre partie effectue une *mesure*. Si en revanche on pouvait éviter cette mesure, alors on pourrait éviter la dissipation.

Mais alors, peut-on *tester* les parties indépendamment les unes des autres ? Une partie aurait une fonction qui dépendrait de ses interférences avec son environnement. Peut-on alors continuer à dire que c'est *la* fonction de cette partie ? Une façon de s'en sortir serait d'inclure, dans la description de la fonction testée, la liste des environnements possibles. Mais pour une machine à calculer, la liste des environnements d'une partie est la liste de tous les calculs. Cela signifierait que le test d'une partie consiste à faire tous les calculs possibles avant de certifier que cette partie fonctionne correctement, ce qui est tout-à-fait inenvisageable vu le nombre exponentiellement élevé de calculs possibles.

Cette discussion du test modulaire est issue de [2]. Ce papier suggère que des arguments concernant la structure logique du design de l'implémentation empêcheraient la réalisation non-dissipative d'une machine de Bennett. Mais ces arguments ne peuvent être définitifs. Ce n'est pas *une preuve qu'il ne peut pas exister* d'implémentations non-dissipatives (même si on voit mal comment un concepteur pourrait échapper à ce besoin de modularité). En effet, pour les objets fabriqués par l'homme, exhiber une réalisation d'un objet qui fonctionne est une preuve que c'est possible ; mais ne pas savoir le faire ne peut constituer une preuve que c'est impossible. D'une part, on ne peut supposer que la créativité des ingénieurs va s'arrêter.

D'autre part, une telle démonstration fondée sur une théorie n'a de valeur que dans le cadre où cette théorie est valable, et ce cadre peut à tout moment être bousculé.

III. 2. Pistes pour poursuivre

Dans ce contexte, comment avancer ? Examinons rapidement quelques pistes qui ont été suivies depuis trente ans, avant d'envisager d'autres pistes possibles.

Une classe de tentatives consiste à ne pas prendre en compte cette discussion sur les graphes et à s'intéresser à la cause électronique identifiée des dissipations : les transitions de variables booléennes, implémentées par des variations de tension. On peut chercher à supprimer certaines catégories de transitions. On élimine les transitions qui sont des allers-retours intermédiaires dans les circuits combinatoires (*glitches*), on élimine les transitions dans les parties de circuits inutilisées (en supprimant localement l'horloge : *clock gating*). Dans cette catégorie, certaines classes de circuits asynchrones minimisent automatiquement le nombre de transitions, à la fois en supprimant les transitions inutiles, et en n'activant pas les parties de circuits inutilisées (voir par exemple [15]). Mais toutes ces méthodes, si elles ont un intérêt certain pour abaisser la dissipation, n'en conservent pas moins une dissipation proportionnelle à la complexité des calculs, donc au temps de calcul.

Une tentative très intéressante concerne le *calcul conservatif* de Fredkin et Toffoli [16]. Ces auteurs considèrent le calcul comme un assemblage de composants élémentaires conservatifs et réversibles (portes de Fredkin, et *Billiard Ball Computer*). C'est un modèle à base de variables booléennes mais qui, par certains côtés (en particulier par la symétrie introduite dans le calcul), s'inspire de la méthode utilisée dans le papier de Bennett. Mais, alors que les arguments sont très convaincants pour montrer la possibilité de réaliser des fonctions combinatoires conservatives, les choses sont plus délicates pour ce qui concerne les fonctions séquentielles. Dans cette théorie, les mémoires sont inutiles car la synchronisation est effectuée automatiquement par un espace-temps discret. Malheureusement, l'espace-temps de la nature n'est pas discret, et la synchronisation est plus problématique. Pour illustrer ce point, supposons que nous utilisions les montages proposés par [16] et que nous remplaçons tous les *flits unité* (qui réalisent une propagation de une unité d'espace en une unité de temps) par des points-mémoire. Nous obtenons alors un montage pipe-liné et il est hors de doute que l'assemblage fonctionnerait correctement en réalisant la fonction logique souhaitée. Mais ce serait une synchronisation beaucoup trop forte, car entraînant une dissipation proportionnelle au temps de calcul, comme on peut s'en convaincre en étudiant les graphes engendrés. Il faudrait trouver une synchronisation intermédiaire, si possible minimale, permettant d'assurer que le calcul s'effectue correctement, mais ce papier n'indique pas de piste dans ce sens. Notre suggestion n'est pas une réfutation, mais elle nous met sur la piste d'une recherche sur les liens entre la dissipation et la synchronisation. Il semble que la synchronisation nécessaire à un calcul soit liée aux propriétés géométriques de ce calcul particulier, et c'est peut-être ce que nous indique le résultat de Bennett. Le papier [16] possède en outre un autre intérêt, il permet de quantifier la dissipation qui est en jeu, par l'intermédiaire de l'évaluation de

la taille des informations à jeter ou à recycler (*garbage*). Il serait intéressant d'étudier les relations entre la taille de ce *garbage* et les besoins en synchronisations.

Une autre proposition très intéressante concerne le *calcul adiabatique*. On peut consulter par exemple [17] et [18]. On peut classer dans la même catégorie le calcul *hot-clock* [19]. Le principe consiste à charger ou décharger les équipotentielles (qui représentent les variables booléennes) à l'aide de dispositifs dont les bornes ne voient que peu de différence de potentiel (rampes que nous avons signalé en I. 3.). Il faut disposer d'une série de signaux d'horloge en forme de rampe, séquencées d'une manière particulière. Il faut également symétriser les calculs (d'une façon qui ressemble à la symétrie mise en oeuvre par [1] et [16]). Ces propositions sont, elles aussi, tout-à-fait convaincantes pour ce qui concerne les opérateurs combinatoires, mais moins pour ce qui concerne les fonctions séquentielles (comme les registres et leurs points-mémoire).

À propos des points-mémoire, il faut prendre conscience d'une subtilité de l'argument de Bennett. Celui-ci ne considère jamais qu'une mémoire pourrait ne pas dissiper malgré ses convergences. Au contraire, il s'appuie sur ces convergences et sur le *principe de Landauer* indiquant que ce sont ces convergences d'une mémoire qui sont à l'origine de la dissipation, pour déboucher sur une méthode permettant d'éliminer les convergences dans le graphe global. Si on veut suivre sa méthode, il est indispensable d'argumenter sur le graphe global de chaque calcul, et non sur l'association de composants. Le résultat de Bennett revient en effet à dire que la dissipation d'un assemblage de composants n'est pas la somme des dissipations des composants. Or, de nombreux auteurs (dont tous ceux que nous venons de citer) proposent des façons non-dissipatives d'implémenter les portes combinatoires primitives, et croient pouvoir généraliser en disant que, puisque tout montage d'électronique numérique est un assemblage de ces primitives, alors la dissipation de l'assemblage, qui serait la somme des dissipations nulles des primitives, sera elle-même nulle. Mais il faudrait pour cela argumenter sur l'additivité de la dissipation, puisque c'est justement ce que conteste Bennett. Dans le papier [2], nous avons utilisé une telle propriété d'additivité pour contester la possibilité de réalisation d'une machine de Bennett. Elle était fondée sur la dissipation logique liée au test.

Nous pouvons également rappeler à cette occasion qu'il ne suffit pas de dire : *un point mémoire est obtenu en rebouclant une fonction combinatoire*, et se servir de cet argument pour transférer la non-dissipation du combinatoire à celle du point-mémoire. C'est déjà une opération de composition. Ce rebouclage impose, en vue d'obtenir une stabilité, une amplification (qui va être dissipatrice) alors même que c'est en supprimant cette amplification qu'on pourrait obtenir des circuits combinatoires non-dissipatifs. Ce ne sont pas les mêmes implémentations électroniques des primitives qui sont utilisées, d'une part dans les blocs combinatoires, d'autres part dans les points-mémoire. Quand un point-mémoire dissipe moins, il est moins stable. L'argument de Bennett tendrait à dire que si on arrive à supprimer les convergences, alors on n'a plus besoin de ces stabilités.

Parmi toutes les tentatives citées ci-dessus, aucune ne suit vraiment les arguments de Bennett qui ont l'avantage de ne pas se placer au niveau d'une technologie de réalisation pratique particulière, mais se situent au niveau du modèle de calcul, en comptabilisant les changements

d'états d'un automate qui sont associés à des convergences.

La piste que nous aurions plutôt envie de suivre consiste à s'interroger sur l'instant et le lieu où se place cette convergence (et donc cette dissipation). La démonstration de Bennett supprime ces convergences en supprimant la séparation entre des parties de la machine : en supprimant le lieu et l'instant d'une communication entre des parties, on supprimerait les convergences. On débouche alors sur une question de synchronisation dans un système distribué : à partir de quel niveau de diminution des synchronisations, celles-ci permettent-elles encore le calcul? Le graphe de la figure 19, qui ne comporte pas de convergence, ne nécessite aucune synchronisation. Il ne communique avec l'extérieur que par les signaux *start* et *stop*. C'est une machine close. Ce n'est pas une machine distribuée, elle n'est pas *composée de composants*.

Dans cet esprit d'étude des synchronisations, nous nous sommes intéressé aux circuits asynchrones. Rappelons que dans ces circuits, il ne s'agit pas de supprimer les synchronisations, mais de supprimer l'horloge globale qui est à l'origine de la majeure partie de la dissipation en implémentant une synchronisation trop forte. Ces circuits modélisent un espace-temps d'une nature différente et proposent donc un autre modèle formel du calcul. La suppression de l'horloge globale a pour effet de supprimer l'existence d'un ordre total entre les événements du calcul (transitions de variables booléennes). Il n'est plus possible de parler de simultanéité, mais seulement d'ordre causal pour certaines paires d'événements [3]. Le temps n'est plus un nombre, mais un ordre partiel. Nous quittons le modèle newtonien d'un temps global pour aborder un espace-temps qui ressemble fort à celui de la relativité et qui semble faire appel à la même causalité.

Dans ce contexte, la notion d'état global n'a plus de sens, et la modélisation par un *automate* ne tient plus. La notion même d'*information* (et donc d'entropie) doit changer de formulation car elle repose sur les états d'un automate. La discussion de Bennett, fondée sur des convergences sur des états d'automate, n'est plus adaptée [20]. Comment reformuler ces questions? Il est clair qu'il faut changer de modèle de calcul, tout en tenant compte des propriétés de l'espace-temps physique. Nous sommes là dans un domaine à l'articulation entre calcul formel et fondements logiques de la physique. Devons nous attendre de nouvelles connaissances sur l'espace-temps et la thermodynamique relativiste que pourraient nous apporter les physiciens? Nous préférons penser que, au contraire, c'est la formulation de ces modèles de calcul asynchrone qui nous apportera une nouvelle façon de regarder la nature et qui pourrait déboucher sur de nouvelles connaissances en physique.

III. 3. Conclusion

Ce papier de Bennett montre la possibilité de calculer sans dissiper. Il ne peut constituer une preuve qu'il serait possible de réaliser une machine à calculer qui ne dissiperait pas. En effet une preuve d'existence d'un objet technique ne peut se faire qu'en exhibant une réalisation qui possède la propriété voulue. Ceci n'a pas encore été réalisé : il n'existe toujours pas

d'ordinateur qui ne chauffe pas (et qui ne consomme pas d'électricité). Notre conviction (qui, elle non plus, ne saurait s'appuyer sur une preuve) est que la réalisation d'une telle machine restera impossible, en raison des arguments de test modulaire évoqués dans [2].

Mais ce papier de Bennett nous semble d'une importance majeure si on considère la richesse des questions qu'il soulève, concernant les liens entre le temps du calcul et le temps de la physique. Au-delà de leur relation dans la réalisation d'une machine à calculer, cela débouche sur des questions plus fondamentales concernant les fondements logiques de la physique, et interroge sur un accès à ces questions par le biais de la création de nouveaux modèles de calcul distribué. Ces aspects nous réservent probablement de belles surprises.

Références

- [1] BENNETT (C. H.), Logical reversibility of computation, *IBM J. Res. Develop.*, pp. 525-532, nov. 1973.
- [2] MATHERAT (P.), JAEKEL (M.-T.), Dissipation logique des implémentations d'automates - dissipation du calcul, *Technique et Science Informatiques*, **15-8**, pp. 1079-1104, 1996. Téléchargeable (en français et en anglais) à : <http://www.comelec.enst.fr/~matherat/publications/tsi96/>
- [3] MATHERAT (P.), JAEKEL (M.-T.), Concurrent computing machines and physical space-time, *Mathematical Structures in Computer Science*, **13-5**, pp. 771-798, oct. 2003, CUP. Téléchargeable à : <http://www.comelec.enst.fr/~matherat/publications/mscs03/>
- [4] MEAD (C. A.), CONWAY (L. A.), *Introduction to VLSI systems*, Addison-Wesley, 1980.
- [5] BENNETT (C. H.), The thermodynamics of computation - a review, *Int. J. of Theor. Phys.*, **21-12**, pp. 905-940, dec. 1982.
- [6] MAXWELL (J. C.), *Theory of heat*, Text-books of Science, London : Longmans, Green Co, pp. 328-329, 1875, quatrième édition, (première éd. en 1871).
- [7] LEFF (H. S.), REX (A. F.), Resource letter MD-1 : Maxwell's demon, *Am. J. Phys.*, **58**, pp. 201-209, 1990.
- [8] BENNETT (C. H.), Demons, engines and the second law, *Scientific American*, pp. 88-96, 1987.
- [9] SZILARD (L.), Über die Entropieverminderung in einem thermodynamischen System bei Eingriffen intelligenter Wesen, *Zeit. Phys.*, **53**, pp. 840-856, 1929. Traduction en anglais : On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings, *Behavioral Science*, **9**, pp. 301-310, 1964.
- [10] GABOR (D.), *Light and information*, Richie lecture, University of Edimburg, 1951. *Progress in Optics* **1**, pp. 109-153, 1961.
- [11] BRILLOUIN (L.), *La science et la théorie de l'information*, Masson et Cie, 1959, réédition : Éditions Jacques Gabay, 1988.
- [12] SHANNON (C. E.), WEAVER (W.), *The mathematical theory of communication*, Illinois University Press, 1949.

- [13] LANDAUER (R.), Irreversibility and heat generation in the computing process, *IBM J. Res. Develop.*, pp. 183-191, 1961.
- [14] DEUTSCH (D.), Quantum theory, the Church-Turing Principle and the universal quantum computer, *Proc. R. Soc.*, **A400**, pp. 97-117, 1985.
- [15] EBERGEN (J. C.), SEGERS (J.), BENKO (I.), Parallel Program And Asynchronous Circuit Design, in *Asynchronous Digital Circuit Design*, Workshop in Computing, G. Birtwistle and A. Davis editors, pages 50-103, BCS Springer, 1995.
- [16] FREDKIN (E.), TOFFOLI (T.), Conservative logic, *International Journal of Theoretical Physics*, **21**-(3/4), pp. 219-253, 1982.
- [17] KOLLER (J. G.), ATHAS (W. C.), Adiabatic Switching, Low Energy Computing, and the Physics of Storing and Erasing Information, *Proceedings of the Workshop on Physics and computation*, october 2-4, 1992, Dallas, Texas, IEEE Computer Society Press, pp. 267-270.
- [18] MERKLE (R. C.), Reversible electronic logic using switches, *Nanotechnology*, **4**, pp. 21-40, 1993.
- [19] SEITZ (C. L.) *et al.*, Hot-Clock nMOS, *Technical Report, California Institute of Technology, Pasadena*, 1985. CaltechCSTR :1985.5177-tr-85, <http://caltechctr.library.caltech.edu/365/>
- [20] MATHERAT (P.), Le temps du calcul et le temps des ordinateurs, <http://hal.archives-ouvertes.fr/ccsd-00089817>