Dissipation logique des implémentations d'automates - dissipation du calcul

Philippe Matherat, Marc-Thierry Jaekel

Technique et science informatiques. Volume 15 – n° 8/1996, pages 1079 à 1104

RECHERCHE

Dissipation logique des implémentations d'automates — dissipation du calcul

Philippe Matherat* **— Marc-Thierry Jaekel****

* *Laboratoire d'informatique de l'École normale supérieure, CNRS*
45 rue d'Ulm, 75005 Paris

** *Laboratoire de physique théorique de l'École normale supérieure, CNRS-UPS*
24 rue Lhomond, 75005 Paris

RÉSUMÉ. *La discussion de principe de la dissipation d'énergie par les ordinateurs révèle les contraintes qu'impose la logique sur les systèmes physiques devant remplir une fonction logique. Nous sommes amenés à définir une notion de dissipation logique pour un automate fini. Nous discutons les contraintes associées à l'implémentation physique et exhibons le rôle joué par la modularité dans la testabilité. Il en résulte que les machines à calculer concrètes, qui sont nécessairement modulaires, dissipent proportionnellement au temps de calcul.*

ABSTRACT. *As revealed by discussion of principle of energy dissipation by computers, logics imposes constraints on physical systems designed for a logical function. We define a notion of logical dissipation for a finite automaton. We discuss the constraints associated with physical implementations and exhibit the role played by modularity for testability. As a result, practical computers, which are necessarily modular, dissipate proportionnally to computation time.*

MOTS-CLÉS : *implémentation physique d'automate, démon de Maxwell, graphe d'automate et modularité, machine de Turing, dissipation du calcul.*

KEY WORDS : *physical implementation of automata, Maxwell demon, graph of automata and modularity, Turing machine, dissipation of computation.*

1. Introduction

Les automates finis sont des objets mathématiques tandis que la notion de dissipation est issue du deuxième principe de la Thermodynamique et concerne les systèmes physiques. D'un côté, les automates sont définis dans un contexte où les états *successifs* ne le sont qu'au sens des nombres entiers et non successifs dans le temps. De l'autre côté, l'évolution d'un système physique quelconque ne se réduit pas aux transitions entre les états d'un automate fini. Mais l'implémentation des automates sous forme de systèmes physiques rencontre des contraintes que la logique impose à la physique et réciproquement.

Technique et science informatiques. Volume 15 – n° 8/1996, pages 1079 à 1104

Un système physique, qu'il existe spontanément ou qu'il ait été installé par un expérimentateur, est un ensemble d'éléments qui obéissent à des lois dynamiques. Son évolution peut être décrite par les conditions initiales et les lois du mouvement. Mais le fait que toute évolution puisse ou non prendre la forme d'un ensemble fini d'*états* liés par des *transitions* n'est pas laissé au choix du physicien. Pourtant, dans un grand nombre de situations pratiques, le système physique étudié n'est pas imposé à l'expérimentateur, mais il est au contraire conçu, fabriqué, testé par l'expérimentateur lui-même. Quand on conçoit une machine, on choisit l'affectation des degrés de liberté macroscopiques. Si l'on choisit un piston se déplaçant dans un cylindre entre deux positions extrêmes, on met en place la matière nécessaire pour assurer le guidage et on rejette *a priori* les montages qui n'assureraient pas une rigidité suffisante ou souffriraient d'une usure trop précoce. De même, un concepteur de circuits électroniques logiques cherchant à implémenter un automate imposera la fonction logique macroscopique et rejettera tous les montages n'assurant pas la fiabilité nécessaire.

La question de l'*existence* d'une implémentation physique réalisant une fonction logique souhaitée, de façon stable et fiable, n'a pas une réponse évidente. Elle débouche sur la question de la preuve d'un montage physique. Il est indispensable de se convaincre que l'objet réalise bien la fonction voulue pour toutes les configurations d'utilisation et continuera à la réaliser dans le futur. Ceci est en général garanti par un *test* qui vérifie que l'objet physique se comporte comme sa définition logique.

Admettons que de telles implémentations d'automates existent et qu'elles ne se réduisent pas à des systèmes physiques quelconques. Nous chercherons à mettre en évidence leurs propriétés particulières.

Rentrent dans la catégorie de ces machines à fonctions choisies par un expérimentateur, beaucoup d'instruments des physiciens, beaucoup d'appareils de mesure. Les lois fondamentales de la nature sont observées à l'aide d'appareils auxquels on a assigné des fonctions qui doivent pouvoir être décrites dans un langage logique. Ainsi, même si les systèmes physiques ne se réduisent pas à des automates, l'expérimentation sur ces systèmes physiques nécessite de faire intervenir des automates.

Nous ne nous interrogerons pas ici de façon générale à propos de la relation entre la structure du langage logique et la forme des lois physiques, mais nous étudierons les contraintes imposées (par leur fonction logique) sur les systèmes physiques implémentant un automate fini déterministe ou une machine de Turing déterministe, en rapport en particulier avec la question de la dissipation.

1.1. *Historique*

Nous rappelons ici très brièvement les étapes essentielles qui marquèrent la relation entre le calcul et la dissipation. Une bibliographie détaillée peut être trouvée dans [LEF 90].

Le premier exemple d'automate intervenant dans un processus physique fut introduit par J.-C. Maxwell en liaison avec le caractère statistique du deuxième

principe de la Thermodynamique [MAX 75].

Reprenant l'analyse du démon de Maxwell, Szilard précisa qu'un tel dispositif physique, qui mémorise de l'information, doit aussi dissiper, sinon il permettrait de réaliser un mouvement perpétuel de seconde espèce [SZI 29]. En s'appuyant sur le deuxième principe de la Thermodynamique, il calcula qu'une mémoire un bit, donc un automate à deux états, tel que ses deux états de mémorisation correspondent à des augmentations d'entropie S_1 et S_2, doit vérifier :

$$e^{-S_1/k} + e^{-S_2/k} \leq 1$$

où k est la constante de Boltzman ($= 1,38.10^{-23}$ Joules / Kelvin). Ce résultat est établi par un raisonnement sur un cycle complet d'une machine thermique monotherme comportant un gaz parfait et une mémoire.

Certains auteurs virent dans la *mesure* (initialisation de la mémoire) l'origine de cette dissipation (Gabor [GAB 51], Brilloin [BRI 59]), alors que Landauer préféra situer cette dernière dans l'*effacement* (oubli de l'information précédemment mémorisée) [LAN 61]. Ces deux interprétations semblent s'opposer mais nous montrerons section 5 qu'il n'en est rien pour des automates finis.

Réinterprétant Szilard, Brillouin suggère un principe de Carnot généralisé s'écrivant $\Delta(S - I) \geq 0$ où S représente l'entropie et I l'information, toutes les deux dans la même unité. A l'opposé, Landauer suggère qu'un système mémorisant un bit d'information à la température T doit dissiper au minimum une énergie de l'ordre de $kT \ln 2$ à chaque effacement du bit, quelle que soit sa réalisation physique (mécanique, électronique, etc.) [LAN 61]. Ce point de vue insiste sur le caractère général de la dissipation *minimale*, liée à la fonction logique, même si l'énergie réellement dissipée peut dépendre de choix adoptés dans la réalisation physique de la mémoire. Ce « principe de Landauer » (comme le nommera plus tard Bennett) focalise l'attention sur les *irréversibilités logiques* : un dispositif n'assurant pas de réversibilité logique ne pourrait prétendre à la réversibilité physique.

Ces deux principes (de Brillouin et de Landauer) déplacent donc, chacun à leur manière, la discussion de la dissipation, depuis le domaine de la physique vers le domaine de la logique. Ils suggèrent l'utilisation d'une notion de *dissipation logique*, mesurée par la quantité d'information perdue, qu'on pourrait ensuite transposer dans le domaine de la physique sous la forme d'une énergie minimale dissipée.

Le principe de Landauer, que nous appellerons plutôt *critère*, n'a jamais été confronté à l'expérience en raison de la faiblesse de l'énergie dissipée invoquée. Elle est, à la température ambiante, inférieure de nombreux ordres de grandeurs aux énergies dissipées par les meilleurs dispositifs fabriqués actuellement, ce qui laisse penser que la « dissipation de Landauer » est masquée par des effets dissipatifs plus importants dont l'origine est mal comprise. Des discussions de principe montrent aussi que son expression ne peut être qu'approximative et doit en particulier être remise en cause à très basse température (voir par exemple [LIK 82]).

À la suite de Landauer, Bennett montra qu'à toute machine de Turing S, on peut

associer une machine de Turing R, qui effectue les mêmes calculs que la première, mais qui mémorise l'historique du calcul, puis qui efface cette historique, le tout de façon logiquement réversible. Il en déduisit que tout calcul pourrait être effectué avec une dissipation qui n'est pas liée au nombre d'étapes du calcul, mais seulement à la taille du résultat [BEN 73].

Ici, nous ne discuterons ni la validité ni l'évaluation physique du critère de Landauer, mais nous étudierons les propriétés de la dissipation logique. Bien que définie logiquement, cette dissipation caractérise des systèmes physiques. Ainsi, nous montrerons que les contraintes de réalisation physique de systèmes informatiques induites par la nécessité de testabilité conduisent à des structures logiques particulières qui ont pour conséquence leur dissipation logique. Nous verrons que la construction de Bennett, même si elle permet la réversibilité logique de tout calcul, ne fournit pas une implémentation non-dissipative que l'on peut expliciter avant le début du calcul (qui ne soit pas fonction du calcul particulier). Pour les machines réversibles universelles, les contraintes d'implémentation physique conduisent à des réalisations dissipatives.

1.2. *Plan*

Les notions d'*automate fini* et de *dissipation logique* méritent tout d'abord d'être précisées. L'existence de cette deuxième notion est supposée implicitement par les considérations de Brillouin et Landauer, mais aucune définition n'en a jusqu'à présent été donnée.

Nous discuterons l'implémentation physique des automates. Contrairement à l'automate défini formellement, sa réalisation physique doit satisfaire des contraintes supplémentaires afin d'assurer sa fonction logique. Elle doit permettre en particulier de tester que l'objet physique satisfait bien sa fonction. Nous choisirons de ne considérer que des objets physiques dont la fonction est testable et n'évolue pas dans le temps.

Nous introduirons ensuite la notion de *modularité* pour une implémentation d'automate. Cette structure permet de simplifier considérablement le test des implémentations et donc de réaliser des automates de complexité très grande. Nous montrerons qu'en contrepartie, ces implémentations conduisent à de la dissipation.

Enfin, nous discuterons les possibilités d'implémentation physique de machines de Turing déterministes et leur dissipation. Nous montrerons que les implémentations de machines à calculer dissipent proportionnellement au temps de calcul.

2. Automates finis

Les automates finis sont utilisés dans des contextes très variés [PER 95]. Lorsqu'on les utilise pour définir un langage, leurs états ne sont pas comparables aux états d'un système physique. En revanche, lorsqu'on les utilise pour décrire un montage électronique séquentiel, un état de l'automate mathématique correspond à

un état physique stable du montage électronique et les transitions de l'automate mathématique correspondent à des transitions physiques ordonnées dans le temps physique. Dans cette situation, toute transition vers un état est une validation de la chaîne d'entrée passée qui a conduit à cet état. C'est cette deuxième attitude que nous souhaitons adopter.

Notre définition d'automate sera très proche des définitions classiques, mais elle en différera par certains points, pour permettre une définition des termes *convergence* et *divergence*. Nous chercherons à utiliser le plus possible le vocabulaire habituel (voir par exemple [MIN 67] ou [STE 90]).

Nous verrons un automate comme une *boîte noire* (figure 1). Les entrées sont des stimuli (S) issus du monde extérieur. Les sorties sont des réponses (R) destinées au monde extérieur. Les valeurs différentes possibles de S sont en nombre fini (symboles choisis dans un alphabet fini). De même pour R (les alphabets sont en général distincts, c'est-à-dire avec un nombre différent de symboles pour S et pour R).

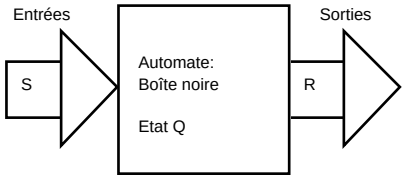

Figure 1. *Automate fini*

Pour désigner les valeurs successives de l'état interne Q de l'automate, ainsi que les valeurs successives des entrées et des sorties, nous utiliserons un indice entier t (temps). Il ne s'agit pas du temps physique, la succession n'est que celle des nombres entiers. Lorsque nous définirons les implémentations physiques d'automates, cet indice t deviendra un temps physique discret (rythmé par une horloge extérieure à l'automate).

L'évolution de l'automate s'écrit à l'aide des deux fonctions F et G suivantes :

$$\begin{cases} Q_{t+1} = G(Q_t, S_t) \\ R_t = F(Q_t) \end{cases}$$

où Q_t, S_t et R_t désignent l'état de l'automate, les stimuli sur les entrées et les réponses en sortie, à l'instant t. G est appelée *fonction de transition*.

Nous utiliserons la notation graphique de G qui associe un cercle à chaque état et une flèche à chaque changement d'état (figure 2), qu'on appelle *diagramme d'états* ou *graphe de l'automate*.

Figure 2. *Une transition*

Dire que G est une fonction revient à dire que nous ne nous intéresserons qu'aux automates *déterministes*.

Dans notre définition de la fonction F, la réponse R_t ne dépend que de l'état Q_t. En conséquence, un changement de l'entrée ne peut changer la sortie qu'à l'instant suivant, *via* un changement d'état. Cette définition stipule que toute l'information utilisable en sortie réside dans l'état de l'automate. C'est ce que nous appellerons une fonction *séquentielle.*

Pour simplifier, nous imposerons à F d'être injective. Le monde extérieur peut connaître l'état instantané de l'automate en connaissant la valeur instantanée de la réponse. Ceci simplifiera le test d'une implémentation physique d'automate.

La suite des symboles d'entrée sera appelée *mot d'entrée*, de même en sortie *mot de sortie*. Leur longueur peut être infinie si le diagramme comporte un circuit (retour à un état déjà visité).

2.1. *Divergence et convergence*

Tout d'abord, nous imposerons que les flèches qui partent d'un même état arrivent sur des états différents. Si deux symboles A et B distincts déclenchent une transition de l'état Q vers l'état Q', alors nous dirons que cette transition est déclenchée par l'étiquette « A ou B » et nous ne ferons figurer qu'une seule flèche sur le diagramme d'état (figure 3). Ainsi, les diagrammes d'états ne sont pas des multigraphes, mais des graphes orientés et étiquetés.

Définition : Si deux transitions au moins partent d'un même état (elles correspondent à des valeurs différentes de l'entrée et elles se dirigent vers des états différents), nous dirons alors que nous sommes en présence d'une **divergence**.

Remarque : En cas d'absence de divergence, il y a deux possibilités :

— entrée *indifférente* : l'environnement fournit une entrée, mais l'automate effectue la même transition pour toutes les valeurs de cette entrée ; nous étiquetterons la transition par l'union de toutes les valeurs possibles de l'entrée,

— entrée *implicite* : l'automate attend une valeur particulière de l'entrée et l'environnement ne peut que fournir cette valeur.

Pour les discussions qui suivent, ces deux cas sont équivalents.

Définition : Si deux transitions au moins arrivent sur un même état, nous dirons que nous sommes en présence d'une **convergence** .

Définition : Un automate est dit **réversible** si et seulement si son graphe ne comporte pas de convergence.

Remarque : L'équivalence entre réversibilité et absence de convergence n'est compatible qu'avec une définition de la fonction F qui implique qu'un changement

de la réponse ne peut qu'être consécutif à un changement de l'état. Toute l'information sur le *passé* de l'automate réside dans l'état interne. Si l'état est atteint par une convergence, il y a perte d'information sur le passé de l'automate.

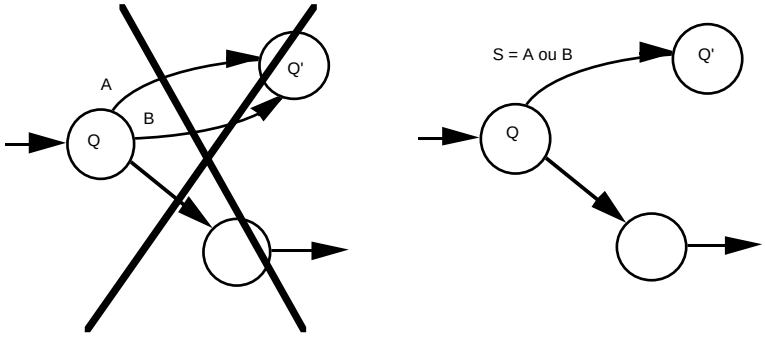

Figure 3. *Une seule flèche de Q vers Q'*

Pour nous situer par rapport aux définitions de [STE 90], nous considérerons que :

— tous les états sont *acceptants* : c'est ce qui permet aux états logiques de devenir des états physiques stables. Toutes les transitions passées sont *réussies*.

— il n'y a pas de *rebut* : la fonction de transition est une fonction partielle. Ceci signifie que pour certains états, il se peut que certains symboles ne déclenchent pas de transition. Dans ce cas, ce symbole est interdit en entrée à cet instant. Les définitions d'automates comportant un rebut conduisent à un nombre de flèches quittant chaque état égal au nombre de symboles du vocabulaire d'entrée. Ainsi tout état est source de divergence, ce que nous ne souhaitons pas.

L'automate n'est compatible qu'avec une classe d'environnements, ceux qui ne présentent comme stimuli que des mots compatibles avec un *parcours* du graphe (*calcul*). C'est la définition de l'automate qui entraîne la définition de la classe d'environnements compatibles. Pour les implémentations physiques d'automate, ceci conduira à exclure les environnements physiques non compatibles avec la définition de l'automate.

3. Implémentation physique d'automate

Nous souhaitons mettre en correspondance les états de l'automate logique avec des états stables d'un système physique et mettre en correspondance les transitions de l'automate logique avec les transitions physiques.

Nous allons exclure les automates qui seraient définis par une formule récursive et dont le caractère fini serait ainsi indécidable (nous discuterons de l'éventualité de cette situation section 8). En outre, nous ne considérerons ci-dessous que des automates tels que pour chacun la définition conduit à un graphe stable dans le temps.

La « boîte noire » de la figure 1 devient un objet physique. Les entrées et les sorties deviennent des communications avec l'environnement. Le temps devient physique. Celui-ci n'est pas discret, mais nous allons supposer ici que nous pouvons le discrétiser, à l'aide d'une horloge externe qui envoie à l'automate des *tops* sur une entrée supplémentaire (entrée CK figure 4). Il faut supposer que les stimuli sont également synchronisés avec cette horloge, ce qui est une contrainte imposée par l'automate sur son environnement. Ces tops d'horloge déclenchent des transformations internes à l'automate et nous supposons qu'après un certain temps plus court qu'une période d'horloge, l'état physique est *stabilisé* d'une manière telle qu'il puisse représenter un des états logiques de l'automate. Ceci n'implique pas un état physique sans évolution, mais implique une correspondance non ambiguë avec un seul état logique.

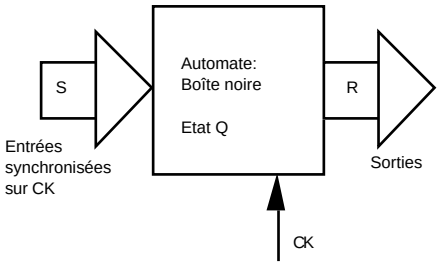

Figure 4. *Implémentation physique d'automate*

3.1. *Existence*

Un tel système physique n'est pas un objet que l'on cherche à connaître après en avoir constaté l'existence, mais c'est au contraire un objet que l'on a décrit par sa fonction avant de savoir si son existence est possible, compatible avec les propriétés fondamentales des systèmes physiques, ces dernières pouvant imposer des limitations.

Nous ferons l'hypothèse de l'existence de telles implémentations physiques d'automates (en tenant compte éventuellement de certaines contraintes d'implémentation) et nous étudierons les conséquences de nos définitions. Si seule une classe réduite d'objets physiques répond aux définitions et aux contraintes d'implémentation, nous saurons que ces conséquences ne s'appliquent qu'à cette classe réduite.

Nous nous laisserons guider par l'existence de montages d'électronique logique (principalement les ordinateurs) qui se comportent comme ces objets supposés, avec les contraintes d'implémentation et d'utilisation habituelles : alimentation électrique, température, valeurs possibles des entrées, conditions d'entrance-sortance, synchronisation des entrées sur l'horloge, etc. Nous ne citons ces contraintes que pour indiquer ce que peut être l'existence, sans considérer qu'il s'agit d'une preuve d'existence, et nous ne supposerons aucune de ces contraintes comme nécessaires.

3.2. *Test d'une implémentation physique*

Afin de donner à une fonction logique une existence physique, le concepteur cherche une implémentation qui correspond au cahier des charges exprimé par le graphe, puis s'engage à ce que ce fonctionnement logique soit vérifié quelle que soit l'utilisation qui est faite de l'implémentation trouvée, dans les limites de conditions d'utilisation définies dans un contrat. La conformité de chaque implémentation avec le graphe est contrôlée par un *test.*

Par test, nous entendons que l'objet physique est stimulé par une machine qui présente sur ses entrées la suite de symboles nécessaire pour explorer la totalité du graphe, c'est-à-dire passer par tous les états et emprunter toutes les transitions. Nous ne considérerons que les automates pour lesquels cela est possible.

Cette notion de test est liée à la stabilité de la fonction dans le temps. Elle interdit que le graphe puisse évoluer en cours de fonctionnement. L'état de l'automate change, mais non son graphe. On suppose que l'on peut déduire un comportement futur et dans d'autres environnements à partir du bon fonctionnement à un moment donné sur une machine de test. Cela conduit à l'introduction d'une notion de *date de test*.

En fait, le test est indispensable pour définir l'implémentation physique de l'automate. En effet, il est hors de question de s'assurer du fonctionnement de l'automate par une description dynamique de tous ses constituants. Le test nous permet d'associer un *graphe caractéristique* à l'objet physique, sans nous soucier davantage du fonctionnement physique interne de l'objet.

3.3. *Coût d'une implémentation*

On définira le *coût* d'une implémentation par la complexité en temps de calcul du programme de test. Le test doit vérifier la fonction de transition, donc tester toutes les transitions. Le nombre de transitions est inférieur ou égal au produit du

nombre d'états par le nombre de symboles d'entrée. Pour un vocabulaire donné, le nombre de transitions croît donc comme le nombre d'états. Pour tester une transition particulière, il se peut qu'il soit nécessaire de parcourir une partie du graphe. Au pire, le coût croît donc comme le carré du nombre d'états.

4. Démons de Maxwell

Les démons de Maxwell sont des exemples de systèmes physiques qui se comportent comme des automates. Les mémoires un bit comme celles de Szilard et Landauer sont des implémentations physiques d'automates à deux états comme celui de la figure 5.

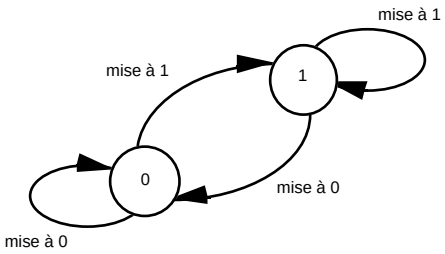

Figure 5. *Mémoire un bit*

Chaque état voit deux flèches converger vers lui, donc l'automate n'est pas réversible. Toute transition est à la fois un enregistrement d'un bit (*initialisation* ou *mesure*) et un *effacement* du bit précédemment mémorisé.

Cette mémoire permet de réaliser un démon de Maxwell, comme le montre la figure 6, dérivée de l'interprétation de Bennett du démon de Szilard [BEN 87]. Cette figure représente un cylindre fermé par deux pistons qui emprisonnent une molécule, le tout à une température uniforme.

Il faut rappeler qu'un système physique simple (sans mémoire), comme une trappe actionnée par le passage de la molécule [SMO 12], ne peut agir comme un démon de Maxwell [FEY 63]. Pour agir en ce sens, il est nécessaire d'enregistrer la position de la molécule afin d'utiliser cette information à une date ultérieure (dans la figure 6, la troisième et la quatrième étapes dépendent de la deuxième). Notons que mesure, mémorisation et effacement sont indissociables et étroitement liés au traitement de l'information spécifié par le graphe de la mémoire. La fonction logique de l'automate caractérise et résume le rôle supplémentaire joué par un observateur, tel le démon de Maxwell, par rapport à un système physique simple. L'argument de Szilard montre que, selon le deuxième principe de la Thermodynamique, l'automate doit dissiper de l'énergie pendant son fonctionnement.

Les possibilités de mémorisation et de perte d'information des automates peuvent ne pas se résumer à des graphes aussi simples que celui de la figure 5. Les

convergences ne sont pas toujours associées à des divergences qui leur sont symétriques. Néanmoins, nous montrerons dans la section suivante comment les convergences et divergences permettent de caractériser la dissipation de l'automate.

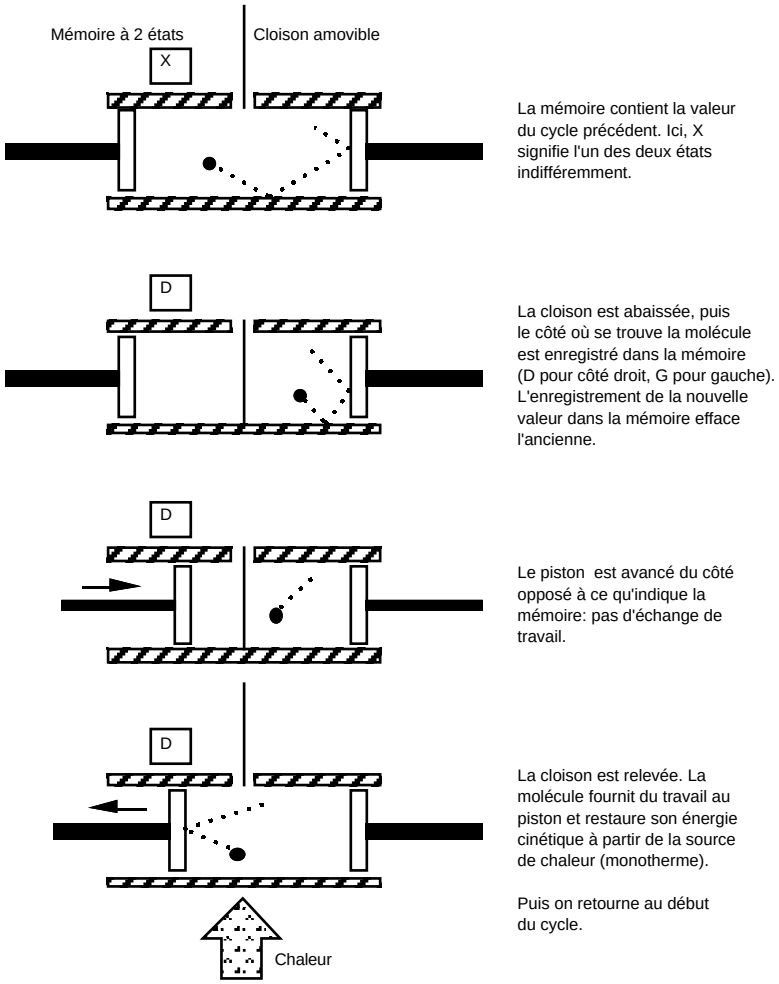

Figure 6. *Démon de Szilard*

Technique et science informatiques. Volume 15 – n° 8/1996, pages 1079 à 1104

5. Dissipation logique

Une notion de dissipation logique, liée uniquement au graphe de l'implémentation et non à des particularités physiques, peut être définie.

Considérons un exemple d'automate un peu plus complexe que la mémoire un bit et représenté par la figure 7. Considérons les deux mots suivants, de dix symboles binaires, reconnus par cet automate :

mot 1 : 0100001010
mot 2 : 0011100110

Ces deux mots correspondent à deux parcours du graphe (calculs) qui sont différenciés par les options prises lors de la sortie des états B, C et G.

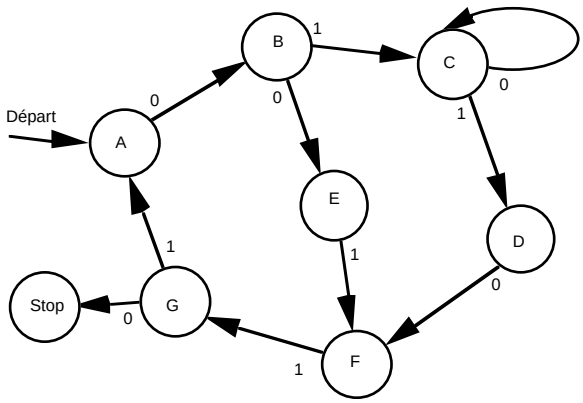

Figure 7. *Exemple d'automate dissipatif*

L'automate, pour sortir d'un de ces états, utilise l'information fournie par le symbole d'entrée. Si on veut quantifier cette information par les probabilités des symboles d'entrée, alors soient p_{0i} et p_{1i} les probabilités de sortir de l'état i en lisant un 0 ou un 1 (avec $p_{0i} + p_{1i} = 1$). L'information nécessaire (mesurée en bits) pour sortir de l'état i est ([SHA 49]) :

$$-(p_{0i} \log_2 p_{0i} + p_{1i} \log_2 p_{1i}).$$

Cette expression est égale à un bit si $p_{0i} = p_{1i}$. Elle est égale à zéro bit si l'entrée est indifférente ou implicite, c'est-à-dire si une seule flèche part de l'état considéré (dans notre exemple, nous avons considéré que les transitions sans divergence sont associées à des entrées implicites).

Les mots 1 et 2 correspondent aux parcours (calculs) suivants :

mot 1 : Départ A **B C C C C C** D F **G** Stop (7 bits de choix)

mot 2 : Départ A **B** E F **G** A **B** E F **G** Stop (4 bits de choix)

Nous avons écrit en gras les états (B, C et G) qui sont quittés grâce à une entrée qui porte un bit d'information (si les deux valeurs de l'entrée sont équiprobables) et nous avons indiqué entre parenthèses la quantité d'information portée par le mot d'entrée correspondant au calcul.

Les divergences à la sortie des états B, C et G sont empruntées grâce à l'information apportée par le mot d'entrée. Ainsi, lorsque l'automate se trouve dans un des états C, D, ou E, il *garde la trace* de la façon dont il est sorti de l'état B. Il *mémorise* l'information du mot d'entrée qui lui a permis de sortir de B. En revanche, lorsqu'il a atteint ou dépassé l'état F, il a perdu cette information. Dans l'esprit du critère de Landauer, l'automate devrait dissiper à cause de cette convergence sur l'état F.

Définition : Nous appellerons **dissipation logique** (ou *information dissipée*) la quantité d'information perdue par l'automate lors des *convergences*. Le montant de cette dissipation logique se calcule sur les *divergences* du graphe.

La dissipation logique dépend de la particularité du mot d'entrée par rapport à la classe des mots reconnus par l'automate. Sa mesure n'est pas forcément de un bit par symbole binaire. Il faut considérer la probabilité du mot particulier parmi l'ensemble des mots possibles. Si cet ensemble contient N mots équiprobables, l'information est ($\log_2 N$) bits. Si cet ensemble de mots n'est pas fini, on peut considérer la probabilité d'un mot parmi les mots de même longueur. Cette définition de la dissipation logique comme la quantité d'information d'une suite de symboles (mot), s'accorde avec les résultats de Shannon [SHA 49] et de Brillouin [BRI 59].

La dissipation logique est relative à l'automate. Si l'automate est spécialisé pour un mot de sortie s particulier (pas de divergence), la quantité d'information du mot d'entrée e sera nulle et toute l'information nécessaire pour engendrer s sera contenue dans la structure du graphe de l'automate. Un tel automate ne dissipe pas. À l'opposé, si l'automate est tel que celui de la figure 5, alors la quantité d'information du mot e sera égale à la longueur du mot de sortie s (un bit de choix par symbole binaire émis). L'automate de la figure 5 a donc une structure qui ne fait que mémoriser et décaler le mot d'un cran vers les instants supérieurs. Cet automate dissipe un bit par symbole binaire lu en entrée.

On peut s'imaginer que la difficulté dans le lien entre quantité d'information et dissipation (divergences ou convergences) est à l'origine de la différence d'interprétation quant à l'origine de la dissipation dans l'expérience de Szilard (mesure ou effacement). Le fait de considérer ici des automates finis permet de réconcilier ces deux interprétations. L'opposition entre mesure et effacement n'a lieu que si l'on ne considère que des caractéristiques locales des graphes et non le graphe dans son entier. En effet, le graphe de la figure 5 pourrait sembler équivalent à un arbre binaire infini si le caractère fini n'était pas imposé.

Dans tout ce qui suit, nous dirons simplement *dissipation* pour dissipation logique et nous la mesurerons en bits.

Nous considérerons essentiellement des automates comportant des circuits. Dans ce cas, si une portion du calcul (parcours) est telle que les états de départ et

d'arrivée sont identiques, la dissipation logique est égale exactement à l'information contenue dans la portion de mot d'entrée correspondante. Toute l'information lue en entrée est dissipée.

6. Implémentation modulaire d'automate

Nous avons vu que le coût d'une implémentation physique d'automate, défini par la complexité du test, croît comme le nombre de transitions de l'automate, qui croît lui-même comme le nombre d'états (ou au pire son carré). Or le nombre d'états des automates utilisés dans la pratique peut être gigantesque. Par exemple le nombre d'états d'une mémoire de n bits est égal à 2^n, alors que n dépasse couramment 10^6 dans les ordinateurs actuels. Le temps de test d'une telle fonction dépasse largement l'âge de l'Univers ($\approx 3.10^{17}$ secondes), même si la période élémentaire du programme de test est aussi petit que le temps de Planck ($\approx 5.10^{-44}$ s). On ne peut imaginer l'implémentation de telles fonctions que si leur validation est obtenue à partir des tests des parties qui les composent. Mais ceci impose l'existence d'une fonction stable pour ces parties. Nous qualifierons de *modulaire* une implémentation qui permet le test séparé des parties. Avant d'en donner une définition plus précise, il faut illustrer ceci par un exemple, montrant en particulier que la modularité est l'objet d'un choix.

6.1. *Exemple de choix d'implémentation : compteur modulo quatre*

Supposons que l'on souhaite réaliser le compteur cyclique modulo quatre défini par le graphe de la figure 8.

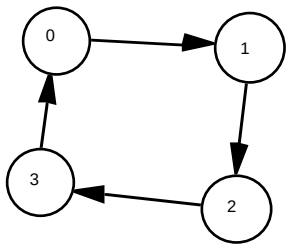

Figure 8. *Compteur modulo quatre*

Ce type de compteur est utilisé pour réaliser des diviseurs de fréquence. Il ne possède pas d'entrée d'information, mais il change d'état à chaque top d'horloge. Son graphe ne comporte pas de divergence ni de convergence. Sa dissipation logique est nulle et on peut imaginer le réaliser avec une dissipation physique aussi faible que l'on veut. Une idée de réalisation physique serait d'utiliser une roue qui tourne, sur laquelle seraient gravés quatre secteurs représentant les quatre états. On

conçoit aisément que ce genre d'implémentation ne rencontre qu'une dissipation liée à des frottements qui pourraient être diminués par une amélioration technique, dissipation non liée à la fonction logique.

En revanche, la méthode couramment utilisée par les électroniciens consiste à coder l'état sur deux chiffres binaires mémorisés dans des *mémoires un bit*. Décrivons un exemple d'une telle implémentation.

Définissons tout d'abord l'automate *Bascule T* comme sur la figure 9 qui en donne le symbole et le graphe. À chaque top d'horloge CK, l'état précédent et la valeur de l'entrée T déterminent le nouvel état de la mémoire, qui est connu à l'extérieur grâce à la sortie Q.

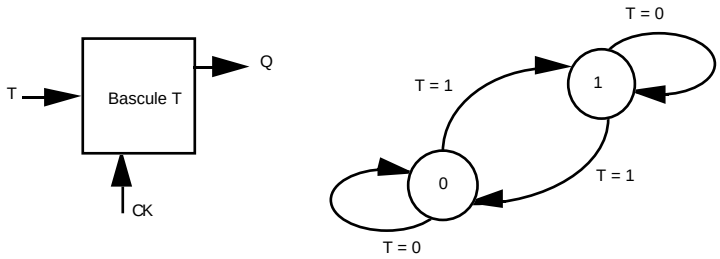

Figure 9. *Bascule T*

Si cette bascule est fabriquée et testée séparément, elle peut être utilisée comme module (en deux exemplaires) pour implémenter le compteur modulo quatre, comme sur la figure 10, où l'état est codé en binaire sur les deux sorties des bascules A et B (A est le poids faible).

Le graphe de cette implémentation, représenté à droite de la figure 10, s'obtient en ouvrant les connexions aux entrées T des bascules. Il est le produit cartésien de deux graphes de bascules T. Chaque changement d'état est associé à une divergence et une convergence de quatre flèches, ce qui correspond au fait que les deux bascules possèdent chacune une entrée qui peut porter un bit d'information. Le parcours particulier du compteur modulo quatre dans ce graphe se réduit au circuit dessiné en gras. Les flèches en maigre ne sont jamais empruntées lorsque les connexions entre les bascules sont installées. Mais il faut les comptabiliser pour le calcul de la dissipation puisque ces bascules sont testées indépendamment de leurs interconnexions et que le graphe caractéristique de chacune n'est pas modifié par l'association modulaire.

On se trouve donc ici face à deux graphes d'implémentations possibles pour une même fonction définie par le graphe de la figure 8. La première implémentation (roue qui tourne) a un graphe identique à celui de la figure 8 et est non dissipative. La seconde (deux bascules T) a le graphe de la figure 10. Elle est modulaire et dissipative.

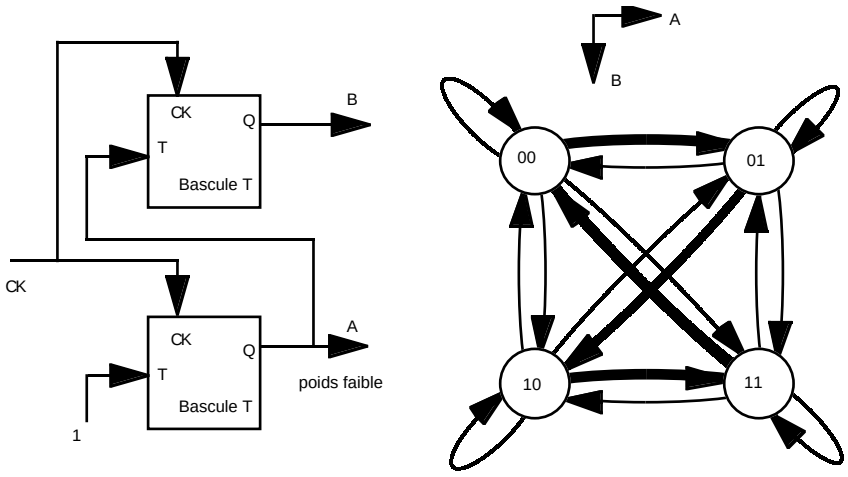

Figure 10. *Compteur deux bits*

6.2. *Modularité et propriété d'extensivité de la dissipation logique*

Définition : Nous dirons qu'une implémentation physique d'automate est **modulaire** si elle consiste en une association de sous-parties (*modules*) qui sont des implémentations physiques d'automates testables séparément, assemblées entre elles par des communications qui relient la sortie d'un module aux entrées d'autres modules.

L'assemblage n'est pas testé globalement, il est en général trop complexe. Par contre, chacun des modules est testé séparément. Il revient au concepteur de prouver que l'assemblage modulaire réalise bien la fonction qu'il souhaite, à partir des fonctions de chaque module. Nous ne nous soucierons pas de cette preuve ici. Elle pourrait être obtenue par exemple par une simulation sur une machine de Turing.

Dans un assemblage modulaire, le graphe caractéristique de chaque module est indépendant des autres modules de l'automate : il est défini par un test indépendant. L'association modulaire ne modifie pas les graphes caractéristiques des modules.

Le graphe de l'implémentation modulaire est exhibé par le test. Il s'obtient en ouvrant les connexions d'entrées des modules, ce qui traduit le fait que ces derniers sont stimulés indépendamment par la machine de test. Le graphe de l'implémentation est donc le produit cartésien des graphes des modules. L'état global est spécifié par l'énumération des états des modules. Le nombre d'états du graphe de l'implémentation est le produit des nombres d'états des modules. La fonction de transition de l'implémentation est le produit cartésien des fonctions de transitions des modules.

Écrivons la fonction de transition pour deux modules A et B (la généralisation à plus de deux modules est immédiate). Nous noterons l'état, le stimulus et la fonction de transition de chacun avec son nom en exposant. La fonction de

transition G^{AB} de l'implémentation globale est définie par :

$$Q_{t+1}^{AB} = (Q_{t+1}^{A}, Q_{t+1}^{B}) = G^{AB}(Q_t^A, Q_t^B, S_t^A, S_t^B).$$

Le fait que la fonction de transition de chaque module soit testable séparément (quel que soit l'évolution de l'autre module) implique que celle de l'implémentation puisse s'écrire :

$$G^{AB}(Q_t^A, Q_t^B, S_t^A, S_t^B) = (G^A(Q_t^A, S_t^A), G^B(Q_t^B, S_t^B)).$$

Le nombre de flèches qui quittent l'état (Q^A, Q^B) est égal au produit des nombres de flèches qui quittent les états Q^A et Q^B dans les graphes des modules séparés. De même avec les flèches qui atteignent un état. Ainsi, la dissipation de l'implémentation est la somme des dissipations des modules. La dissipation logique apparaît donc comme une grandeur **extensive**.

Remarque : À moins d'être une simple juxtaposition de modules sans communication, une implémentation modulaire comprend au moins un module avec une entrée reliée à la sortie d'un autre module et qui possède donc au moins une divergence. Dès que cette divergence est associée à une convergence, celle-ci force la dissipation de l'implémentation modulaire.

Pour illustrer ceci, reprenons l'exemple des deux bascules implémentant le compteur modulo quatre. Dans la figure 10, la bascule de poids faible voit son entrée T reliée à une constante. On pourrait simplifier en remplaçant cette bascule par un compteur modulo deux dont le symbole et le graphe sont représentés figure 11.

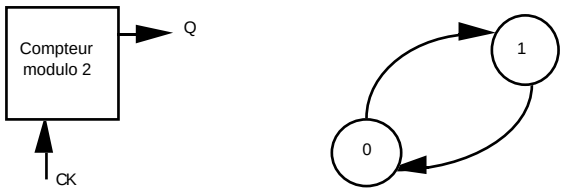

Figure 11. *Compteur modulo deux*

L'implémentation du compteur modulo quatre devient alors celle de la figure 12. Dans cet assemblage modulaire, le compteur modulo deux est non-dissipatif et la bascule T est dissipative. L'assemblage est donc dissipatif, ce qui traduit le fait qu'un des modules possède une entrée, ce qui est nécessaire pour qu'une information soit communiquée d'un module à l'autre afin de réaliser une fonction moins spécialisée que la simple juxtaposition de modules sans interaction.

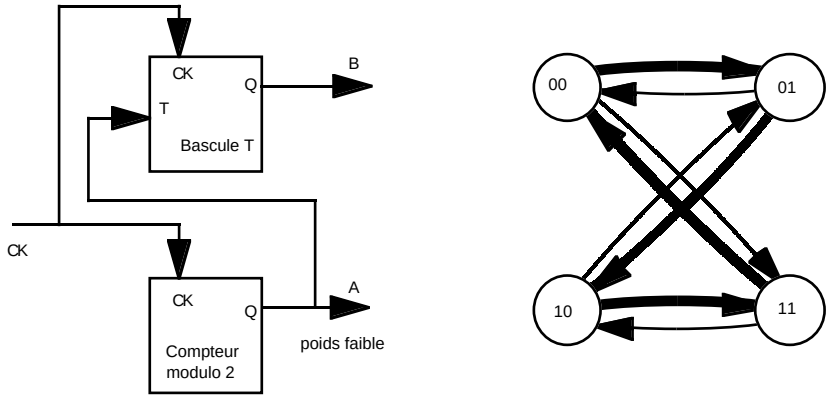

Figure 12. *Un des modules est non-dissipatif*

6.3. *Coût d'une implémentation modulaire*

Avec la définition du coût du paragraphe 3.3, nous voyons que le coût d'une implémentation modulaire est plus faible que celui d'une implémentation non modulaire (et de façon exponentielle). En effet, la complexité du test de l'implémentation modulaire est la somme des complexités des tests des modules et ne croît pas comme le nombre d'états de l'implémentation globale.

La modularité permet de diminuer la spécialisation de la structure du graphe de l'implémentation, mais cela se traduit par une augmentation du nombre de flèches et du nombre de convergences, donc de la dissipation.

7. Dissipation du calcul

Pour discuter de la dissipation du *calcul* en général, on peut considérer les machines de Turing [TUR 36]. Nous n'envisagerons ici que des machines de Turing *déterministes* et leur implémentation physique à l'aide d'automates dont le graphe est stable dans le temps et testable, ce qui définit une date de test (voir section 3).

Pour déterminer si les implémentations de machines de Turing doivent dissiper, on peut se demander s'il existe une implémentation physique qui soit, sur le plan logique, équivalente globalement à une machine de Turing, mais qui ne dissiperait pas.

La description logique de Turing comporte des parties qui communiquent entre elles : la *bande* et la *tête* (nous désignerons par « la tête » tout ce qui, dans la machine, n'est pas une bande). Ceci correspond à plusieurs automates finis :

— un automate dans la tête de la machine,

— une juxtaposition d'une infinité de cases mémoires dans la (ou les) bande(s). Chaque case d'une bande est un automate fini dont le nombre d'états est au moins égal au nombre de symboles de l'alphabet.

Deux automates finis (la tête et une case mémoire d'une bande) communiquent entre eux lors des lectures/écritures de la tête sur la bande et changent d'état éventuellement ensemble.

Si l'implémentation de la machine est calquée sur cette description, c'est-à-dire si c'est un assemblage de modules copiant les fonctions de ces parties et testables séparément, nous nous retrouvons dans le cas d'une implémentation modulaire et il y aura deux causes de dissipation proportionnelle à la durée du calcul :

— L'automate d'une case mémoire de la bande est fini alors que le nombre d'opérations d'écriture ne doit pas être borné. Ainsi, l'opération d'écriture d'un symbole rencontre périodiquement une convergence qui fait perdre une information précédemment mémorisée.

— L'automate de la tête peut comporter des convergences dans son graphe, ce qui est le cas au moins pour les machines de Turing universelles, d'après le lemme suivant.

Lemme : Le graphe de l'automate de tête d'une machine de Turing universelle comporte au moins une convergence.

Démonstration : Une machine de Turing universelle peut simuler toute machine de Turing, en particulier une qui se lance dans un calcul qui ne s'arrête pas. Comme son automate de tête est fini, il comporte au moins un circuit. Pour que ce circuit ne comporte pas de convergence, il faudrait qu'il contienne tous les états de l'automate. Son évolution serait alors périodique. Or il existe des calculs infinis non périodiques. La non-périodicité pour un automate fini implique au moins une convergence dans le graphe.

Cette modularité est-elle nécessaire ?

Si l'on veut garder l'aspect *programmable*, c'est-à-dire si l'on veut que la machine soit testable indépendamment du programme (sans tester tous les programmes possibles), alors il est clair que la tête et la bande doivent être portés par deux modules indépendants. On peut ainsi conclure qu'une implémentation de machine de Turing universelle est nécessairement modulaire et cela conduit à une dissipation proportionnelle au nombre d'étapes du calcul.

7.1. *Implémentations finies de machines de Turing spécialisées*

Dans la pratique, les implémentations de machine de Turing sont finies, c'est-à-dire avec une bande de longueur finie. Bien sûr, le caractère fini borne la complexité des calculs envisageables, mais tous les calculs d'une complexité inférieure à une borne fixée à l'avance peuvent être effectués, comme on le fait habituellement avec les ordinateurs qui sont aussi des machines finies.

Dans ce cadre, la programmabilité conduit encore à la modularité et à la dissipation. Mais si l'on considère une machine spécialisée pour un calcul unique, alors on peut se demander si un calcul particulier qui s'arrête peut être implémenté par un automate non dissipatif (pouvant trouver une implémentation non modulaire).

Si on considère un programme particulier (une bande INPUT particulière) qui conduit à un calcul qui s'arrête, alors l'assemblage tête + bande est globalement

équivalent à un automate linéaire non dissipatif, comme celui de la figure 13. En effet, comme la bande est *à l'intérieur*, c'est un automate fini sans entrée qui ne repasse jamais deux fois par le même état (sinon il repasserait indéfiniment par cet état et ne s'arrêterait pas). Il est donc clair que tout calcul qui s'arrête est un graphe non-dissipatif.

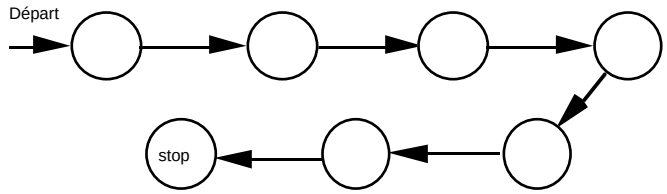

Figure 13. *Automate linéaire*

Mais la nécessité de la modularité s'introduit alors par un argument de taille en nombre d'états : une machine de Turing comportant une bande capable de mémoriser 100 bits comporte 2^{100} états (à multiplier par le nombre d'états de la tête), ce qui dépasse déjà les temps de test envisageables. Et pourtant, cette valeur de la taille mémoire est très faible.

Ainsi, même pour une machine finie, même pour une machine n'effectuant qu'un seul calcul (donc spécialisée et sans entrée), la modularité est nécessaire pour que les parties ne dépassent pas une *taille raisonnable* en nombre d'états.

7.2. *Machine réversible de Bennett*

Bennett a proposé une machine dont la dissipation n'est pas liée au temps de calcul et construite ainsi : à toute machine de Turing S (c'est-à-dire à tout automate de tête de machine de Turing), il est possible d'associer une machine de Turing réversible R qui fait évoluer de la même façon que S toute bande de travail, mais qui en outre mémorise l'historique du calcul et qui l'utilise ensuite pour *défaire* le calcul, l'ensemble étant déterministe et réversible [BEN 73]. D'autres exemples simples de machines de Turing réversibles ont été proposés depuis, voir par exemple [MOR 89]. Il s'agit de réversibilité logique pour le calcul global (Bennett écrit « whole machine state »). La preuve de Bennett exhibe pour la machine R un automate de tête que l'on peut expliciter avant le début du calcul (ne dépendant que de l'automate de tête de la machine S et non de la bande INPUT) et capable de se lancer dans tout calcul (sur toute bande fournie après la date de test de la tête).

Mais les conclusions de Bennett sur la réversibilité logique de la machine globale (tête + bandes) n'explicitent pas un graphe réversible de cette machine qui serait indépendant de la bande INPUT et qui permettrait une implémentation *testable avant le calcul*. En effet, seul le graphe de l'automate de tête est explicité avant le calcul, mais seule la *machine globale* (c'est-à-dire le calcul global) est réversible. Le graphe de la machine globale est spécialisé et fonction de la bande

INPUT. Il est explicitable, mais c'est le calcul lui-même qui l'explicite (si le calcul s'arrête). Il n'est donc connu qu'après la fin du calcul et ne peut donc pas être le graphe d'une implémentation testable avant le début du calcul.

La réversibilité logique du calcul global ne doit pas laisser croire à la possibilité d'implémenter une machine non dissipative en copiant la structure logique de Bennett, puisque le découpage tête/bande entraînerait nécessairement une modularité donc une dissipation proportionnelle à la durée du calcul.

Les graphes des modules pris séparément comportent des convergences alors que celui de la machine globale (du calcul global) n'en comporte pas. Ceci se comprend en pensant que l'information utile est permutée entre la tête et les bandes et n'est donc jamais perdue pour la machine globale, alors qu'elle est perdue par moments pour certains modules.

La machine globale est une machine spécialisée, fonction de la bande INPUT. Quand le calcul est terminé, on a rendu son graphe explicite. Il s'agit d'un graphe linéaire, comme celui de la figure 13, dont le nombre d'états est égal au nombre d'étapes de la machine R, c'est-à-dire : $4n + 4r + 5$, où n est le nombre d'étapes de la machine S, c'est-à-dire la complexité du calcul en temps et où r est le nombre de symboles du résultat.

La construction de Bennett ne permet donc pas d'implémenter une machine universelle non-dissipative, mais conduit à construire autant de machines réversibles que de calculs distincts. L'implémentation physique de telles machines reste, pour des raisons de testabilité, très fortement limitée en complexité.

8. Discussions

8.1. *Séparation logique*

La définition de l'implémentation physique d'un automate par son test contient implicitement l'idée que l'on peut faire une *séparation logique* entre intérieur et extérieur de l'automate et que les échanges d'information entre l'intérieur et l'extérieur (dans les deux sens) ne doivent emprunter que les voies définies par les entrées et les sorties. Cela signifie que l'on doit exclure la possibilité de corrélations, entre l'état physique interne de l'automate et son environnement, qui auraient une signification logique mais qui ne seraient pas liées aux échanges via les entrées/sorties. Il peut néanmoins exister des corrélations physiques si elles n'interviennent pas dans la fonction logique.

La séparation entre l'intérieur et l'extérieur n'est pas obligatoirement l'isolement d'une partie connexe de l'espace, qui serait limitée par la « cloison » de la boîte noire, mais elle revient à isoler fonctionnellement un système physique dont une partie des coordonnées définit l'état de l'automate, celui-ci ne connaissant son environnement que par le biais de l'information présente sur ses entrées. Par définition, un tel automate ignore l'origine de l'information présente sur ses entrées ainsi que l'usage fait de ses sorties. En outre, sa *dynamique logique* est indépendante de la nature des connexions de ses sorties. Cette séparation n'est pas une conséquence des lois physiques, mais c'est au contraire une hypothèse logique

nécessaire pour envisager la fiabilité et le test des implémentations.

L'intervention de la logique dans la physique ne se réduit pas à distinguer certains systèmes physiques devant remplir une fonction logique. Les automates sont aussi des éléments indispensables pour l'étude des systèmes physiques. En effet, la description des systèmes physiques nécessite d'effectuer sur ceux-ci des interventions particulières, permettant d'obtenir de l'information sur leur état, et appelées *mesures*. Par définition, l'information dégagée par la mesure doit pouvoir être mémorisée et traitée par des systèmes logiques. Les mesures font intervenir des automates, qui réalisent le nécessaire interface entre des observables physiques quantiques et des valeurs observées soumises à un traitement logique classique [WIT 63]. Ce rôle caractéristique de l'automate est illustré en particulier par le démon de Maxwell.

Séparation logique et modularité sont des propriétés fondamentales des implémentations physiques d'automates, imposées par la testabilité. Elles ont des conséquences importantes pour la constitution physique des automates. Elles impliquent que dans une implémentation modulaire, les connexions internes à l'automate qui sont des connexions externes aux modules (d'une sortie d'un module à l'entrée d'un autre module de la même implémentation) ne véhiculent d'autre information utilisable que testable classiquement. C'est-à-dire qu'une implémentation modulaire ne peut traduire que des fonctions séquentielles *classiques*. En d'autres termes, les réalisations non classiques que l'on peut envisager (exploitant des propriétés quantiques par exemple [DEU 85]) sont nécessairement non modulaires. Leur complexité est alors limitée par les contraintes de testabilité.

8.2. *Description d'automates à l'aide de fonctions récursives*

Nous avons considéré qu'un automate est défini par une liste finie d'états et de transitions et que cette liste est donnée explicitement. Cette restriction est rendue nécessaire pour la construction physique et le test de l'implémentation.

Mais d'un point de vue strictement formel, on peut se demander s'il est envisageable de décrire le graphe à l'aide d'une formule récursive. Il se pourrait alors qu'il soit difficile de dire si le graphe est fini, uniquement en regardant la formule de description. Si le graphe est le résultat d'un calcul, l'indécidabilité de l'arrêt du calcul se reportera sous forme d'une indécidabilité sur la structure et la finitude du graphe. Si le calcul du graphe s'arrête, ce calcul a pour conséquence d'expliciter le graphe et permet donc de construire l'implémentation et de la tester.

Une situation plus délicate serait que l'implémentation se construise en cours de calcul grâce à un autre automate (robot) qui assemblerait des composants à partir des instructions d'un programme. Cela a été exclu par notre définition qui impose le test avant l'utilisation. Mais cette éventualité pourrait permettre une autre forme d'existence de la machine réversible de Bennett. En effet, elle n'est pas explicite avant le calcul, mais son algorithme de construction est connu (à partir de la bande INPUT et de la tête spécifiée par Bennett) et c'est le calcul qui la construit. Dans ce cas, l'analyse de la dissipation ne peut pas être limitée au décompte des divergences

et des convergences dans le graphe résultat, mais doit aussi tenir compte du graphe du robot qui assemble les composants. On peut alors appliquer les discussions précédentes à cet ensemble.

9. Conclusion

Nous avons donné une définition de la dissipation logique d'une implémentation d'automate fini en fonction de son graphe.

Nous avons défini la modularité des implémentations d'automates et montré la dissipation liée à cette modularité. Une machine à calculer dissipe proportionnellement à la durée du calcul si la machine est programmable ou de taille raisonnable, car alors elle est nécessairement modulaire. Sinon, la spécialisation d'une machine non-dissipative ou la complexité de son test la rendent incompatible avec l'universalité sous-tendant la notion de calcul.

La modularité diminue la spécialisation. Elle permet d'atteindre des calculs plus complexes et plus variés avec un test plus simple. En contrepartie, elle entraîne de la dissipation. Les machines de Turing peuvent être à la fois universelles et simples. Par contre, la machine réversible de Bennett est spécialisée ou complexe à tester. La dissipation apparaît, à travers la modularité, comme ce qui permet à des machines simples d'effectuer des calculs variés et d'une complexité arbitraire.

Nous n'avons pas discuté des mécanismes physiques pouvant entraîner de la dissipation, ni *a fortiori* cherché à préciser quelle est la valeur du cœfficient de conversion entre dissipation logique et dissipation physique. Une telle discussion devrait être rapprochée d'une discussion des limitations physiques dans l'enregistrement des données, leur transmission et leur conservation (limitations liées à la sensibilité des mesures, à la causalité et à la stabilité). Nous avons vu que la dissipation logique est une condition nécessaire à l'implémentation de certains automates. Réciproquement, le rôle joué par les automates dans la mesure, donc dans le test, montre le lien étroit entre la dissipation physique et la *séparation logique* nécessaire au fonctionnement logique des implémentations physiques d'automates.

10. Remerciements

Nous remercions Guy Cousineau, Wladimir Mercouroff et Jacques Stern, ainsi que les lecteurs de la revue TSI, qui nous ont aidés à conduire ce texte vers sa forme actuelle.

11. Bibliographie

[BEN 73] BENNETT C. H., « Logical Reversibility of Computation », *IBM J. Res. Develop.*, p. 525-532, novembre 1973.

Technique et science informatiques. Volume 15 – n° 8/1996, pages 1079 à 1104

[BEN 87] BENNETT C. H., « Demons, Engines and the Second Law », *Scientific American*, p. 88-96, novembre 1987.

[BRI 59] BRILLOUIN L., *La Science et la Théorie de l'Information*, Masson et Cie, 1959, réédité par les Editions Jacques Gabay en 1988.

[DEU 85] DEUTSCH D., « Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer », *Proc. R. Soc.*, A400, p. 97-117, 1985.

[FEY 63] FEYNMAN R., *The Feynman Lectures on Physics*, Mécanique, chap. 46, Addison-Wesley Publish. Co., Reading Mass., 1963, traduction française Interéditions, 1979.

[GAB 51] GABOR D., « Light and Information », *Richie lecture*, University of Edimburg, 1951, publié dans : *Progress in Optics*, Vol. 1, Chap. 4, p. 109-153, 1961.

[LAN 61] LANDAUER R., « Irreversibility and Heat Generation in the Computing Process », *IBM Journal*, p. 183-191, juillet 1961.

[LEF 90] LEFF H. S., REX A. F., « Resource Letter MD-1: Maxwell's demon », *Am. J. Phys.*, Vol. 58, n° 3, p. 201-209, mars 1990.

[LIK 82] LIKHAREV K. K., « Classical and Quantum Limitations on Energy Consumption in Computation », *Int. J. of Theor. Physics*., Vol. 21, n° 12, p. 311-326, 1982.

[MAX 75] MAXWELL J. C., *Theory of Heat*, Text-books of Science, Longmans, Green & Co, 4ème édition, p. 328-329, Londres 1875, (1ère édition 1871).

[MIN 67] MINSKY M. L., *Computation : Finite and Infinite Machines*, Prentice-Hall, 1967.

[MOR 89] MORITA K., SHIRASAKI A., GONO Y., « A 1-Tape 2-Symbol Reversible Turing Machine », *Trans. IEICE Japan*, Vol. E72, p. 223-228, 1989.

[PER 95] PERRIN D., « Les débuts de la théorie des automates », *T. S. I.*, Vol. 14, n° 4, p. 409-433, 1995.

[SHA 49] SHANNON C. E., WEAVER W., *The Mathematical Theory of Communication*, The University of Illinois Press, 1949.

[SMO 12] SMOLUCHOWSKI M., « Experimentell nachweisbare der üblichen Thermodynamik widersprechende Molekularphänomene », *Phys. Z.*, Vol. 13, p. 1069-1080, 1912.

[STE 90] STERN J., *Fondements mathématiques de l'informatique*, Mc Graw Hill, 1990.

[SZI 29] SZILARD L., « Über die Entropieverminderung in einem thermodynamischen System bei Eingriffen intelligenter Wesen », *Zeitschrift für Physik*, Vol. 53, p. 840-856, 1929, traduction en langue anglaise : « On the Decrease of Entropy in a Thermodynamic System by the Intervention of Intelligent Beings », *Behavorial Science*, Vol. 9, p. 301-310, 1964.

[TUR 36] TURING A. M., « On Computable Numbers, with an Application to the Entscheidungs Problem », *Proceedings of the London Mathematical Society Ser 2*, Vol. 42, p. 230, 1936 et Vol. 43, p. 544-546, 1937.

[WIT 63] DE WITT B. S., « Dynamical Theory of Groups and Fields », *Relativity, Groups and Topology*, Les Houches, 1963, De Witt and de Witt eds, (Gordon Breach, Science Pubs., 1964), p. 605.