Introduction à l'algorithmique des objets partagés

Bernadette Charron–Bost
Robert Cori

Lix, Ecole Polytechnique,
91128 Palaiseau Cedex, France,
`charron@lix.polytechnique.fr`
`cori@lix.polytechnique.fr`

Antoine Petit

Lifac, ENS Cachan,
61 av. Président Wilson
94235 Cachan Cedex
`petit@lifac.ens-cachan.fr`

Résumé

Ces notes constituent une introduction à l'algorithmique répartie utilisant des mémoires partagées par plusieurs processus, on y présente les exécutions comme des ensembles dont les éléments sont des actions élémentaires ou opérations, et qui sont munis de deux relations satisfaisant certains axiomes. On introduit la notion de registre et l'on construit des registres complexes à l'aide de composants élémentaires. La source de cet article est principalement *On interprocess communication I* & *II* de L. Lamport.

1 Introduction

Il est d'usage de considérer deux paradigmes importants de communication en algorithmique répartie : la transmission de messages et l'utilisation d'objets partagés, nous nous plaçons ici dans le deuxième cadre. Un objet partagé est en quelque sorte un automate qui change d'état suivant les actions effectuées par les processus qui y font appel et qui fournit des valeurs à ces processus. La classification des objets partagés suivant leurs capacités à résoudre certains problèmes constitue un des objectifs majeurs des travaux récents en algorithmique répartie, à cela il faut ajouter les algorithmes de réalisation d'objets complexes à l'aide d'objets plus simples et la preuve de l'impossibilité de certaines réalisations. Dans cet article nous considérons les objets partagés les plus élémentaires : les registres sûrs et les registres atomiques et nous montrons comment se résolvent, sur ces objets particuliers, les questions citées plus haut. Nous commençons par décrire un modèle de représentation formelle des systèmes d'exécution distribués, ce modèle est basé sur la théorie mathématique des ensembles ordonnés, il est dû à L. Lamport. Nous considérons ensuite les registres en montrant que cette famille d'objets élémentaires possède des propriétés riches : on peut réaliser des algorithmes qui sont loin d'être triviaux, et on peut démontrer des résultats d'impossibilité en faisant appel à des arguments complexes. Ainsi en se limitant aux registres on a déjà une vue assez large des techniques employées dans le domaine et du style des résultats que l'on peut obtenir.

La source de notre présentation est principalement l'article de L. Lamport *On interprocess communication I* & *II*. Par rapport à cet article nous avons ajouté quelques notions supplémentaires en tentant d'être aussi rigoureux que possible. Le plus souvent les preuves complètes des résultats annoncés sont effectuées, en particulier celle de l'impossibilité de la réalisation d'un registre atomique, à un lecteur et un écrivain, par des registres réguliers du même type dans le cas où le lecteur n'écrit pas. Nous avons aussi tenté de montrer intuitivement le fonctionnement de tel algorithme ou de tel objet. La dernière partie est une présentation nouvelle de la construction d'un registre atomique à l'aide de registres réguliers due à J. Tromp. Nous montrons en particulier que l'algorithme de J. Tromp est valide en dehors de l'hypothèse du temps global. Nous espérons que cet article aiguisera la curiosité du lecteur et l'encouragera à consulter d'autres travaux dans le domaine.

2 Axiomatique des relations de causalité

Dans ce paragraphe nous introduisons le modèle mathématique qui servira de base à tous les développements de cet article. On considère un ensemble E d'actions élémentaires muni d'un ordre partiel, puis une famille de sous-ensembles de E, les opérations. On peut définir de manière naturelle deux relations $\longrightarrow$ et $\dashrightarrow$ sur les opérations, ces relations satisfont alors certaines conditions. En omettant l'ensemble sous-jacent E et en retenant comme structure de base l'ensemble des opérations muni des relations $\longrightarrow$ et $\dashrightarrow$, on obtient la notion d'exécution.

2.1 Exécution de système, modèle

Soit E un ensemble dont les éléments sont considérés comme des actions élémentaires, muni d'une relation d'ordre partiel $\prec$ strict modélisant la relation de causalité entre ces actions élémentaires. Dans la suite, $\preceq$ désignera la fermeture réflexive de la relation $\prec$. Une opération (ou action) non-élémentaire est constituée d'un ensemble non-vide d'actions élémentaires. C'est un élément de l'ensemble des parties non vides de E, ce dernier ensemble est noté $\mathcal{P}^*(E)$ dans la suite. La relation $\prec$ induit sur $\mathcal{P}^*(E)$ une relation notée $\longrightarrow$ définie de la façon suivante.

Pour deux opérations A et B, on a $A \longrightarrow B$ si :

$$\forall a \in A,\ \forall b \in B,\ a \prec b. \tag{1}$$

Puisque la relation $\prec$ modélise la dépendance causale entre opérations élémentaires, la relation $\longrightarrow$ traduit elle aussi une dépendance causale mais entre actions non–élémentaires. Dans la suite, nous dirons que A est une *cause* de B lorsque $A \longrightarrow B$. Pour qu'une action A influe sur une action B, il n'est pas nécessaire que $A \longrightarrow B$, il suffit seulement qu'une action élémentaire de A soit une cause d'une action élémentaire de B. Cette relation de "causalité faible" est ainsi modélisée par la relation notée $\dashrightarrow$ que l'on définit comme suit.

Pour deux opérations A et B on a $A \dashrightarrow B$ si :

$$\exists a \in A,\ \exists b \in B,\ a \preceq b. \tag{2}$$

Il est alors facile de vérifier que l'ensemble $\mathcal{P}^*(E)$, muni des deux relations $\longrightarrow$ et $\dashrightarrow$ satisfait les conditions suivantes :

(A_1) La relation $\longrightarrow$ est une relation d'ordre.

(A_2) Si $A \longrightarrow B$ alors $A \dashrightarrow B$ et $\neg(B \dashrightarrow A)$.

(A_3) Si $A \longrightarrow B \dashrightarrow C$ ou $A \dashrightarrow B \longrightarrow C$ alors $A \dashrightarrow C$.

(A_4) Si $A \longrightarrow B \dashrightarrow C \longrightarrow D$ alors $A \longrightarrow D$.

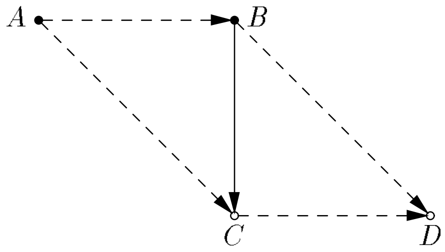


FIG. 1 - Une exécution n'admettant pas de modèle

Dans [L86a], Lamport considère l'axiome supplémentaire suivant :

(A_5) Pour tout A, l'ensemble des B tels que $\neg(A \longrightarrow B)$ est fini.

En utilisant en plus l'axiome (A_2), (A_5) assure que chaque action A a un ensemble fini de causes. Notons que contrairement aux axiomes A_1–A_4, l'axiome A_5 ne se déduit pas des définitions (1) et (2) de $\longrightarrow$ et $\dashrightarrow$ en termes de la relation d'ordre partiel $\prec$.

En omettant l'ensemble E et la relation $\prec$ on peut définir directement la notion d'exécution de système :

Définition 2.1 *Une exécution de système est un ensemble dénombrable $\mathcal{S}$ muni de deux relations $\longrightarrow$ et $\dashrightarrow$ satisfaisant les axiomes A_1–A_5. Les éléments de $\mathcal{S}$ sont appelés opérations, actions ou événements. La relation $\longrightarrow$ est dite causalité forte et $\dashrightarrow$ causalité faible.*

L'ensemble partiellement ordonné $(E, \prec)$ à partir duquel nous avons défini par (1) et (2) les deux relations $\longrightarrow$ et $\dashrightarrow$ est appelé *modèle* de $< \mathcal{P}^*(E), \longrightarrow, \dashrightarrow >$. Plus généralement, on définit un modèle d'une exécution de système de la façon suivante :

Définition 2.2 *Un ensemble partiellement ordonné $(E, \prec)$ est un* modèle *d'une exécution de système $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ s'il existe une application $\mu : \mathcal{S} \longrightarrow \mathcal{P}^*(E)$ telle que :*

$$A \longrightarrow B \iff \forall a \in \mu(A),\ \forall b \in \mu(B) :\ a \prec b$$
$$A \dashrightarrow B \iff \exists a \in \mu(A),\ \exists b \in \mu(B) :\ a \preceq b.$$

Il est alors naturel de se demander si toute exécution de système admet un modèle. La réponse est négative comme le montre l'exemple de l'exécution de système $\mathcal{S}$ décrite dans la Figure 1.
En effet, si $\mathcal{S}$ admettait un modèle alors il existerait a, b, c, d appartenant respectivement à $\mu(A)$, $\mu(B)$, $\mu(C)$ et $\mu(D)$ tels que $a \preceq b$ et $c \preceq d$ puisque $A \dashrightarrow B$ et $C \dashrightarrow D$. Comme $B \longrightarrow C$, on a $b \prec c$. Par transitivité de la relation $\prec$, on en déduit que $a \prec d$. Ainsi on a $A \dashrightarrow D$ qui n'est pas une relation de l'exécution de système $\mathcal{S}$.

Plus généralement, on peut remarquer que toute exécution de système $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ qui admet un modèle satisfait les propriétés suivantes :

(M_1) $\forall A \in \mathcal{S},\ A \dashrightarrow A$.

(M_2) Si $A \dashrightarrow B \longrightarrow C \dashrightarrow D$ alors $A \dashrightarrow D$.

Dans [Ang], Anger montre que les propriétés M_1 et M_2 suffisent en fait à assurer l'existence d'un modèle , nous reprenons ici cette preuve.

Théorème 2.3 *Une exécution de système admet un modèle si et seulement si elle satisfait M_1 et M_2.*

Preuve. La discussion qui précède montre que toute exécution de système qui admet un modèle satisfait M_1 et M_2.
Réciproquement, supposons qu'une exécution de système $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ satisfasse M_1 et M_2. Considérons deux copies disjointes $\mathcal{S}_d$ et $\mathcal{S}_f$ de $\mathcal{S}$ et soit E leur réunion. Pour toute opération A de $\mathcal{S}$, on note A_d et A_f les copies de A dans $\mathcal{S}_d$ et $\mathcal{S}_f$, de plus on munit E de la relation notée $\prec$ égale à la fermeture transitive de la relation $\stackrel{'}{\longrightarrow}$ que l'on définit comme suit.
Pour tout $A_d, B_d \in \mathcal{S}_d$, pour tout $A_f, B_f \in \mathcal{S}_f$ on a :

$$A_d \stackrel{'}{\longrightarrow} B_f \iff A \dashrightarrow B \tag{3}$$

$$A_f \stackrel{'}{\longrightarrow} B_d \iff A \longrightarrow B \tag{4}$$

En particulier, comme $\mathcal{S}$ satisfait M_1, pour toute exécution d'opération A, $A \dashrightarrow A$ et donc $A_d \stackrel{'}{\longrightarrow} A_f$. On peut alors considérer A_d et A_f comme modélisant respectivement le début et la fin de A.

Lemme 2.4 *La relation $\prec$ est une relation d'ordre sur E qui vérifie :*

$$A_f \prec B_d \iff A \longrightarrow B.$$

Preuve. Nous montrons d'abord que $A_f \prec B_d \iff A \longrightarrow B$, puis que $\prec$ est une relation d'ordre.

1) Par définition de $\stackrel{'}{\longrightarrow}$, si $A \longrightarrow B$ alors $A_f \stackrel{'}{\longrightarrow} B_d$ et $A_f \prec B_d$.

2) Réciproquement, supposons que $A_f \prec B_d$. Ou bien $A_f \stackrel{'}{\longrightarrow} B_d$ et le résultat est immédiat, ou bien il existe alors une chaîne n'utilisant que la relation $\stackrel{'}{\longrightarrow}$ reliant A_f à B_d. Cette chaîne est de la forme :

$$A_f = D_f^0 \stackrel{'}{\longrightarrow} C_d^1 \stackrel{'}{\longrightarrow} D_f^1 \stackrel{'}{\longrightarrow} C_d^2 \stackrel{'}{\longrightarrow} D_f^2 \stackrel{'}{\longrightarrow} \cdots \stackrel{'}{\longrightarrow} D_f^{n-1} \stackrel{'}{\longrightarrow} C_d^n = B_d. \tag{5}$$

Dans $\mathcal{S}$, ceci s'écrit encore :

$$A = D^0 \longrightarrow C^1 \dashrightarrow D^1 \longrightarrow C^2 \dashrightarrow D^2 \longrightarrow \cdots \dashrightarrow D^{n-1} \longrightarrow C^n = B.$$

En appliquant l'axiome (A_4), on obtient $A \longrightarrow C_2$, puis en l'utilisant de nouveau $A \longrightarrow C_3$ et finalement $A \longrightarrow B$.
Pour montrer que $\prec$ est une relation d'ordre il suffit de vérifier qu'elle est irréflexive, la transitivité découlant immédiatement de sa définition. Supposons qu'il existe un élément X de E tel que $X \prec X$.

- Si X est de la forme $X = A_d$ alors il existe B_f tel que $A_d \stackrel{'}{\longrightarrow} B_f \prec A_d$. Par (3) on déduit $A \dashrightarrow B$ et la première partie du Lemme entraîne que $B \longrightarrow A$. Ceci contredit l'axiome A_2.

- Le cas où X est de la forme $X = A_f$ se traite de façon analogue.

Nous avons ainsi montré que $\prec$ est une relation d'ordre. $\square$

Fin de la preuve du Théorème :
Considérons l'application μ qui à A dans $\mathcal{S}$ associe le sous ensemble $\{A_d, A_f\}$ de E, montrons que :

$$A \longrightarrow B \Longleftrightarrow \forall A_i \in \mu(A), \forall B_j \in \mu(B) : A_i \prec B_j.$$

Si $\forall A_i \in \mu(A)$, $\forall B_j \in \mu(B)$ on a $A_i \prec B_j$ on déduit $A_f \prec B_d$ et $A \longrightarrow B$ d'après le Lemme 2.4. Réciproquement, si $A_f \prec B_d$ on a aussi par transitivité de $\prec$, puisque $B_d \prec B_f$ et $A_d \prec A_f$:

$$A_d \prec A_f \prec B_d \prec B_f$$

Passons à la relation $\dashrightarrow$ et montrons que :

$$A \dashrightarrow B \Longleftrightarrow \exists A_i \in \mu(A), \exists B_j \in \mu(B) : A_i \preceq B_j$$

Comme la relation $\prec$ est la fermeture transitive de $\overset{\prime}{\longrightarrow}$, par (3) nous avons :

$$A \dashrightarrow B \Rightarrow A_d \prec B_f.$$

Réciproquement, supposons que $A_i \preceq B_j$,pour certains $(i, j) \in \{d, f\}$. En utilisant la transitivité de la relation $\prec$ ainsi que les relations $A_d \preceq A_i$ et $B_j \preceq B_f$, nous en déduisons $A_d \preceq B_f$. Par construction, $A_d \neq B_f$ et donc $A_d \prec B_f$. Comme la relation $\prec$ est la fermeture transitive de la relation $\overset{\prime}{\longrightarrow}$, il en résulte que, ou bien $A_d \overset{\prime}{\longrightarrow} B_f$ et alors $A \dashrightarrow B$, ou bien il existe C et D dans $\mathcal{S}$ tels que :

$$A_d \overset{\prime}{\longrightarrow} C_f \prec D_d \overset{\prime}{\longrightarrow} B_f.$$

Par (3), nous en déduisons que $A \dashrightarrow C$ et $D \dashrightarrow B$. D'autre part, grâce au Lemme 2.4, nous avons $C \longrightarrow D$. L'axiome M_2 nous permet de conclure que $A \dashrightarrow B$. En conclusion, $(E, \prec)$ est un ensemble partiellement ordonné qui constitue un modèle de $< \mathcal{S}, \longrightarrow, \dashrightarrow >$.$\square$

Remarque
Notons que la réunion $\cup_{A \in \mathcal{S}} \mu(A)$ est égale à E. D'autre part, pour toutes les exécutions d'opérations A et B nous avons :

$$A \neq B \Rightarrow \mu(A) \cap \mu(B) = \emptyset.$$

Ainsi, l'application μ est injective.

2.2 Modèle temps global

Lorsque les actions élémentaires sont réalisées par un seul processus (séquentiel) ou lorsque le système est assujetti à une horloge globale, l'ensemble $(E, \prec)$ est totalement ordonné et chaque exécution d'opération A correspond à un intervalle fermé non vide de E que l'on note $[d_A, f_A]$. On dit alors que $(E, \prec)$ constitue un modèle *temps global* du système d'exécution. Il est clair que toute exécution de système admettant un modèle temps global vérifie la condition :

(GT) Pour tout A, B appartenant à $\mathcal{S}$, on a : $A \longrightarrow B$ ou $B \dashrightarrow A$

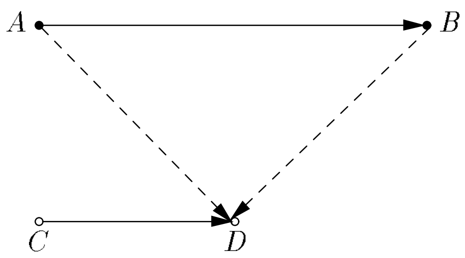


FIG. 2 - Une exécution n'admettant pas de modèle temps global

puisqu'on a nécessairement $f_A \prec d_B$ ou $d_B \preceq f_A$. Notons qu'une exécution de système, même si elle admet un modèle, ne satisfait pas nécessairement GT et n'admet donc pas en général de modèle temps global (cf. Figure 2). Autrement dit, l'axiome GT ne se déduit pas des axiomes A_1-A_5 et M_1-M_2.

Dans la suite de cette section, nous nous proposons de montrer que la condition GT est en fait suffisante pour qu'une exécution de système admette un modèle temps global[1]. Pour cela, nous rappelons tout d'abord le théorème de Russell–Wiener ([Wie]), et nous en donnons la preuve, bien qu'elle soit classique, par souci d'être complets.

Théorème 2.5 *Pour tout ensemble partiellement ordonné* $< \mathcal{S}, \longrightarrow >$*, les deux conditions suivantes sont équivalentes :*

- (i) *Il existe un ensemble totalement ordonné* $(E, <)$ *et une application* $\mu : \mathcal{S} \longrightarrow Int^*(E)$ *(où* $Int^*(E)$ *désigne l'ensemble des intervalles non vides de* E*) telle que :*

$$A \longrightarrow B \Longleftrightarrow \forall X \in \mu(A),\ \forall Y \in \mu(B) : X < Y$$

- (ii) *Le système* $< \mathcal{S}, \longrightarrow >$ *satisfait* l'axiome de Russell–Wiener[2],

$$\forall A, B, C, D \in \mathcal{S}, \quad (A \longrightarrow B) \wedge \neg(C \longrightarrow B) \wedge (C \longrightarrow D) \Rightarrow A \longrightarrow D. \tag{6}$$

Preuve. Le fait que (i) implique (ii) est immédiat. Soit un ensemble totalement ordonné $< \mathcal{S}, \longrightarrow >$ qui satisfait l'axiome de Russel-Wiener, considérons les antichaînes maximales (au sens de l'inclusion) de $\mathcal{S}$; ces antichaînes sont généralement appelées *moments* de $\mathcal{S}$. Notons E l'ensemble des moments de $\mathcal{S}$. Définissons sur E la relation $<$ par :

$$X < Y \Longleftrightarrow \exists A \in X,\ \exists B \in Y\ : A \longrightarrow B.$$

La transitivité de la relation $<$ résulte de l'axiome de Russell–Wiener. D'autre part, tout moment étant une antichaîne, la relation $<$ est acyclique et ordonne donc E. Enfin, par maximalité des moments, l'ordre $<$ est un ordre total.

[1] Notons que toute exécution de système qui satisfait l'axiome GT satisfait aussi M_1-M_2 et admet donc un modèle.

[2] Les ensembles partiellement ordonnés satisfaisant cet axiome sont appelés des *ordres d'intervalle*. Le lecteur intéressé pourra trouver différentes caractérisations de ces ensembles partiellement ordonnés dans [Mor].

Pour chaque élément A de $\mathcal{S}$, notons $\mu(A)$ l'ensemble de tous les moments X contenant A. Comme $\{A\}$ est une antichaîne, il existe un moment contenant A et donc $\mu(A)$ est non vide. Montrons que $\mu(A)$ est un intervalle de E. Soient X, Y, Z trois moments tels que $X < Y < Z$. Supposons que X et Z appartiennent à $\mu(A)$. Si Y n'est pas dans $\mu(A)$ alors, par maximalité de l'antichaîne Y, il existe un élément B dans Y tel que $A \longrightarrow B$ ou $B \longrightarrow A$. Etudions le cas où $A \longrightarrow B$. Comme $Y < Z$, il existe un élément C dans Y et un élément D dans Z tels que $C \longrightarrow D$. En utilisant de nouveau le fait que Y est une antichaîne maximale, nous avons $\neg(C \longrightarrow B)$. L'axiome de Russell–Wiener nous assure qu'alors $A \longrightarrow D$, ce qui contredit le fait que Z est une antichaîne. Le cas où $B \longrightarrow A$ se traite de façon analogue.

Montrons maintenant que pour tout A, B de E,

$$A \longrightarrow B \Longleftrightarrow \forall X \in \mu(A), \forall Y \in \mu(B) : X < Y.$$

Supposons que $A \longrightarrow B$. Considérons deux moments quelconques X et Y tels que $X \in \mu(A)$ et $Y \in \mu(B)$, i.e. $A \in X$ et $B \in Y$. Par définition de la relation $<$, on en déduit que $X < Y$.

Réciproquement, supposons que

$$\forall X \in \mu(A), \forall Y \in \mu(B) : X < Y$$

avec $\neg(A \longrightarrow B)$, i.e. avec A et B incomparables pour la relation $\longrightarrow$ ou avec $B \longrightarrow A$. Si $B \longrightarrow A$ alors, d'après ce qui précède, pour tout $X \in \mu(A)$ et tout $Y \in \mu(B)$ on a $Y < X$. Comme la relation $<$ est acyclique, ceci contredit l'hypothèse de départ.
Si A et B sont incomparables pour la relation $\longrightarrow$ alors tout moment de $\mathcal{S}$ contenant A contient aussi B et réciproquement. Par suite, dans ce cas, $\mu(A) = \mu(B)$ et, comme $<$ est anti–réflexive, l'hypothèse de départ n'est pas vérifiée (en particulier pour $Y = X$). $\Box$

Remarque
Soit $\mathcal{S}$ un ensemble muni d'une relation d'ordre $\longrightarrow$ satisfaisant l'axiome de Russel-Wiener. En définissant la relation $\dashrightarrow$ par

$$\forall A, B \in \mathcal{S},\ A \dashrightarrow B \Longleftrightarrow \neg(B \longrightarrow A)$$

les deux relations $\longrightarrow$ et $\dashrightarrow$ constituent une exécution. En effet les axiomes A_1 et A_2 sont des conséquences immédiates des définitions, A_3 résulte de la transitivité de $\longrightarrow$ et A_4 découle de GT, ainsi l'axiome A_4 peut être considéré comme une version affaiblie de l'axiome de Russell–Wiener.

Corollaire 2.6 *Une exécution de système admet un modèle temps global si et seulement si elle satisfait l'axiome GT.*

Preuve. Nous avons précédemment remarqué que toute exécution de système admettant un modèle temps global satisfait GT. Réciproquement, considérons $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ une exécution de système satisfaisant l'axiome GT. Etant donné que $\mathcal{S}$ satisfait A_4, cette exécution de système satisfait aussi l'axiome de Russell–Wiener et donc, par le Théorème 2.5, il existe un ensemble totalement ordonné $(E, <)$ et une application $\mu : \mathcal{S} \longrightarrow Int^*(E)$ telle que

$$A \longrightarrow B \Longleftrightarrow \forall X \in \mu(A),\ \forall Y \in \mu(B) : X < Y.$$

Par suite, pour les opérations A et B de $\mathcal{S}$, on a

$$A \dashrightarrow B \Longleftrightarrow \exists X \in \mu(A), \exists Y \in \mu(B) : X < Y \text{ ou } X = Y.$$

D'autre part, d'après l'axiome A_5, les ensembles $\mu(A)$ sont (par construction) finis et peuvent s'écrire :

$$\mu(A) = [d_A, f_A].$$

Par conséquent, $(E, <)$ est un modèle temps global de $< \mathcal{S}, \longrightarrow, \dashrightarrow >$. □

Le théorème de Ore (cf. [Ore]) donne une caractérisation de la *dimension* d'un ordre partiel en termes de *plongement* dans les ensembles $\mathbf{R}^k$, tout ensemble totalement ordonné est isomorphe à un sous–ensemble (totalement ordonné) de $\mathbf{R}$. Notons ϕ un plongement quelconque de E dans $\mathbf{R}$ et considérons, pour toute exécution d'opération A, l'ensemble $\phi(\mu(A))$ égal à l'image de $\mu(A)$ par ϕ. Cet ensemble est une partie finie non vide de $\mathbf{R}$ et le plus petit intervalle qui le contient est égal à $[\phi(d_A), \phi(f_A)] = I_A$. Ceci prouve que l'existence d'un modèle temps global est équivalente à l'existence d'un modèle temps global pour lequel $E = \mathbf{R}$. On retrouve ainsi la définition originale d'un modèle temps global donnée par Lamport dans [L86a].

3 Vues hiérarchiques

Pour décrire le comportement d'un système, on peut se placer à différents niveaux de détail. Une opération dite de haut niveau consiste ou encore est implantée par un ensemble d'opérations de plus bas niveau. Dans cette section, nous préciserons les relations qui existent entre deux vues différentes d'un même système, l'une étant de plus bas niveau que l'autre. D'autre part, nous donnerons une définition formelle de ce que signifie le fait d'implanter un système à l'aide d'un système de plus bas niveau.

3.1 Vue de niveau supérieur

Considérons une exécution de système $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ et soit $\mathcal{H}$ un ensemble d'éléments sensés modéliser des exécutions d'opérations de niveau supérieur à celles de $\mathcal{S}$. Puisqu'une opération dite de haut niveau consiste en un ensemble d'opérations de plus bas niveau, chaque élément de $\mathcal{H}$ est une partie de $\mathcal{S}$. On définit alors naturellement sur $\mathcal{H}$ les relations de précédence $\stackrel{*}{\longrightarrow}$ et $\stackrel{*}{\dashrightarrow}$ de la façon suivante :

$$G \stackrel{*}{\longrightarrow} H \;\text{ si pour tout }\; A \in G \;\text{ et pour tout }\; B \in H \,\text{on a :}\, A \longrightarrow B \tag{7}$$

$$G \stackrel{*}{\dashrightarrow} H \;\text{ s'il existe }\; A \in G \;\text{ et }\; B \in H \;\text{ tels que }\; A \longrightarrow B \,\text{ou}\; A = B. \tag{8}$$

Etant donnés (7) et (8) et puisque $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ satisfait A_1-A_4, il suffit que tout élément de $\mathcal{H}$ soit une partie non vide de $\mathcal{S}$ pour que $< \mathcal{H}, \stackrel{*}{\longrightarrow}, \stackrel{*}{\dashrightarrow} >$ satisfasse aussi à A_1-A_4. En ce qui concerne l'axiome A_5, il suffit que tout élément de $\mathcal{H}$ soit une partie finie de $\mathcal{S}$ et que chaque élément de $\mathcal{S}$ appartienne à un nombre fini d'éléments de $\mathcal{H}$ pour que cet axiome soit vérifié par $< \mathcal{H}, \stackrel{*}{\longrightarrow}, \stackrel{*}{\dashrightarrow} >$. Puisqu'il est naturel de supposer que toute exécution d'opération de bas niveau fait partie de l'exécution d'une opération de plus haut niveau, nous posons la définition suivante :

Définition 3.1 *Une* vue *d'une exécution de système* $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ de niveau supérieur *est donnée par un sous ensemble $\mathcal{H}$ de $\mathcal{P}^*(\mathcal{S})$ tel que :*

(H_1) *Tout élément H de $\mathcal{H}$ est une partie finie de $\mathcal{S}$.*

(H_2) *Tout élément de $\mathcal{S}$ appartient à un nombre fini non nul d'éléments de $\mathcal{H}$.*

Dans la plupart des cas, les ensembles $\mathcal{H}$ que l'on considère sont des partitions (en parties finies) de $\mathcal{S}$. Cependant, la Définition 3.1 permet de raisonner sur des exécutions de système dans lesquelles une opération de bas niveau fait partie de plusieurs opérations de plus haut niveau.
Il est facile de vérifier que si une exécution de système $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ admet un modèle, i.e. vérifie M_1-M_2, alors toute vue de niveau supérieur vérifie aussi M_1-M_2. Une façon plus constructive de montrer cela est de remarquer que si $(E, \prec)$ est un modèle de $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ pour une certaine application μ alors l'application μ^* définie par :

$$\begin{array}{rcrcl} \mu^* & : & \mathcal{H} & \longrightarrow & \mathcal{P}^*(E) \\ & & H & \longmapsto & \mu^*(H) = \cup_{A \in H} \mu(A) \end{array}$$

fait de $(E, \prec)$ un modèle d'une vue $\mathcal{H} \subseteq \mathcal{P}^*(\mathcal{S})$ de niveau supérieur.

3.2 Notion d'implantation

Nous nous proposons ici de définir formellement le fait qu'une exécution de système en implante une autre. Si une exécution de système $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ implante une autre exécution de système $< \mathcal{H}, \xrightarrow{\mathcal{H}}, \overset{\mathcal{H}}{\dashrightarrow} >$ alors $\mathcal{H}$ doit clairement constituer une vue de $\mathcal{S}$ de niveau supérieur. Mais que dire des relations de précédence $\xrightarrow{\mathcal{H}}$ et $\overset{\mathcal{H}}{\dashrightarrow}$? Les éléments de $\mathcal{S}$ représentent les exécutions d'opération qui ont réellement lieu tandis que les éléments de $\mathcal{H}$ représentent les groupements d'opérations que l'on veut considérer à un niveau d'abstraction plus élevé. Les relations $\xrightarrow{*}$ et $\overset{*}{\dashrightarrow}$, induites par $\longrightarrow$ et $\dashrightarrow$ sur $\mathcal{H}$, constituent les relations de précédence réelles sur les éléments de $\mathcal{H}$ tandis que $\xrightarrow{\mathcal{H}}$ et $\overset{\mathcal{H}}{\dashrightarrow}$ représentent celles que l'on aimerait pouvoir considérer sur $\mathcal{H}$.

Ainsi, par exemple dans le domaine des bases de données, les éléments de $\mathcal{S}$ représentent les lectures et écritures d'un seul item tandis que les éléments de $\mathcal{H}$ modélisent les *transactions*, i.e. les lectures et écritures d'items multiples. La possibilité de *sérialiser* correspond au fait de pouvoir considérer les différentes transactions comme ayant lieu séquentiellement alors qu'elles sont en fait exécutées de façon concurrente. Sur cet exemple, on constate que pour définir une notion d'implantation "raisonnable", il est beaucoup trop contraignant de demander que l'ordre réel $\xrightarrow{*}$ soit égal à l'ordre total $\xrightarrow{\mathcal{H}}$ car cela interdirait toute exécution concurrente des opérations de base. On impose donc seulement que, si dans une vue de niveau supérieure une exécution d'opération en précède réellement une autre alors cette précédence est compatible avec les relations de précédence que l'on aimerait pouvoir considérer. Plus formellement, on définit l'implantation d'une exécution de système par une autre de la façon suivante :

Définition 3.2 *Une exécution de système* $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ implante *une autre exécution de système* $< \mathcal{H}, \xrightarrow{\mathcal{H}}, \overset{\mathcal{H}}{\dashrightarrow} >$ *si* $\mathcal{H}$ *est une vue de* $\mathcal{S}$ *de niveau supérieur et si la relation d'ordre* $\xrightarrow{\mathcal{H}}$ *est une extension de la relation d'ordre* $\xrightarrow{*}$ *définie par (7) sur* $\mathcal{H}$*, i.e. si* $\xrightarrow{*} \subseteq \xrightarrow{\mathcal{H}}$.

La proposition suivante prouve que lorsque l'on veut implanter une exécution de système par une autre exécution de plus bas niveau il suffit de considérer les opérations qui figurent réellement dans le système que l'on veut implanter et négliger toutes les opérations appartenant aux systèmes environnants. On montre ainsi que la relation "implante" est en quelque sorte une congruence.

Proposition 3.3 *Soit* $< \mathcal{S} \cup \mathcal{I}, \longrightarrow, \dashrightarrow >$ *une exécution de système où* $\mathcal{S}$ *et* $\mathcal{I}$ *sont deux ensembles disjoints. Soit* $< \mathcal{H}, \xrightarrow{\mathcal{H}}, \overset{\mathcal{H}}{\dashrightarrow} >$ *une exécution de système implantée par* $< \mathcal{S}, \longrightarrow, \dashrightarrow >$ *et soient* $\xrightarrow{*}$ *et* $\overset{*}{\dashrightarrow}$ *les deux relations définies par (7) et (8) sur* $\mathcal{S} \cup \mathcal{I}$*. Alors il existe des relations de précédence* $\xrightarrow{\mathcal{H} \cup \mathcal{I}}$ *et* $\overset{\mathcal{H} \cup \mathcal{I}}{\dashrightarrow}$ *sur* $\mathcal{S} \cup \mathcal{I}$ *telles que :*

– $< \mathcal{H} \cup \mathcal{I}, \xrightarrow{\mathcal{H}\cup\mathcal{I}}, \overset{\mathcal{H}\cup\mathcal{I}}{\dashrightarrow} >$ *est une exécution de système qui est implantée par* $< \mathcal{S} \cup \mathcal{I}, \longrightarrow, \dashrightarrow >$;

– *Les restrictions à* $\mathcal{H}$ *de* $\xrightarrow{\mathcal{H}\cup\mathcal{I}}$ *et* $\overset{\mathcal{H}\cup\mathcal{I}}{\dashrightarrow}$ *sont respectivement égales à* $\xrightarrow{\mathcal{H}}$ *et* $\overset{\mathcal{H}}{\dashrightarrow}$ *;*

– *Les restrictions à* $\mathcal{I}$ *de* $\xrightarrow{\mathcal{H}\cup\mathcal{I}}$ *et* $\overset{\mathcal{H}\cup\mathcal{I}}{\dashrightarrow}$ *sont des extensions de* $\longrightarrow$ *et* $\dashrightarrow$ *respectivement.*

Ce résultat technique est crucial dans les applications car il permet de prouver qu'une exécution de système en implante une autre quel que soit le contexte, et donc quelles que soient les opérations qui ont lieu dans le système et qui ne sont pas observables. Par exemple, pour implanter une base de données, on peut ne considérer que les opérations de lecture et d'écriture et faire abstraction de toutes les procédures d'initialisation et d'entrée–sortie. La Proposition 3.3 assure que le fait de considérer que les transactions ont lieu selon un ordre total $\xrightarrow{\mathcal{H}}$ n'induit pas sur l'ensemble de toutes les exécutions opération (observables ou non) des relations de précédence incohérentes avec celles qui existent réellement. Nous renvoyons à [L85] pour la preuve quelque peu technique de cette proposition.

4 Registres

Dans ce qui suit, on considère des exécutions dans lesquelles toutes les opérations sont soit des actions internes à des processus, soit des accès à des objets partagés. Les objets partagés étudiés ici sont des registres; on peut les considérer comme les objets partagés les plus simples. Les opérations que l'on peut effectuer sur un registre sont des *écritures* et des *lectures*. Une écriture attribue une valeur v au registre : elle consiste à "remplacer l'ancienne valeur par v". Une lecture obtient une valeur de la part du registre. On classifie les registres selon les conditions satisfaites par les valeurs obtenues lors d'une lecture, celle-ci peut être ou non concurrente à une, ou même à plusieurs, écritures. On peut considérer qu'un *registre* généralise la notion de mémoire, fondamentale en algorithmique séquentielle; dans ce contexte, l'écriture représente la notion classique d'affectation d'une valeur à une variable et la lecture celle de la recherche de la valeur affectée. Dans la suite on suppose qu'un registre est à un seul écrivain, sa complexité dépend alors des paramètres suivants :

– Le nombre de valeurs qu'il peut prendre. On distinguera principalement les registres binaires, qui ne prennent que deux valeurs différentes, et les registres à $k > 2$ valeurs distinctes.

– Le nombre de lecteurs qui accèdent au registre; le registre peut être ainsi à un seul lecteur ou à plusieurs lecteurs.

Lorsque la suite de lectures et d'écritures est totalement ordonnée par la relation $\longrightarrow$, il est naturel de considérer que la valeur obtenue par une lecture L est celle attribuée par la dernière écriture E qui précède L. Si les lectures et écritures se produisent de façon concurrente, plusieurs hypothèses peuvent être formulées sur les valeurs obtenues lors d'une lecture. Nous en retenons deux qui correspondent à des situations rencontrées fréquemment :

1. *Registres sûrs.* On considère ici l'hypothèse minimale que l'on puisse raisonnablement retenir. Dans un registre sûr, la valeur obtenue lors d'une lecture L qui n'est pas concurrente avec une écriture est celle qui a été écrite par la dernière écriture qui a précédé L. Aucune hypothèse n'est faite sur la valeur lue par une lecture concurrente à une ou plusieurs écritures; cette valeur peut être quelconque et n'avoir aucun rapport avec les valeurs écrites.

2. *Registres atomiques.* La notion d'atomicité, fréquente en algorithmique répartie, consiste à supposer que l'exécution s'est déroulée "comme si elle s'était produite de manière séquentielle ". Ainsi, l'ensemble des lectures et écritures sur un registre atomique peut être ordonné totalement par un ordre $\prec$, de façon telle que le comportement du registre soit le même que celui d'un registre pour lequel tous les accès se font séquentiellement. C'est à dire que la valeur obtenue lors d'une lecture L est celle attribuée par la dernière écriture qui précède L par l'ordre $\prec$.

Dans la suite nous examinerons comment réaliser des registres complexes à l'aide de registres plus simples; ainsi on réalisera par exemple un registre à un nombre quelconque de valeurs à l'aide de registres binaires, un registre multi-lecteurs à l'aide de registres à un seul lecteur, un registre atomique à l'aide de registres sûrs.

4.1 Axiomatique des registres

Nous commençons par préciser les notions informelles que nous venons d'introduire.

4.1.1 Exécutions de registres

Une exécution est, comme au paragraphe précédent, un ensemble $\mathcal{S}$ muni de deux relations $\longrightarrow$ et $\dashrightarrow$ satisfaisant aux axiomes $A_1 - A_5$. Une *exécution de registres* fait intervenir dans l'ensemble $\mathcal{S}$ certains événements qui sont des lectures et d'autres des écritures.

Définition 4.1 *Une* exécution d'un registre $(\mathcal{S}, \longrightarrow, \dashrightarrow, \mathcal{E}, \mathcal{L})$ *est une exécution où l'ensemble des opérations $\mathcal{S}$ contient deux sous-ensembles disjoints $\mathcal{E}, \mathcal{L}$ dont les éléments sont appelés respectivement écritures et lectures tels que :*

(B_0) *L'ensemble des écritures est totalement ordonné par la relation $\longrightarrow$ et la première écriture précède par $\longrightarrow$ tous les événements de $\mathcal{L} \cup \mathcal{E}$;*

(B_1) *Deux opérations quelconques E et L, appartenant respectivement à $\mathcal{E}$ et $\mathcal{L}$, sont en relation par $\dashrightarrow$, c'est-à-dire :*

$$\forall E \in \mathcal{E}, \quad \forall L \in \mathcal{L}, \qquad E \dashrightarrow L \text{ ou } L \dashrightarrow E$$

Remarques

1. L'axiome B_0 résulte de ce que l'on ne considère que des registres à un seul écrivain.[3]

2. Les relations entre les événements autres que les lectures et les écritures sont supposées quelconques, il n'est fait aucune hypothèse sur le lien entre ceux-ci et les éléments de $\mathcal{L}$ et $\mathcal{E}$, seuls les axiomes $A_1 - A_5$ leur sont applicables.

Les écritures d'une exécution d'un registre forment une suite (qui peut être infinie) :

$$E_0 \longrightarrow E_1 \longrightarrow E_2 \cdots \longrightarrow E_i \longrightarrow E_{i+1} \longrightarrow \cdots$$

Pour chaque lecture L, cette suite est partitionnée en cinq intervalles disjoints définis par :

[3] Des registres à plusieurs écrivains sont considérés par exemple par Israëli dans [IS].

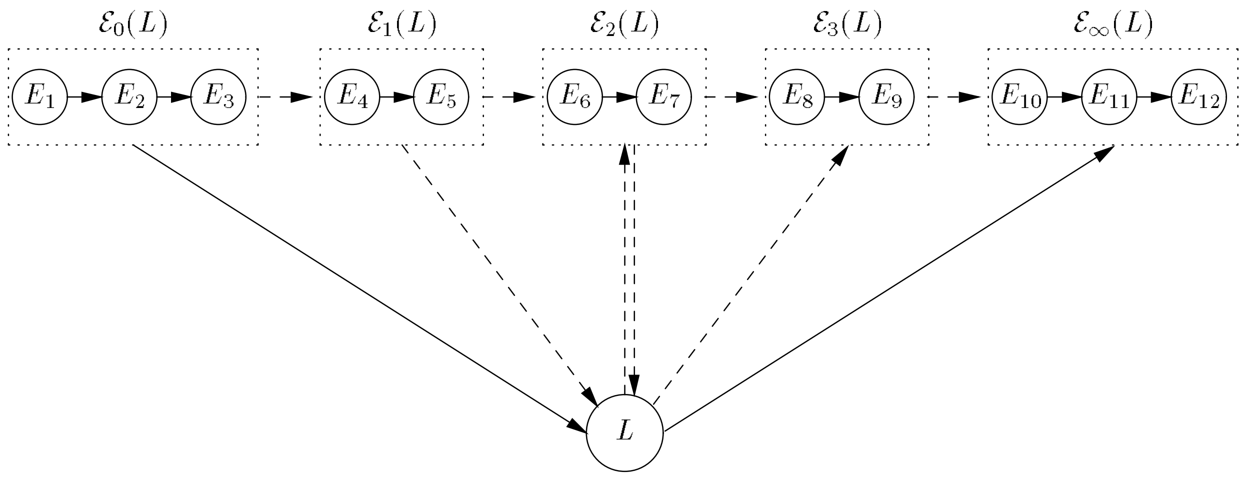


FIG. 3 - Partition d'une suite d'écritures par une lecture

$$\left\{\begin{array}{lcl}
\mathcal{E}_0(L) & = & \{E \in \mathcal{E} | E \longrightarrow L\} \\
\mathcal{E}_1(L) & = & \{E \in \mathcal{E} | E \dashrightarrow L, \neg(L \dashrightarrow E), \neg(E \longrightarrow L)\} \\
\mathcal{E}_2(L) & = & \{E \in \mathcal{E} | E \dashrightarrow L, L \dashrightarrow E\} \\
\mathcal{E}_3(L) & = & \{E \in \mathcal{E} | L \dashrightarrow E, \neg(E \dashrightarrow L), \neg(L \longrightarrow E)\} \\
\mathcal{E}_\infty(L) & = & \{E \in \mathcal{E} | L \longrightarrow E\}.
\end{array}\right.$$

Cette décomposition est illustrée sur la Figure 3

Les écritures de $\mathcal{E}_1(L), \mathcal{E}_2(L), \mathcal{E}_3(L)$ sont dites *concurrentes* à L. On dit qu'une lecture L est *isolée* si elle n'est concurrente à aucune écriture soit si :

$$\mathcal{E}_1(L) = \mathcal{E}_2(L) = \mathcal{E}_3(L) = \emptyset$$

dans ce cas toute écriture précède ou suit L par $\longrightarrow$.

Remarques

1. On vérifie, en utilisant les axiomes $A1 - A5$ sur les exécutions que si $E \in \mathcal{E}_i(L)$ et si $E \longrightarrow E'$, alors $E' \in \mathcal{E}_j(L)$ pour un indice j supérieur ou égal à i.

2. Si l'on se place dans l'hypothèse du temps global (GT), les ensembles $\mathcal{E}_1(L)$ et $\mathcal{E}_3(L)$ sont alors vides pour toute lecture L.

4.1.2 Exécutions valuées et registres

Jusqu'ici, nous avons considéré des événements de lecture et d'écriture sans prendre en compte les *valeurs* lues ou écrites. Un moyen de les prendre en compte consiste à faire intervenir des exécutions *valuées*, pour ces exécutions une valeur est affectée à chaque écriture ; et des exécutions *bi-valuées*, où une valeur est aussi attribuée à chaque lecture.

Soit V un ensemble fini dont les éléments sont appelés des *valeurs*; dans la suite on considérera que V contient ou bien deux éléments, (registres binaires) ou bien $k > 2$ (registres à k valeurs). Une *exécution valuée* est donnée par une exécution de registre $(\mathcal{S},\ \longrightarrow\ ,\ \dashrightarrow\ ,\ \mathcal{L},\ \mathcal{E})$ et une application ε de $\mathcal{E}$ dans V. Pour chaque écriture E, $\varepsilon(E)$ représente la valeur écrite dans le registre lors de l'opération d'écriture E.

Une *exécution bi-valuée* est une exécution valuée augmentée d'une application λ de $\mathcal{L}$ dans V. Pour une lecture L, $\lambda(L)$ modélise la valeur obtenue par L dans le registre, c'est la valeur lue par le processus qui effectue l'action L.

Dans certains cas on souhaite exprimer qu'une valeur lue dans un registre au cours de la lecture L a été écrite au cours de l'écriture E ; pour cela on définit une *fonction de lecture*, c'est une application de $\mathcal{L}$ dans $\mathcal{E}$. On dit qu'une exécution $(\mathcal{S},\ \longrightarrow\ ,\ \dashrightarrow\ ,\mathcal{L},\mathcal{E})$, bi-valuée par (λ,ε) est *induite* par la fonction de lecture π, si π est une application de $\mathcal{L}$ dans $\mathcal{E}$ et si pour toute lecture L on a

$$\lambda(L) = \varepsilon(\pi(L))$$

Une exécution bi-valuée $(\mathcal{S},\ \longrightarrow\ ,\ \dashrightarrow\ ,\mathcal{L},\mathcal{E},\lambda,\varepsilon)$, n'est pas nécessairement induite par une fonction de lecture, car les lectures peuvent très bien obtenir un résultat qui n'a jamais été écrit. D'autre part si une telle fonction de lecture existe elle n'est pas nécessairement unique.

On considère généralement un registre comme un objet partagé qui enregistre une valeur lorsqu'un processus écrit et qui fournit une valeur lorsqu'un processus lit. A cette façon dynamique de voir le registre on en préfère ici une autre, équivalente à la précédente, mais qui se prête mieux à un cadre plus formel. Ainsi, dans ce qui suit on considère un registre comme une fonction, qui a pour donnée une exécution valuée (où les écritures sont affectées d'une valeur), et dont le résultat est de donner des valeurs aux lectures, obtenant ainsi une exécution bi-valuée. De façon plus précise on a :

Définition 4.2 *Un registre est une fonction qui, à toute exécution valuée* $(\mathcal{S},\ \longrightarrow\ ,\ \dashrightarrow\ ,\mathcal{E},\mathcal{L},\varepsilon)$, *associe une application* λ *de* $\mathcal{L}$ *dans* V *telle que pour toute lecture* L, $\lambda(L)$ *ne dépend que de l'ensemble des événements qui précèdent* L *par* $\dashrightarrow$, *des relations* $\longrightarrow$ *ou* $\dashrightarrow$ *qui existent entre ceux-ci et des valuations des écritures qui font partie de cet ensemble.*

Remarque.

Dans cette définition un registre est un objet déterministe qui donne la même valeur en lecture pour deux exécutions valuées identiques. On se limite aux registres déterministes ici. Pour définir des registres non déterministes, il suffirait de considérer qu'un registre est une fonction de l'ensemble des exécutions valuées dans l'ensemble des parties des exécutions bi-valuées, ou de manière équivalente que c'est une famille de couples formés d'une exécution valuée et d'une exécution bi-valuée; des résultats semblables à ceux donnés ci-dessous pourraient alors être développés.

4.1.3 Registres sûrs et atomiques

Dans la définition d'une exécution bi-valuée, qui a été donnée ci-dessus, aucune restriction n'a été imposée à la valeur obtenue par une lecture. Celle-ci peut être écrite avant celle-ci, après celle-ci, ou même être une valeur qui n'a pas été écrite. Afin de modéliser des situations rencontrées pratiquement dans les systèmes distribués, il est nécessaire de donner des hypothèses restrictives; c'est ce que nous faisons ici.

Définition 4.3 *Une exécution bi-valuée* $(\mathcal{S}, \longrightarrow, \dashrightarrow, \mathcal{E}, \mathcal{L}, \varepsilon)$ *est séquentielle si toutes les lectures sont isolées et si* λ *est induite par l'application qui à toute lecture* L *associe la dernière écriture de* $\mathcal{E}_0(L)$.

Définition 4.4 *Une exécution bi-valuée est sûre, si sa restriction à l'ensemble des lectures isolées et de toutes les écritures est une exécution séquentielle.*

Ainsi dans une exécution sûre, une lecture non isolée peut obtenir un résultat quelconque, et une lecture isolée obtient comme résultat la dernière valeur écrite. On remarque que la valuation des lectures n'est pas nécessairement induite par une fonction de lecture. Une exécution bi-valuée dans laquelle aucune lecture n'est isolée est de manière évidente sûre.

Définition 4.5 *Une exécution bi-valuée est atomique, s'il existe une extension linéaire* $\prec$ *de* $\longrightarrow$, *telle que l'exécution formée par les mêmes événements et pour laquelle la relation de causalité forte est égale à* $\prec$, *est une exécution bi-valuée séquentielle.*

Ainsi une exécution bi-valuée $(\mathcal{S}, \longrightarrow, \dashrightarrow, \mathcal{E}, \mathcal{L}, \varepsilon)$ est atomique si λ est induite par une fonction de lecture π et s'il existe un ordre total $\prec$ tel que :

$$A \longrightarrow B \quad \Rightarrow \quad A \prec B$$

$$\pi(L) \prec L \quad \text{et} \quad \pi(L) \prec A \prec L \;\Rightarrow\; A \notin \mathcal{E}$$

La Figure 4 donne trois exemples d'exécutions bi-valuées. La première est séquentielle, car toutes les lectures et les écritures sont en relation par $\longrightarrow$ et que chaque lecture est valuée par la même valeur que la dernière écriture qui la précède. La seconde est sûre (mais ni séquentielle ni atomique), car les seules lectures isolées sont L_1 et L_5 la restriction de l'exécution à celles-ci et aux écritures donne

$$E_0 \longrightarrow L_1 \longrightarrow E_1 \longrightarrow E_2 \longrightarrow E_3 \longrightarrow E_4 \longrightarrow L_5$$

et l'on a bien $\lambda(L_1) = \varepsilon(E_0)$, $\lambda(L_5) = \varepsilon(E_4)$. La troisième est atomique, car on peut trouver une extension linéaire $\prec$ de $\longrightarrow$ sur l'ensemble de toutes les actions donnée par :

$$E_0 \prec L_1 \prec L_2 \prec E_1 \prec E_2 \prec L_3 \prec E_3 \prec L_4 \prec E_4 \prec L_5$$

extension qui possède les propriétés requises.

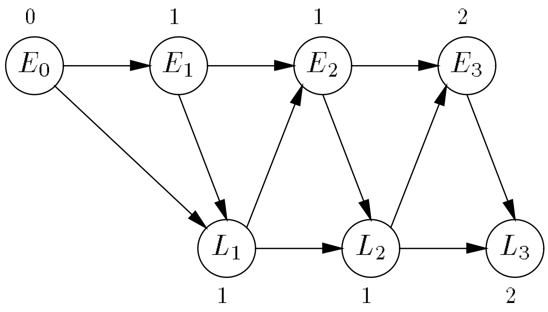
On caractérise les exécutions atomiques par l'existence d'une fonction de lecture satisfaisant certaines conditions :

Proposition 4.6 *Une exécution de registre est atomique si et seulement si elle est induite par une fonction de lecture* π *telle que :*

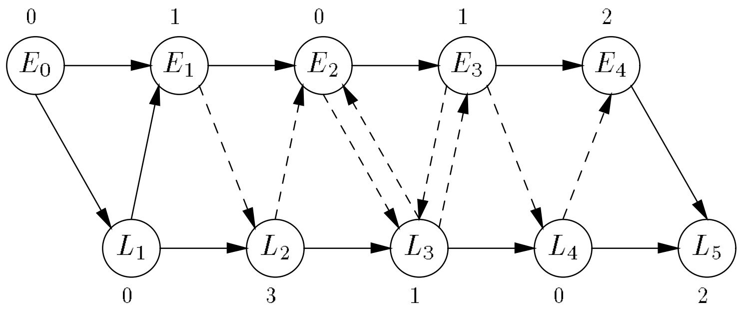
$$(i) \quad \forall L \in \mathcal{L} : \quad \neg(L \longrightarrow \pi(L))$$

$$(ii) \quad \forall E \in \mathcal{E}, \quad \forall L \in \mathcal{L} : \quad E \longrightarrow L \quad \Rightarrow \quad E \longrightarrow \pi(L) \text{ ou } \pi(L) = E$$

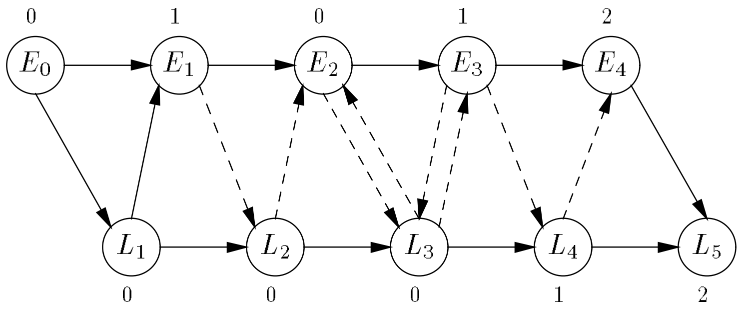
$$(iii) \quad \forall L_1, L_2 \in \mathcal{L} : \quad L_1 \longrightarrow L_2 \quad \Rightarrow \quad \neg(\pi(L_2) \longrightarrow \pi(L_1))$$



Exécution séquentielle



Exécution sûre



Exécution atomique

FIG. 4 - Exécutions séquentielles, sûres et atomiques, les valuations sont indiquées par un nombre situé à côté de chaque action

Preuve. Supposons l'exécution atomique, soit π la fonction de lecture associée et $\prec$ l'extension linéaire de $\longrightarrow$ satisfaisant les conditions données par la définition d'atomicité. Celles-ci impliquent $\pi(L) \prec L$, soit puisque $\prec$ est une extension linéaire de $\longrightarrow$

$$\neg(L \longrightarrow \pi(L))$$

Si $E \longrightarrow L$, alors $E \prec L$ et $\pi(L)$ est ou bien égale à E ou bien est située entre E et L dans l'ordre total $\prec$, soit puisque les écritures sont totalement ordonnées par $\longrightarrow$:

$$E \longrightarrow \pi(L) \,\text{ou}\; E = \pi(L)$$

Si $L_1 \longrightarrow L_2$, alors par l'ordre total $\prec$, $\pi(L_1)$ précède L_1 et donc L_2. Par l'atomicité de l'exécution $\pi(L_2)$ ne peut précéder $\pi(L_1)$ dans l'ordre $\prec$ qui se confond avec $\longrightarrow$ dans l'ensemble de écritures ainsi $\pi(L_1) = \pi(L_2)$ ou $\pi(L_1) \longrightarrow \pi(L_2)$ ce qui est équivalent à:

$$\neg(\pi(L_2) \longrightarrow \pi(L_1))$$

Réciproquement, supposons l'existence d'une fonction de lecture satisfaisant les conditions $(i), (ii)$ et (iii). Considérons un ordre total $\prec$ sur les événements qui soit une extension linéaire de $\longrightarrow$ pour les écritures et tel que, pour chaque écriture E, toutes les lectures L satisfaisant $\pi(L) = E$ sont placées juste après E en respectant la relation $\longrightarrow$ sur celles-ci, un tel ordre total existe puisque $\longrightarrow$ est une relation d'ordre partiel. Il suffit de montrer pour terminer que $\prec$ est bien une extension de $\longrightarrow$ sur les lectures et les écritures. Si $E \longrightarrow L$ on a d'après (ii) et puisque $\prec$ est une extension de $\longrightarrow$ sur les écritures : $E \preceq \pi(L) \prec L$. Si $L \longrightarrow E$ alors $\pi(L) \longrightarrow E$ en raison de (i) et donc $\pi(L) \prec L \prec E$. Enfin si $L_1 \longrightarrow L_2$, et $L_2 \prec L_1$ ceci impliquerait $\pi(L_2) \longrightarrow \pi(L_1)$ par définition de $\prec$ et ceci est contradictoire à la condition (iii). $\square$

De manière naturelle on définit la notion de registre sûr ou atomique :

Définition 4.7 *Un registre est sûr (resp. atomique) s'il associe à une exécution valuée une exécution bi-valuée sûre (resp. atomique).*

4.2 Réalisation de registres

4.2.1 Définition

La réalisation d'un objet partagé à l'aide d'autres objets est une notion fondamentale en algorithmique repartie. Cette notion est intuitivement simple, il s'avère pourtant difficile d'en donner une définition mathématique précise. Un traitement rigoureux nécessiterait de longs développements, nous nous limitons ici à une présentation informelle. Nous utilisons la notion d'implantation introduite au paragraphe 3.2.

Nous avons aussi besoin d'exécutions faisant appel à plusieurs registres et non pas un seul, comme cela a été le cas jusqu'ici. Une *exécution faisant intervenir k registres*, $k \geq 1$ est une exécution contenant $2k$ ensembles deux à deux disjoints : $\mathcal{E}_1, \ldots \mathcal{E}_k$ et $\mathcal{L}_1 \ldots \mathcal{L}_k$, tels que pour tout i, $1 \leq i \leq k$, le couple $\mathcal{E}_i$, $\mathcal{L}_i$ satisfasse les conditions d'une exécution d'un registre.

Définition 4.8 *La réalisation d'un registre X par des registres $X_1, \ldots X_k$ comporte deux opérations :*

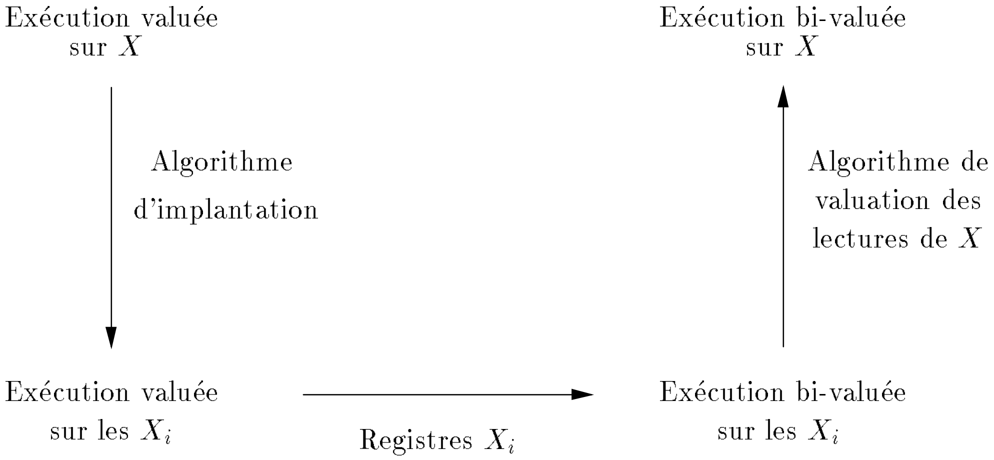


FIG. 5 - Schéma d'une réalisation

- *Un algorithme d'implantation des lectures et des écritures sur X en des opérations sur les X_i*
- *Un algorithme de calcul des valeurs obtenues par les lectures sur X.*

Le premier algorithme construit, à partir d'une exécution valuée $(\mathcal{S}, \longrightarrow, \dashrightarrow, \varepsilon)$ quelconque sur le registre X, une exécution valuée $(\mathcal{S}', \stackrel{'}{\longrightarrow}, \stackrel{'}{\dashrightarrow}, \varepsilon')$ sur des registres $X_1, \ldots X_k$, exécution qui est une implantation de la première au sens défini au paragraphe 3.2. chaque écriture et chaque lecture sur X est remplacée par un ensemble d'actions sur $X_1, \ldots X_k$. Les registres X_i agissent sur cette implantation pour fournir une valuation λ' des lectures de $\mathcal{L}'$. Le deuxième algorithme calcule la bi-valuation λ sur $(\mathcal{S}, \longrightarrow, \dashrightarrow)$ à partir de celle sur $(\mathcal{S}', \stackrel{'}{\longrightarrow}, \stackrel{'}{\dashrightarrow})$, la valeur d'une lecture L de $\mathcal{S}$ ne dépend que des valeurs associées aux opérations qui l'implantent dans $\mathcal{S}'$, et de la valeur de certaines variables locales. La Figure 5 schématise les différentes étapes d'une réalisation.

Dans la suite nous explicitons la réalisation de registres multi-lecteurs à partir de registres à un seul lecteur et de registres à k valeurs à partir de registres binaires. Ces réalisations fournissent ainsi des exemples qui illustrent les notions introduites ici.

4.2.2 Réalisation de registres à plusieurs lecteurs

Une exécution d'un registre est à *m lecteurs* si l'on peut partitionner $\mathcal{L}$ en m parties disjointes $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m$ telles que la restriction de $\longrightarrow$ à chacun des $\mathcal{L}_i$, i = 1, ..., m soit une relation d'ordre totale. On dit par la suite d'un élément de $\mathcal{L}_i$, qu'il s'agit d'une lecture *effectuée par le processus i*. Un registre est dit à m lecteurs, s'il construit une valuation des lectures pour toute exécution valuée à m lecteurs. On peut réaliser un registre à m lecteurs, à l'aide de m registres à un seul lecteur en utilisant les algorithmes suivants pour l'écrivain et les lecteurs.

- *Algorithme d'implantation:* A chaque écriture E de X, valuée par v, on fait correspondre une suite $\{E_1, E_2 \cdots, E_m\}$ d'écritures, totalement ordonnées par $\stackrel{'}{\longrightarrow}$, où E_i représente l'écriture de la valeur v dans le registre X_i.

Pour toute écriture E implantée par $\{E_1, E_2 \cdots, E_m\}$, la valuation de E_i est égale à celle de E et on note $\varepsilon_i(E_i) = \varepsilon(E)$.

Une lecture L de X par le processus i est implantée par une lecture L_i du registre X_i

- *Relations dans $\mathcal{S}$ et $\mathcal{S}'$* : On suppose que $\longrightarrow$ est égale à $\xrightarrow{*}{}'$ et que $\dashrightarrow$ est égale à $\overset{*}{\dashrightarrow}{}'$. Cette notation signifie que $U \longrightarrow V$ est équivalent à $u \longrightarrow {}'v$ pour tous les u et v qui font partie des implantations de U et V respectivement. De même $U \dashrightarrow V$ est équivalent à $u \longrightarrow {}'v$ pour au moins un u et un v qui font partie des implantations de U et V respectivement.

- *Algorithme de valuation des lectures :* Pour chaque $i = 1, \ldots, m$, notons $\mathcal{E}'_i$ l'ensemble des écritures E_i provenant de l'implantation de toutes les écritures E. On obtient ainsi une partition de l'exécution $\mathcal{S}'$ en m exécutions $\mathcal{S}_i, i = 1 \ldots m$, chacune d'entre elles est à un seul lecteur. Pour l'exécution $\mathcal{S}_i$, le registre X_i associe une valuation λ_i des lectures de $\mathcal{L}_i$ et la valeur $\lambda(L_i)$ obtenue par la lecture L implantée par L_i est égale à celle, $\lambda_i(L_i)$, attribuée à L_i par X_i.

Remarque : De fait, l'ordre relatif dans lequel s'effectuent les écritures $E_1, E_2 \cdots, E_m$ dans l'implantation n'a pas d'importance; toutefois pour chaque couple E, E' d'écritures sur X telles que $E \longrightarrow E'$, on doit avoir $E_i \xrightarrow{\prime} E'_j$, pour tous les i, j tels que $1 \leq i, j \leq m$.

Proposition 4.9 *Dans la réalisation ci-dessus, si les X_i, $i = 1 \cdots m$, sont des registres sûrs à un lecteur alors X est un registre sûr à m lecteurs.*

Preuve. Soit $\mathcal{S}$ une exécution sur X dans laquelle on a une partition des lectures en m chaînes $\mathcal{L}_i$. Soit L une lecture isolée de $\mathcal{S}$ élément de $\mathcal{L}_i$. La lecture L_i qui implante L dans $\mathcal{S}'$ est aussi isolée, en effet si L_i est concurrente à une écriture E_i alors, par l'algorithme d'écriture, E_i est contenue dans une écriture E de $\mathcal{S}$ qui est concurrente à L, contredisant le fait que L soit isolée. Comme le registre X_i est sûr et L_i isolée, $\lambda_i(L_i)$ est la valeur $\varepsilon_i(E_i)$ écrite par l'écriture E_i qui précède immédiatement L_i pour la relation $\xrightarrow{\prime}$, il est alors facile de vérifier que E, l'écriture de X qui contient E_i est bien l'écriture qui précède immédiatement L pour la relation $\longrightarrow$. □

Si les registres X_i sont atomiques, le registre X construit n'est pas nécessairement atomique comme le montre l'exemple de la Figure 6. Dans cet exemple, il y a deux écritures et deux lectures, E écrit la valeur a et E' la valeur $b \neq a$ dans un registre X, L est une lecture par le processus 1 et L' une autre par le processus 2. On a représenté sur cette figure l'implantation de ces 4 opérations et les relations qui lient les événements de cette implantation. En raison de l'atomicité des registres X_i, L' obtient la valeur écrite par E, et L celle écrite par E'; ce qui ne permet pas de construire un ordre linéaire sur $\{E, E', L, L'\}$ satisfaisant aux conditions d'atomicité de X et qui soit une extension linéaire de $\longrightarrow$.

Toutefois, si l'on suppose que les exécutions sur chacun des registres qui implantent sont toutes séquentielles, alors l'exécution implantée satisfait certaines propriétés qui entraînent les conditions (i) et (ii) de la Proposition 4.6 qui caractérise l'atomicité (mais pas la condition (iii)!).

Proposition 4.10 *Si dans la réalisation ci-dessus l'implantation d'une exécution $\mathcal{S}$ donnée est séquentielle sur chacun des registres X_i, alors il existe une fonction de lecture π sur $\mathcal{S}$ satisfaisant aux conditions suivantes :*

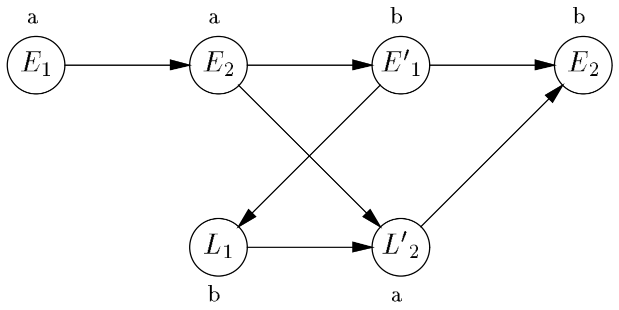


FIG. 6 - Exécution non atomique d'un registre multi-lecteurs

(1) $\pi(L) \dashrightarrow L$

(2) Pour toute écriture E telle que $\pi(L) \longrightarrow E$ on a $L \dashrightarrow E$.

Preuve.

(1) Soit L une lecture du registre X, L_i la lecture de X_i qui l'implante. La séquentialité de l'implantation implique que L_i est isolée vis à vis des écritures de $\mathcal{E}_i$. Soit E_i l'écriture de $\mathcal{E}_i$ qui précède immédiatement L_i, définissons $\pi(L)$ comme l'écriture de $\mathcal{E}$ dont l'implantation contient E_i. On a par l'algorithme de valuation des lectures :

$$\lambda(L) = \lambda_i(L_i) = \varepsilon_i(E_i) = \varepsilon(\pi(L))$$

De plus puisque $E_i \xrightarrow{\prime} L_i$ l'égalité de $\dashrightarrow$ et $\overset{*}{\dashrightarrow}{}'$ implique :

$$\pi(L) \dashrightarrow L$$

(2) Si $\pi(L) \longrightarrow E$, soit L_i l'implantation de L alors la séquentialité de l'exécution sur X_i implique :

$$\pi_i(L_i) \xrightarrow{\prime} L_i \xrightarrow{\prime} E_i$$

Ainsi par l'égalité de $\dashrightarrow$ et $\overset{*}{\dashrightarrow}{}'$ on obtient :

$$L \dashrightarrow E$$

Ce qui termine la preuve. $\square$

4.2.3 Réalisation de registres multivaleurs

Il est facile de réaliser un registre sûr à k valeurs à l'aide de registres sûrs binaires. La construction classique consiste à utiliser $p \geq log_2(k)$ registres binaires: $X_1, X_2, \ldots X_p$ et à représenter le nombre a par sa notation $a_1 a_2 \ldots a_p$ en base 2. L'écriture de a dans X est implantée par la suite des écritures des a_i dans X_i. La lecture de la valeur du registre s'effectue par la lecture consécutive de tous les X_i et la détermination de a à partir de son écriture en base 2. Il est assez facile de voir que le registre X est sûr si les X_i le sont, par contre si les X_i sont atomiques c'est loin d'être le cas de X. En effet, une lecture du registre concurrente au changement de valeur de a à b peut très bien conduire à récupérer les premiers digits de a et les derniers de b, la valeur lue est alors une valeur quelconque qui n'a rien à voir avec a et b.

Une autre construction possible pour représenter un entier a, compris entre 0 et k, consiste à utiliser $k+1$ digits binaires $x_0, x_1, \ldots, x_k$ tels que $x_a = 1$ et $x_i = 0$ pour $i \neq a$. Dans le contexte des registres, on utilise alors $k+1$ registres binaires $X_0, X_1, \ldots, X_k$ pour représenter un nombre compris entre 0 et k. La lecture du registre consiste à lire successivement les registres $X_0, X_1, \ldots X_k$, la valeur du registre est alors i si la première valeur obtenue égale à 1 se trouve dans le registre X_i. L'écriture du nombre a dans le registre X consiste, comme dans plusieurs algorithmes répartis, à procéder en sens inverse et à commencer par écrire 1 dans le registre a puis 0 dans les registres $X_{a-1}, X_{a-2}, \ldots, X_1, X_0$. On suppose que le registre X_k est initialisé à 1, il contient alors cette valeur indéfiniment. De façon plus précise on a :

- *Algorithme d'implantation:* Une écriture valuée par i est implantée par la suite d'actions :

  $$E_i \stackrel{\prime}{\longrightarrow} E_{i-1} \stackrel{\prime}{\longrightarrow} \cdots E_1 \stackrel{\prime}{\longrightarrow} E_0$$

  Où E_i est une écriture dans le registre X_i qui est valuée par 1, et pour $0 \leq j < i$, E_j est une écriture, valuée par 0, dans le registre X_j (il n'y a pas d'écriture de la valeur 0 si $i = 0$).

  Une lecture L de X est implantée par la suite

  $$L_0 \stackrel{\prime}{\longrightarrow} L_1 \stackrel{\prime}{\longrightarrow} \cdots \stackrel{\prime}{\longrightarrow} L_k$$

  où L_j, pour $0 \leq j \leq i$, est une lecture du registre X_j.

- *Relations dans $\mathcal{S}$ et $\mathcal{S}'$* : On suppose que $\longrightarrow$ est égale à $\stackrel{*}{\longrightarrow}{}'$ et que $\dashrightarrow$ est égale à $\stackrel{*}{\dashrightarrow}{}'$

- *Algorithme de valuation des lectures :* Chaque registre X_i construit une valuation λ_i des lectures L_i. La valeur $\lambda(L)$ associée par X à la lecture L s'obtient à partir des valeurs de $\lambda_j(L_j)$ de son implantation :

  $$L_0 \stackrel{\prime}{\longrightarrow} L_1 \stackrel{\prime}{\longrightarrow} \ldots L_{i-1} \stackrel{\prime}{\longrightarrow} L_k$$

  $\lambda(L)$ est égale à i, le plus petit entier tel que $\lambda_i(L_i) = 1$.

Avec cette nouvelle construction, il est clair que toute valeur obtenue lors d'une lecture est égale à la valuation de l'une des écritures (ce qui n'était pas le cas dans la réalisation faisant intervenir la représentation en base 2) on a aussi :

Proposition 4.11 *Si les registres $X_0, \ldots X_k$ sont sûrs, le registre construit par l'algorithme précédent est également sûr.*

Preuve. Soit L une lecture isolée de l'exécution $\mathcal{S}$, et soit E l'écriture de $\mathcal{S}$ qui précède immédiatement L. Notons $i = \varepsilon(E)$, on a les implantations suivantes de E et de L :

$$E_i \stackrel{\prime}{\longrightarrow} E_{i-1} \ldots \stackrel{\prime}{\longrightarrow} E_1 \stackrel{\prime}{\longrightarrow} E_0 \qquad L_0 \stackrel{\prime}{\longrightarrow} L_1 \ldots \stackrel{\prime}{\longrightarrow} L_{k-1} \stackrel{\prime}{\longrightarrow} L_k$$

Elles satisfont $E_j \overset{\prime}{\longrightarrow} L_j$ pour $0 \leq j \leq i$. Par l'égalité de $\longrightarrow$ et $\overset{*}{\longrightarrow}{}'$, pour tout $0 \leq j \leq k$ la lecture L_j est isolée; de plus puisque X_j est sûr :

$$\lambda_j(L_j) = \varepsilon_j(E_j)$$

ainsi l'algorithme de calcul de la valuation de L donne $\lambda(L) = i$, qui est bien la valuation de l'écriture qui précède immédiatement L. $\square$

Proposition 4.12 *Si dans la réalisation ci-dessus, l'implantation d'une exécution $\mathcal{S}$ est séquentielle sur chacun des registres X_i, il existe alors une fonction de lecture π sur $\mathcal{S}$ satisfaisant aux conditions suivantes :*

(1) $\pi(L) \dashrightarrow L$

(2) *Pour toute écriture E satisfaisant $\pi(L) \longrightarrow E$ on a $L \dashrightarrow E$*

Preuve. *(1)* Soit L une lecture implantée par :

$$L_0 \longrightarrow {}' L_1 \longrightarrow {}' \cdots \longrightarrow {}' L_k$$

soit i le plus petit entier tel que $\lambda'(L_i) = 1$, un tel entier existe car $\lambda'(L_k) = 1$ puisqu'aucune lecture n'écrit 0 dans le registre X_k qui est initialisé à 1. Le fait que les exécutions sur chacun des registres soient séquentielles implique que L_i est isolée et que l'écriture E_i, qui précède immédiatement L_i, est telle que $\varepsilon_i(E_i) = 1$. Notons $\pi(L)$ l'écriture de $\mathcal{S}$ qui contient E_i dans son implantation. On a par les algorithmes d'implantation et de calcul des valuations des lectures :

$$\varepsilon(\pi(L)) = i = \lambda(L)$$

De plus $E_i \longrightarrow {}' L_i$ implique

$$\pi(L) \dashrightarrow L$$

(2) Supposons que $\pi(L) \longrightarrow E$. Notons $E' = \pi(L)$, $i = \varepsilon(E')$ et $j = \varepsilon(E)$. Dans l'implantation on a :

$$E'_i \longrightarrow {}' E'_{i-1} \longrightarrow {}' \cdots E'_1 \longrightarrow {}' E'_0 \longrightarrow {}' E_j \longrightarrow {}' E_{j-1} \longrightarrow {}' \cdots E_1 \longrightarrow {}' E_0$$

Si $j \geq i$, il existe alors une écriture E_i. La séquentialité de l'implantation pour le registre X_i et le choix de $E' = \pi(L)$ impliquent :

$$E'_i \longrightarrow L_i \longrightarrow E_i$$

soit $L_i \dashrightarrow E_i$, et par l'égalité de $\dashrightarrow$ et $\overset{*}{\dashrightarrow}{}'$

$$L \dashrightarrow E$$

Si $j < i$, alors il n'existe pas d'écriture E_i, on utilise la séquentialité du registre X_j qui implique :

$$L_j \longrightarrow {}' E_j \quad \text{ou} \quad E_j \longrightarrow {}' L_j$$

dans le premier cas on obtient immédiatement $L \dashrightarrow E$, dans le second nous allons prouver que l'on aboutit à une contradiction. Comme $\lambda_j(L_j) = 0$ et que $\varepsilon_j(E_j) = 1$, c'est qu'il existe un E''_j, tel que $\varepsilon_j(E''_j) = 0$, situé entre E_j et L_j. L'écriture E''_j figure dans l'implantation d'une écriture

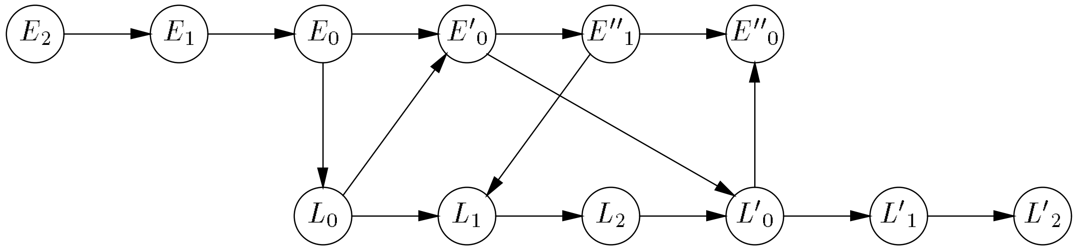


FIG. 7 - Exécution non atomique d'un registre à k valeurs

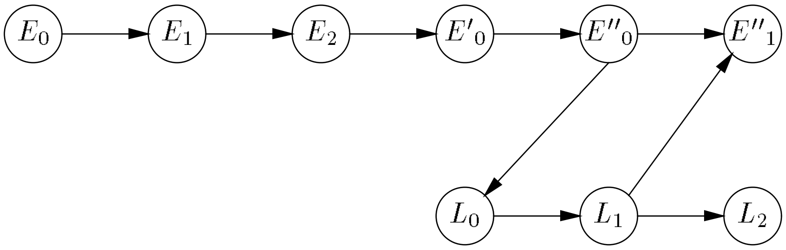


FIG. 8 - Implantation incohérente d'un registre à m valeurs

E'', vérifiant $E' \longrightarrow E'' \longrightarrow E$ et nécessairement telle que $\varepsilon(E'') > j$; notons $h = \varepsilon(E'')$. Si $h \geq i$ on aurait :

$$E'_i \longrightarrow {}'E''_i \longrightarrow {}'L_i$$

qui contredit la définition de $E' = \pi(L)$. Donc $h < i$, et on peut reprendre le raisonnement que l'on vient de faire en remplaçant E par E'' et j par h. En répétant cette opération suffisamment de fois on construit une suite de lectures qui suivent E' et précèdent L_j dont les valuations vont en croissant strictement; une de celles-ci a une valuation supérieure ou égale à i, ce qui contredit la définition de $E' = \pi(L)$. $\square$

Remarques.
1. Le registre ainsi construit n'est pas atomique, comme on peut le voir sur l'exemple représenté Figure 7. Sur cet exemple il y a 3 écritures $E \longrightarrow E' \longrightarrow E''$, valuées respectivement $2, 0, 1$ et deux lectures $L \longrightarrow L'$. On a représenté leurs implantations, l'algorithme de valuation des lectures donne $\lambda(L) = 1$ et $\lambda(L') = 2$, il est donc impossible de linéariser de façon cohérente $\{E, E', E'', L, L'\}$.

2. Si l'on avait choisit d'implanter les écritures dans l'ordre des indices de registres croissants on n'aurait pas obtenu un algorithme cohérent comme le montre l'exemple représenté Figure 8. Il y a dans cet exemple 3 écritures $E \longrightarrow E' \longrightarrow E''$ valuées respectivement $2, 0, 1$ et la lecture L qui bien que postérieure à E' est valuée par $2 = \varepsilon(E)$.

4.3 Registres réguliers

Dans le paragraphe précédent les deux réalisations de registres à plusieurs lecteurs et à plusieurs valeurs ont conduit à la construction de registres satisfaisant certaines propriétés donnés dans les Propositions 4.10 et 4.12, suivant L. Lamport nous appelons ces registres des registres *réguliers*

Définition 4.13 *Une exécution bi-valuée est régulière si elle admet une fonction de lecture* π *satisfaisant aux conditions suivantes pour toute lecture* L *:*

(i) $\pi(L) \dashrightarrow L$

(ii) Pour toute écriture satisfaisant $\pi(L) \longrightarrow E$ *on a* $L \dashrightarrow E$

Un registre régulier est un registre qui à toute exécution valuée associe une exécution régulière.

Remarques

1. La condition *(2)* implique aussi

$$\text{Si } E \longrightarrow L \quad \text{alors} \quad E = \pi(L) \text{ ou } E \longrightarrow \pi(L)$$

en effet, si $E \longrightarrow L$ on ne peut avoir $\pi(L) \longrightarrow E$ car la condition *(2)* est contradictoire à l'axiome *A2*, le résultat découle alors de ce que les écritures sont totalement ordonnées.

2. Un registre régulier satisfait aux conditions (i) et (ii) de la Proposition 4.6.

3. Dans une exécution de registre régulier on peut très bien avoir des lectures $L \longrightarrow L'$ qui sont concurrentes à des écritures $E \longrightarrow E'$ telles que L est valuée par $\varepsilon(E')$ et L' est valuée par $\varepsilon(E)$, on dit que l'on a une *inversion* des lectures. Une telle inversion est impossible dans une exécution de registre atomique. Toutefois cette possibilité d'inversion dans les registres réguliers se limite aux chaînes d'écritures de longueur 2 car si on a une chaîne d'écritures de longueur 3:

$$E \longrightarrow E' \longrightarrow E'' \text{ et } L \longrightarrow L'$$

et si $\pi(L) = E''$, alors $E'' \dashrightarrow L$ implique par l'axiome *(A4)* $E' \longrightarrow L'$, et on ne peut alors avoir $\pi(L') = E$.

Les propositions suivantes donnent des propriétés des registres réguliers:

Proposition 4.14 *Dans l'hypothèse du temps global, toute exécution de registre atomique est régulière.*

Preuve. En effet, la relation $\prec$ construite sur une exécution est une extension de $\longrightarrow$. Pour une lecture L, $\pi(L)$ est l'écriture qui précède immédiatement L pour $\prec$. Elle vérifie donc $\neg(L \longrightarrow \pi(L))$, et en raison de l'axiome de temps global

$$\pi(L) \dashrightarrow L$$

Si $\pi(L) \longrightarrow E$ alors l'atomicité implique

$$\pi(L) \prec L \prec E$$

soit

$$\neg(E \longrightarrow L)$$

et on obtient aussi en raison de l'axiome de temps global :

$$L \dashrightarrow E$$

Ainsi l'exécution est régulière. $\Box$

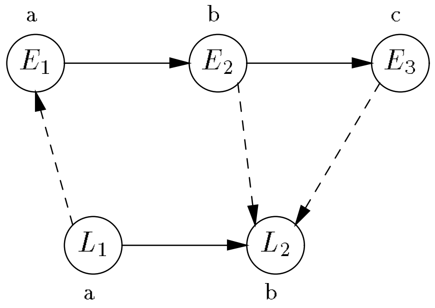


FIG. 9 - Exécution atomique mais non régulière

Remarque
Si on ne suppose pas le temps global une exécution atomique peut ne pas être régulière, les situations des deux lectures L_1 et L_2 de l'exécution présentée Figure 9 ne respectent pas les conditions de régularité puisque l'on a $\pi(L_1) = E_1$ et $\neg(E_1 \dashrightarrow L_1)$ de plus $\pi(L_2) = E_2$, et on a $E_2 \longrightarrow E_3$ mais pas $L_2 \dashrightarrow E_3$.

Proposition 4.15 *Les réalisations de registres multilecteurs et multivaleurs, donnéesau paragraphe 4.2, permettent de construire des registres réguliers, lorsque les registres qui les réalisent le sont.*

Nous ne prouvons pas ici cette proposition, pour le faire il suffirait de vérifier que, dans les preuves des deux Propositions 4.10 et 4.12, du paragraphe précédent, les hypothèses de séquentialité, qui ont été utilisées pour les registres X_i, peuvent être remplacées par des hypothèses de régularité sans que le résultat en soit affecté.

Avec l'hypothèse du temps global, on peut réaliser un registre régulier binaire X à l'aide d'un registre sûr binaire Y. On utilise pour cela une variable locale au processus écrivain, notée q dans la suite, qui permet de ne pas effectuer d'écriture dans le cas où celle-ci ne modifie pas la valeur du registre.

- *Algorithme d'implantation :* Une écriture de la valeur $b = 0, 1$ dans le registre X est implantée par un test sur la variable locale q. Si celle-ci est égale à b le processus ne fait pas d'écriture sur Y. Si $q \neq b$, il y a alors écriture de b dans Y et il faut dans ce cas mettre à jour la variable locale q en lui affectant la valeur b.

  Une lecture L de X est implantée par une lecture de Y.

- *Relations dans $\mathcal{S}$ et $\mathcal{S}'$* : On suppose que $\longrightarrow$ est égale à $\longrightarrow'$ et que $\dashrightarrow$ est égale à $\dashrightarrow'$

- *Algorithme de valuation des lectures :* La valeur obtenue par la lecture L de X est celle obtenue par la lecture de Y.

Proposition 4.16 *Dans l'hypothèse du temps global, si le registre Y est sûr, le registre X réalisé est régulier.*

Preuve. *(informelle)*

On remarque que $\mathcal{S}$ et $\mathcal{S}'$ sont pratiquement identiques, les écritures sur X de $\mathcal{S}$ qui ne modifient pas la valeur du registre sont transformées en des actions internes, il est d'autre part ajouté une action interne aux écritures qui modifient la valeur de la variable. Ainsi les lectures de $\mathcal{S}$ qui étaient isolées dans $\mathcal{S}$ restent isolées dans $\mathcal{S}'$, celles qui sont concurrentes à une (ou des) écriture(s) qui ne modifient pas la valeur du registre deviennent isolées dans $\mathcal{S}'$. Ainsi une lecture de X obtient la valeur de l'écriture qui la précède immédiatement sauf si elle est concurrente à une écriture qui modifie la valeur du registre. Mais dans ce dernier cas, comme le registre est binaire, le résultat obtenu est 0 ou 1 sans que cela n'affecte le caractère régulier de l'exécution. $\square$

Remarque Si on ne suppose pas l'hypothèse du temps global on peut très bien avoir une exécution de registre sûr pour laquelle on a $L \dashrightarrow E$ et $\pi(L) = E$, l'implantation ne modifie pas les relations $\dashrightarrow$. Ainsi elle ne peut être régulière car on n'a pas $\pi(L) \dashrightarrow L$.

5 Impossibilité de certaines réalisations

Dans cette section on montre qu'il n'est pas possible de réaliser un registre atomique à l'aide de registres réguliers, si l'on ne dispose pas d'un registre lu par l'écrivain et écrit par le lecteur et si le nombre de valeurs possibles des registres réguliers est fini. Une telle preuve d'impossibilité exige d'être précis sur le modèle utilisé pour la réalisation que l'on se propose de mettre en place; nous commençons donc par préciser à nouveau ce modèle.

L'écrivain, possède une variable interne qui peut prendre un nombre fini de valeurs, il écrit dans un registre atomique X des valeurs appartenant à un ensemble fini. L'écriture d'une valeur dans X est implantée par une suite d'actions : changement d'état et écritures successives dans des registres réguliers. Chacun des registres réguliers est susceptible de prendre un nombre fini de valeurs. De fait, on peut supposer que la réalisation est effectuée par un seul registre régulier Y en raison du lemme suivant :

Lemme 5.1 *Tout système d'exécution à écritures totalement ordonnées par* $\longrightarrow$ *, et comprenant plusieurs registres réguliers peut être implanté par un système ne comportant qu'un seul registre régulier.*

Preuve. Soit une exécution sur k registres $X_1, X_2, \ldots X_k$ à valeurs respectivement dans $V_1, V_2, \ldots V_k$, on considère le registre X à valeurs dans $V_1 \times V_2 \ldots \times V_k$. Une écriture E_i dans X_i est remplacée par une écriture dans X. La valuation de cette écriture est

$$(x_1, x_2, \ldots x_{i-1}, v_i, x_{i+1} \ldots x_k)$$

si la valuation de E_i est v_i et si les registres autres que X_i contiennent les valeurs x_j au moment de l'écriture E_i. Ces valeurs sont déterminées sans ambiguïté puisque les écritures sont totalement ordonnées par $\longrightarrow$; elle peuvent aussi être contenues dans une variable locale à l'écrivain qui est tenue à jour à chaque écriture.
Une lecture du registre X_i est remplacée par une lecture de X; pour valuer ces lectures, on ne retient que la i^{me} composante de la valeur lue dans X. On remarque qu'une lecture isolée en tant que lecture de X_i peut être concurrente à d'autres écritures et donc n'être plus isolée dans le deuxième système. Toutefois l'hypothèse de régularité de X joue un rôle important puisqu'elle implique que les i^{me} composantes des valuations possibles sont toutes égales. $\square$

Remarque.
Ce lemme n'est pas vrai si l'on suppose que les registres sont simplement sûrs, les arguments invoqués à la fin de la preuve ne pouvant plus être utilisés.

De ce lemme on déduit que s'il existe une réalisation d'un registre atomique, à l'aide de plusieurs registres réguliers dans lesquels n'écrit qu'un seul processus (appelé l'écrivain), alors on peut construire une réalisation à l'aide d'un seul registre régulier. Il suffit pour cela de composer la première avec l'implantation dont l'existence est établie par le lemme. On peut donc limiter la preuve d'impossibilité au cas où la réalisation ne contient qu'un seul registre régulier Y.

Poursuivons la description de la réalisation en décrivant l'implantation des lectures. Le processus qui effectue les lectures (appelé lecteur dans la suite) possède une variable interne, et une lecture de X se traduit par une succession de lectures du registre Y. La valeur obtenue pour X est décidée en fonction des valeurs obtenues par les lectures de Y et de celle de sa variable interne. Chaque écriture d'une valeur dans le registre atomique X se traduit donc par plusieurs écritures consécutives dans le registre régulier Y et chaque lecture de X est réalisée par une succession de lectures de Y.

Afin d'illustrer ce modèle de réalisation, montrons comment on peut réaliser un registre atomique à l'aide d'un registre régulier prenant un nombre infini de valeurs.

Soit V l'ensemble des valeurs possibles pour le registre atomique X, on choisit $V \times \mathbf{N}$ comme ensemble des valeurs de Y, où $\mathbf{N}$ est l'ensemble des entiers naturels. L'écrivain possède une variable interne s qui prend ses valeurs dans $\mathbf{N}$, et le lecteur une variable interne t qui prend ses valeurs dans $V \times \mathbf{N}$. L'écriture de a dans X est implantée par l'incrémentation de la variable locale s de l'écrivain ($s = s + 1$) et par l'écriture de (a, s) dans Y. Une lecture de X est implantée par la lecture de Y qui donne (a, u) et sa comparaison avec la variable interne $t = (b, v)$ du lecteur. Si $u < v$ la valuation de la lecture est alors b sinon c'est a et il faut dans ce second cas mettre à jour t par $t = (a, u)$.

C'est donc bien l'hypothèse de finitude du nombre de valeurs prises par le registre régulier qui nous permettra de prouver l'impossibilité de l'implantation. Pour cela nous utilisons un lemme combinatoire, très simple, sur les suites finies :

Lemme 5.2 *Pour toute suite finie $u = a_1, a_2, \ldots, a_n$ il existe une suite $v = b_1, b_2, \ldots, b_m$ telle que :*
- *(1)* $b_1 = a_1, \quad b_n = a_m$
- *(2)* $b_i = b_j \Rightarrow i = j$
- *(3)* *Pour tout $1 \leq i < m$, il existe $1 \leq j < n$ tel que $b_i = a_j$ et $b_{i+1} = a_{j+1}$*

Dans ce qui suit on appelle une telle suite v, *suite réduite* de u. Informellement les conditions (1), (2), (3), s'explicitent en : u et v ont le même premier élément et le même dernier élément, v ne contient pas deux fois le même élément et tout couple b_i, b_{i+1} d'éléments consécutifs dans v se retrouve comme un couple de deux éléments consécutifs de u.

Preuve. On procède par récurrence sur n. Si $n = 1$ alors $v = a_1$ répond à la question. Soit $u = a_1, a_2, \ldots, a_n, a_{n+1}$ et soit $v = b_1, b_2, \ldots, b_m$ la suite réduite de $a_1, a_2, \ldots, a_n$. Si $\forall i = 1, \ldots m, b_i \neq a_{n+1}$ alors v, a_{n+1} est une suite réduite de u. Sinon il existe un unique i tel que $b_i = a_{n+1}$ et on vérifie que $b_1, b_2, \ldots, b_i$ est une suite réduite de u. $\square$

Nous sommes maintenant en mesure de prouver le :

Théorème 5.3 *On ne peut pas réaliser un registre atomique à un lecteur, et à au moins 2 valeurs, à l'aide de registres réguliers à un nombre fini de valeurs lus par un lecteur et écrits par un écrivain.*

Preuve. En raison du Lemme 5.1 on peut se limiter au cas où il n'y a qu'un seul registre régulier dans l'implantation. Appelons Y ce registre régulier qui réalise un registre X, nous allons montrer que X n'est pas atomique. Pour cela notons V l'ensemble des valeurs prises par Y et k le nombre d'éléments de V.

• Nous commençons par définir un comportement de l'écrivain nous examinerons plus loin des comportements possibles pour le lecteur. Considérons la suite d'écritures suivante sur le registre X

$$E_0 \longrightarrow E_1 \longrightarrow E_2 \longrightarrow \cdots \longrightarrow E_n$$

où les écritures E_i d'indice impair sont valuées par 1 et celles d'indice pair par 0. Pour chaque i, E_{2i+1} est implantée par une suite d'écritures dans Y que l'on note :

$$e_{i,1} \longrightarrow' e_{i,2} \longrightarrow' \cdots \longrightarrow' e_{i,p_i}$$

notons u_i la suite formée des valeurs contenues dans le registre Y lors de ces différentes écritures. Le premier élément de u_i est donc la valeur contenue dans Y avant l'écriture $e_{i,1}$, le second est la valeur écrite par $e_{i,1}$ et ainsi de suite jusqu'à la valeur écrite par e_{i,p_i}; soit v_i une suite réduite de u_i, dont l'existence est assurée par le Lemme 5.2.

• Puisque le nombre de v_i possibles est fini on peut choisir n suffisamment grand pour que k parmi les v_i soient égales. Notons

$$v = b_0, b_1, b_2, \ldots, b_p$$

la valeur commune à ces k occurrences identiques de v_i. On a ainsi obtenu une suite d'écritures sur Y dont les suites réduites se répètent k fois. On peut noter que l'on ne peut pas assurer l'existence d'une répétition effective sur les suites u_i elles mêmes, car on ne suppose pas que la suite d'écritures sur Y réalisant une écriture sur Y est bornée, ainsi les u_i peuvent être tous distincts.

• Examinons maintenant le comportement du lecteur. Celui-ci pour chaque lecture de X effectue une suite de lectures de Y. Une lecture L_i de X est ainsi implantée par les lectures suivantes sur Y :

$$l_{i,1} \longrightarrow' l_{i,2} \longrightarrow' \cdots l_{i,j} \cdots \longrightarrow' l_{i,q_i}$$

On peut assumer l'existence d'une exécution dans laquelle toutes ces lectures sont valuées par la même valeur soit :

$$\lambda_Y(l_{i,1}) = \lambda_Y(l_{i,2}) = \cdots = \lambda_Y(l_{i,q_i})$$

Pour simplifier on dit que le lecteur obtient la valeur b, au lieu de dire qu'il obtient plusieurs fois consécutivement b. Pour chaque L_i on note $\lambda_Y(L_i)$ la valuation commune à toutes les lectures $l_{i,j}$.

Soit deux lectures L et L' du registre X, successives (i. e. telles que $L \longrightarrow L'$ et qu'aucune lecture n'est située entre elles) satisfaisant :

$$\lambda_Y(L) = b_{i+1}, \lambda_Y(L') = b_i, \quad \text{et} \quad \lambda(L) = 1$$

alors on doit avoir, quelque soit l'état interne du lecteur lors de ses lectures :

$$\lambda(L') = 1$$

En effet ces deux lectures peuvent très bien avoir été concurrentes à une même écriture E_{2i+1}, L ayant été valuée par $1 = \varepsilon(E_{2i+1})$, L' qui la suit doit aussi être valuée 1, puisque le registre est atomique.

• Il existe un comportement du lecteur composé d'une suite de lectures successives

$$L_0 \longrightarrow L_1 \ldots L_i \longrightarrow \ldots L_{k-1} \longrightarrow L_k$$

telles que :

$$\lambda_Y(L_0) = b_k, \quad \lambda_Y(L_1) = b_{k-1}, \quad \ldots, \quad \lambda_Y(L_{k-1}) = b_1, \quad \lambda_Y(L_k) = b_0$$

En effet dans la mesure où il y a k écritures dont la suite réduite est $v = b_0, b_1 \cdots b_k$, il suffit de choisir chacune des L_i concurrente avec l'une de ces écritures.

• Dans ce comportement il est clair que l'on doit avoir $\lambda(L_0) = 1$, en effet L_0 peut suivre une écriture valuée par 1 sans être concurrente à aucune autre. En raison de ce qui a été remarqué plus haut on doit avoir aussi $\lambda(L_1) = 1$ et de proche en proche :

$$\lambda(L_2) = 1, \quad \ldots \lambda(L_{k-1}) = 1, \quad \lambda_Y(L_k) = 1$$

mais on est alors en contradiction avec le fait que X est atomique, puisque la dernière lecture L_k qui est telle que $\lambda_Y(L_k) = b_0$ peut très bien n'être en concurrence avec aucune écriture et se trouver succéder à une écriture valuée par 0. □

6 Constructions de registres atomiques

Nous nous intéressons ici à la réalisation de registres atomiques à partir de registres réguliers. Plus précisément, nous nous plaçons dans le cadre de registres mono-écrivain et mono-lecteur. Nous donnons une construction d'un registre multi-valeur atomique à partir d'un registre multi-valeur régulier et de deux registres binaires réguliers. Nous prouvons ensuite la validité de cette construction qui est la plus efficace possible dans le cas d'un registre binaire. Par contre, dans le cas de registres quelconques, il existe des constructions plus performantes dont nous évoquons les grandes lignes pour conclure cette section.

6.1 Principes généraux

On part pour la réalisation d'un registre atomique X à l'aide de registres réguliers d'une idée très simple : on utilise un registre régulier Y qui peut contenir le même type de valeurs que X, une opération de lecture ou d'écriture sur X est implantée par une ou plusieurs opérations de même nature sur Y. Comme le registre Y n'est pas atomique il faut ajouter d'autres opérations de communications, pour ces communications on utilise d'autres registres réguliers. Pour que l'exécution implantée soit atomique il faut construire une fonction de lecture π qui associe une écriture $\pi(L)$ à toute lecture L et qui satisfasse les trois conditions $(i), (ii), (iii)$ de la Proposition 4.6 , conditions que nous rappelons ici :

$$(i) \quad \forall L \in \mathcal{L} : \quad \neg(L \longrightarrow \pi(L))$$

$$(ii) \quad \forall E \in \mathcal{E}, \quad \forall L \in \mathcal{L} : \quad E \longrightarrow L \quad \Rightarrow \quad E \longrightarrow \pi(L) \text{ ou } \pi(L) = E$$

$$(iii) \quad \forall L_1, L_2 \in \mathcal{L} : \quad L_1 \longrightarrow L_2 \quad \Rightarrow \quad \neg(\pi(L_2) \longrightarrow \pi(L_1))$$

Il est assez facile de satisfaire aux conditions (i) et (ii), il suffit en effet, pour une lecture L de choisir comme valuation $\lambda_X(L)$ la valeur obtenue par l'une des lectures de Y qui implante L. La régularité de Y implique alors immédiatement (i) et (ii), toutefois il n'y a aucune raison pour que (iii) soit vérifiée. Dualement il est aussi assez facile de satisfaire (iii), il suffit pour cela d'utiliser une variable locale au lecteur (notée v dans la suite), qui contient la valeur de la première lecture de Y effectuée et de valuer systématiquement toutes les lectures de X par cette valeur. Si avec cet

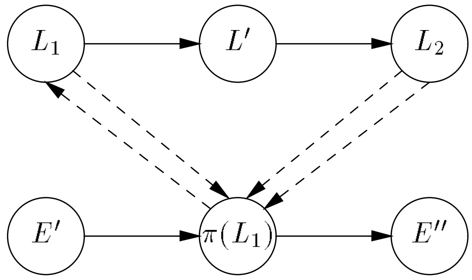


FIG. 10 - Une inversion entre deux lectures

algorithme on satisfait bien (iii), il n'y a aucune raison que (ii) soit satisfaite car les valuations ainsi données aux lectures sont trop anciennes et ne tiennent pas compte des modifications apportées au registre X par les écritures. L'algorithme de J. Tromp que nous décrivons ci-dessous a pour point de départ cette technique simple et se complète par deux constructions plus complexes. Une variable v, locale au lecteur, contient une valeur ancienne lue dans Y par celui-ci; lorsque le lecteur effectue une lecture L il lit plusieurs fois Y et détermine si l'une des valeurs lues dans Y est postérieure à la valuation de la lecture précédente, si c'est le cas il value L par celle-ci, sinon après s'être assuré qu'aucune écriture n'a eu lieu depuis la mise à jour de v, il value L par la valeur de v. Le problème est ainsi d'une part de déterminer avec certitude qu'une valeur de Y est postérieure à une autre et d'autre part qu'aucune écriture n'a eu lieu depuis la dernière lecture. Les deux lemmes suivants apportent une réponse à ces deux questions.

6.1.1 Lemme des lectures inversées

Considérons un registre ayant la valeur α et une écriture E faisant passer la valeur du registre à $\beta \neq \alpha$. Dans le cas d'un registre régulier, deux lectures L_1 et L_2 ordonnées ($L_1 \longrightarrow L_2$) en concurrence avec l'écriture E peuvent obtenir respectivement les valeurs β et α. Si l'on veut réaliser un registre atomique à partir de registres réguliers, une telle inversion doit être détectée par le lecteur. Si la lecture L_1 est valuée par la nouvelle valeur β, il faut donc que le lecteur, lors de la lecture L_2 s'en aperçoive et value cette lecture par la nouvelle valeur β. Pour avertir le lecteur qu'il a écrit une nouvelle valeur l'écrivain utilise un registre régulier que l'on appellera W dans la suite, il est écrit par l'écrivain et lu par le lecteur; la constatation par le lecteur d'une modification de la valeur de W va lui permettre d'ordonner correctement les lectures qu'il effectue. De manière plus précise on a :

Soit deux processus, un sera appelé écrivain et l'autre lecteur, qui communiquent à l'aide de 2 registres réguliers Y et W, le processus écrivain effectue des écritures soit dans Y soit dans W, le lecteur lit dans l'un ou l'autre des registres. Les fonctions de lectures associées à ces deux registres sont notées π et π_W respectivement. Puisque les lectures sur Y ou sur W sont effectuées par un même processus elles sont totalement ordonnées par $\longrightarrow$, il en est de même pour les écritures. On a alors :

Lemme 6.1 *Dans une exécution satisfaisant les conditions ci-dessus, soit deux lectures L_1 et L_2 du registre Y dont les valuations sont inversées, c'est à dire telles que :*

$$L_1 \longrightarrow L_2 \text{ et } \pi(L_2) \longrightarrow \pi(L_1)$$

Alors les lectures de W situées entre L_1 et L_2 sont isolées et leurs images par π_W sont toutes égales à l'écriture de W qui précède immédiatement $\pi(L_1)$.

Preuve. Comme le registre Y est régulier on a par la condition (i) :

$$\pi(L_1) \dashrightarrow L_1$$

En utilisant l'axiome (A_3) à partir de $L_1 \longrightarrow L_2$ on obtient :

$$\pi(L_1) \dashrightarrow L_2$$

En appliquant la condition (ii) dans la définition des registres réguliers on a puisque $\pi(L_2) \longrightarrow \pi(L_1)$:

$$L_2 \dashrightarrow \pi(L_1)$$

Soit en utilisant (A_3), puisque $L_1 \longrightarrow L_2$:

$$L_1 \dashrightarrow \pi(L_1)$$

Ainsi, si L' est une lecture du registre W située entre L_1 et L_2 on obtient une situation représentée sur la Figure 10. Alors pour toute écriture E' de W qui précède $\pi(L_1)$ et pour toute écriture E'' qui la suit on a :

$$E' \longrightarrow \pi(L_1) \dashrightarrow L_1 \longrightarrow L'$$

et

$$L' \longrightarrow L_2 \dashrightarrow \pi(L_1) \longrightarrow E''$$

Soit en appliquant l'axiome (A_4) :

$$E' \longrightarrow L' \longrightarrow E''$$

ainsi si E'_1 est l'écriture de W qui précède immédiatement $\pi(L_1)$ on a $\pi_W(L') = E'_1$. $\square$

Le Lemme peut être énoncé aussi sous la forme :

Corollaire 6.2 *Soit deux lectures L_1 et L_2 d'un registre régulier Y et soit L'_1, L'_2 deux autres lectures d'un autre registre W effectuées par le même processus sur un registre régulier W telles que*

$$L_1 \longrightarrow L'_1 \longrightarrow L'_2 \longrightarrow L_2$$

et

$$\varepsilon_W(L'_1) \neq \varepsilon_W(L'_2)$$

Alors les lectures L_1 et L_2 ne sont pas inversées.

Remarque
Ce corollaire permet de proposer un principe d'implantation d'un registre atomique X à l'aide de deux registres réguliers Y et W; une écriture de la valeur α dans X est implantée par l'écriture de la même valeur α dans Y suivie de plusieurs modifications du registre W. Le lecteur effectue des lectures de Y séparées par plusieurs lectures de W, s'il obtient des valeurs différentes de W entre deux lectures de Y, il peut en déduire qu'il n'y a pas eu d'inversions pour les valeurs obtenues par ces lectures de Y. Si elle permet d'éviter les inversions, cette technique ne permet pas de réaliser un registre atomique car une suite de lectures de W n'est pas assurée de constater un changement des valeurs lues. En effet, plusieurs modifications de W peuvent ne pas être perçues par le lecteur parce que W revient sur une valeur déjà prise, ou parce que l'écrivain n'a pas modifié W "assez vite". Pour palier cet inconvénient on utilise deux registres binaires R et W qui permettent au lecteur de s'assurer que l'écrivain n'a pas effectué d'écriture depuis la dernière lecture.

6.1.2 Lectures et écritures opposées

Dans ce qui suit l'écrivain et le lecteur communiquent à l'aide des registres W et R. Le registre R est utilisé par le lecteur pour montrer à l'écrivain que la précédente valeur du registre a été rendue comme résultat par (au moins) une lecture. Au contraire, l'écrivain utilise le registre W pour indiquer au lecteur qu'une nouvelle écriture a été effectuée. Plus précisément, l'écrivain fera en sorte que les valeurs de W et R soient différentes alors que le lecteur essaiera de les rendre égales. De façon plus précise, soit une exécution comportant deux registres R et W, et concernant deux processus : le lecteur et l'écrivain. L'écrivain écrit dans W et lit dans R en exécutant l'algorithme suivant qui tend à rendre $W \neq R$:

E^a:		Lire (R)
E^b:		Ecrire (W)
$\varepsilon_W(E^b)$	$=$	$1 - \lambda_R(E^a)$

Le lecteur quant à lui, effectue des lectures de W et des écritures dans R de façon quelconque. On a alors :

Lemme 6.3 *Avec les hypothèses ci-dessus, soit $E^a \longrightarrow E^b$ une action de l'écrivain. Soit L une lecture de W (effectuée par le lecteur) telle que $E^b \longrightarrow L$, qui constate $\lambda_W(L) = r$, où r est le contenu de R au moment [4] de L. Alors il existe une écriture F de R (effectuée par le lecteur) telle que :*

$$E^a \dashrightarrow F \longrightarrow L$$

Preuve. L'écriture $E'^b = \pi_W(L)$ associée à L par la fonction de lecture satisfait, puisque le registre W est régulier, les conditions suivantes :

$$E'^b \dashrightarrow L \text{ et } (E^b = E'^b \text{ ou } E^b \longrightarrow E'^b)$$

Ainsi $E = E'$ ou $E \longrightarrow E'$.
Notons $F' = \pi_R(E'^a)$, le fait que le registre R soit régulier implique par (i) :

$$F' \dashrightarrow E'^a$$

On ne peut avoir $L \longrightarrow F'$ car alors

$$E'^a \longrightarrow E'^b \dashrightarrow L \longrightarrow F'$$

implique $E'^a \longrightarrow F'$ par l'axiome (A_4), contredisant l'axiome (A_2). Ainsi les actions F' et L étant toutes les deux exécutées par le lecteur on a :

$$F' \longrightarrow L$$

Examinons maintenant les valeurs r et r' du registre R au moment des actions L et F' respectivement, on a :

$$r = \lambda_W(L) = \varepsilon(E'^b) = 1 - \lambda_R(E'^a)$$
$$r' = \varepsilon_R(F') = \lambda_R(E'^a) \quad \text{puisque} \quad \pi_R(E'^a) = F'$$

[4] Une telle notion a un sens car le registre R est écrit par le processus lecteur qui peut donc connaître sa valeur à un instant donné

Ainsi $r \neq r'$ et le registre R a été modifié entre F' et L, ceci ne peut l'être que par une écriture effectuée par le lecteur, écriture que nous notons F. Comme R est régulier, que $F' \longrightarrow F$ et que $F' = \pi_R(E'^a)$ on a :

$$E'^a \dashrightarrow F$$

En utilisant $E = E'$ ou $E \longrightarrow E'$ on obtient :

$$E^a \dashrightarrow F \longrightarrow L$$

□

De ce lemme on déduit le principe suivant pour un algorithme d'implantation d'un registre atomique. L'écrivain effectue la suite d'opérations :

E^a:	Ecrire (Y)
E^b:	Lire (R)
E^c:	Ecrire (W)

avec $\varepsilon_W(E^c) = 1 - \lambda_R(E^b)$

Le lecteur effectue la suite d'opérations

L^a:	Lire (W)
L^b:	Lire (Y)
L^c:	Ecrire (R)

avec $\varepsilon_R(L^c) = \lambda_W(L^a)$. Si le lecteur obtient comme valeur de W une valeur égale celle de r, il en déduit qu'aucune écriture n'a eu lieu depuis la dernière lecture qui a modifié R et il peut alors valuer la lecture en cours par l'ancienne valeur (contenue dans la variable locale v), par contre s'il constate $R \neq W$ il faut alors procéder à des opérations plus complexes.

6.2 Description de l'algorithme d'implantation

Nous donnons ici en détail la réalisation d'un registre atomique X à l'aide de registres réguliers. On utilise trois registres Y, W et R; les registres Y et W sont écrits par l'écrivain et lus par le lecteur, le registre R est écrit par le lecteur et lu par l'écrivain. Le registre Y contient les mêmes valeurs que le registre X à réaliser, R et W sont des registres binaires qui servent à la communication. On utilise les principes généraux évoqués ci-dessus, l'écrivain maintient $W \neq R$ et le lecteur cherche à rendre $W = R$. Le lecteur effectue des lectures alternées de Y et de W; dans le cas où le lecteur observe des variations de la valeur de W il peut considérer que les valeurs de Y qui encadrent ces variations sont correctement ordonnées. S'il n'observe pas de variation de W il doit alors considérer que la valeur du registre à implanter n'a pas été modifiée depuis la dernière lecture et il doit valuer sa lecture par la valeur précédente, une variable locale au lecteur, notée v lui sert à mémoriser la valeur précédemment lue. De façon plus précise on a :

Réalisation d'une écriture E de X :

E^a:	Ecrire (Y)	
E^b:	Lire (R)	
E^c:	Ecrire (W)	
$\varepsilon_Y(E^a) =$	$\varepsilon_X(E)$,	$\varepsilon_W(E^c) = 1 - \lambda_R(E^b)$

Réalisation d'une lecture L :

L^a:	Lire (W)
L^b:	Lire (Y)
L^c:	Lire (W)
L^d:	Ecrire (R)
L^e:	Lire (Y)
L^f:	Lire (W)
L^g:	Lire (Y)

Valuation de l'écriture L^d :
Notons r la valeur du registre R lors de l'exécution de L^a, on value L^d alors par :

$$\text{Si } \lambda_W(L^a) = \lambda_W(L^c) = 1 - r, \quad \text{alors : } \varepsilon_R(L^d) = 1 - r$$

$$\text{Si } \lambda_W(L^a) \neq 1 - r \text{ ou } = \lambda_W(L^c) \neq 1 - r, \quad \text{alors : } \lambda_R(L^d) = r$$

Intuitivement, on modifie R si et seulement si on constate deux fois de suite $R \neq W$.

Valuation de la lecture L :
On note ici r la valeur du registre R au moment de L^a et r' sa valeur au moment de L^f. Ainsi $r' = r$ si R n'a pas été modifié parL^d et $r' = 1 - r$ sinon. Une et une seule des trois conditions suivantes est alors vérifiée :

$$\begin{array}{ll} (A) & \lambda_W(L^a) = r \\ (B) & \lambda_W(L^a) \neq r \text{ et } \lambda_W(L^f) = r' \\ (C) & \lambda_W(L^a) \neq r \text{ et } \lambda_W(L^f) \neq r' \end{array}$$

et on donne la valeur suivante à $\lambda(L)$ suivant les cas :

$$\begin{array}{ll} \textbf{Cas } A: & \lambda(L) = v \\ \textbf{Cas } B: & \lambda(L) = \lambda_Y(L^e) \\ \textbf{Cas } C: & \lambda(L) = \lambda_Y(L^b) \end{array}$$

Mise à jour de la variable locale v :
La nouvelle valeur v' de la variable locale v après exécution de $L^a - L^g$ est donnée par :

$$\begin{array}{ll} \textbf{Cas } A: & v' = v \\ \textbf{Cas } B: & v' = \lambda_Y(L^e) \\ \textbf{Cas } C: & v' = \lambda_Y(L^g) \end{array}$$

Dans la suite on dit qu'une lecture L est de type A, B ou C suivant que la condition correspondante est ou non satisfaite.

L'algorithme d'implantation est schématisé Figure 11. Sur cette figure, un cercle représente une lecture de W, un rectangle une lecture de Y et une ellipse une écriture de R. Les traits | indiquent la position de la lecture de Y qui sert à valuer L ou à mettre à jour v. Au dessous des lectures de W on a dessiné un cercle vide si la lecture obtient $R = W$ et un cercle plein si celle-ci obtient $R \neq W$. L'ellipse est pleine si l'opération L^d modifie R, elle est vide sinon.

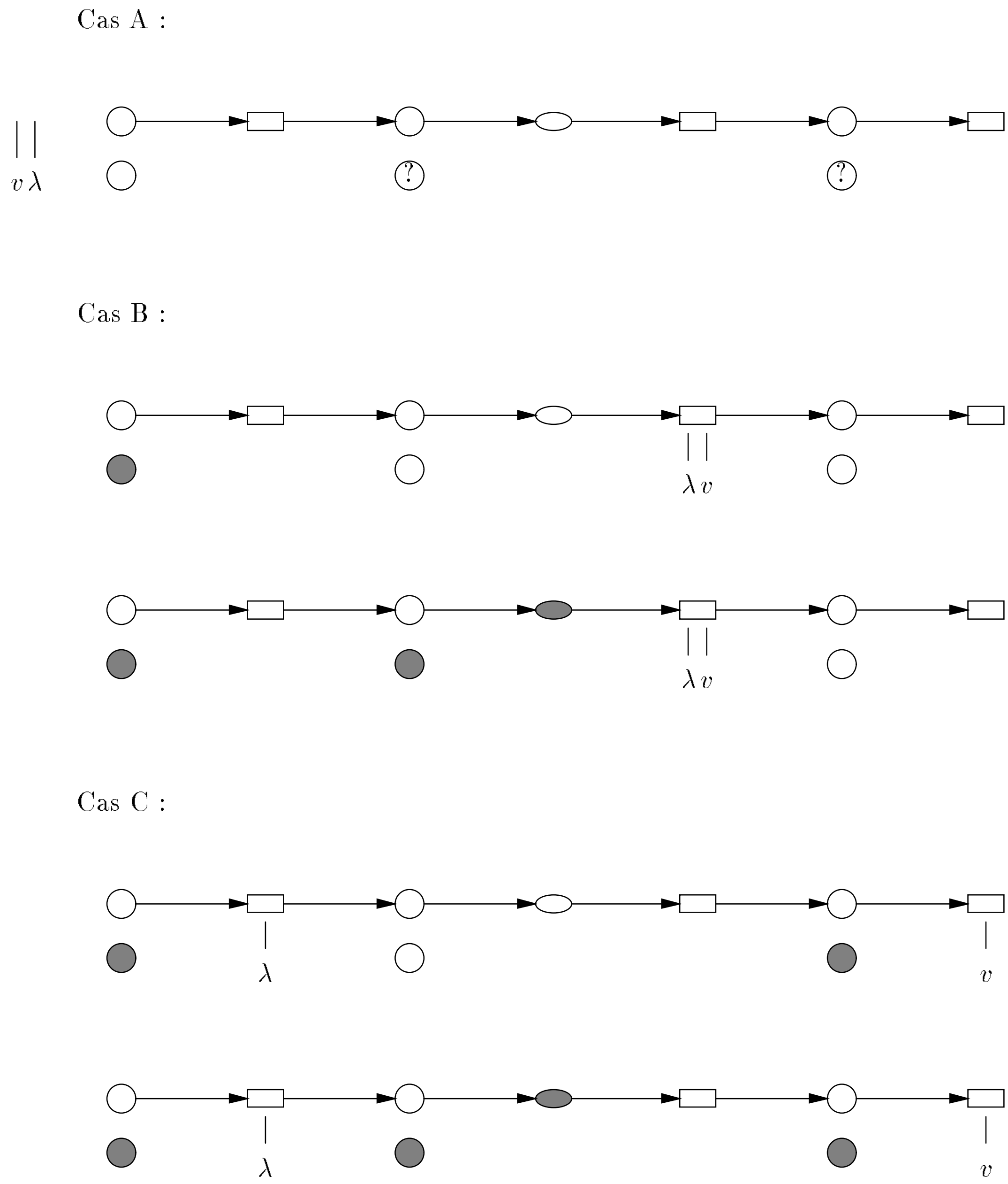
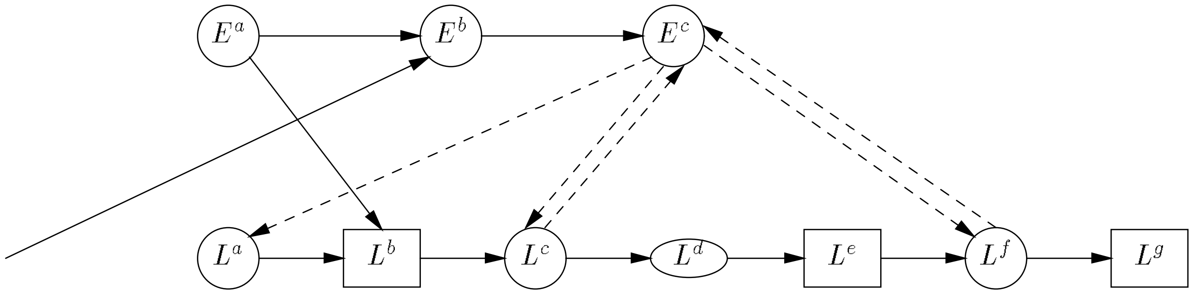


FIG. 11 - L'algorithme d'implantation d'une lecture



L_1 :	$\mid\,\mid$ $v\,\lambda$	0	β	–	0	β	–	β
L_2 :		1	β	0	0	$\mid\,\mid\beta$ $v\,\lambda$	0	β
L_3 :		1	$\mid\beta$ λ	0	0	β	1	$\mid\beta$ v
L_4 :		1	β	1	1	$\mid\,\mid\beta$ $v\,\lambda$	1	β
L_5 :		1	$\mid\beta$ λ	1	1	β	0	$\mid\beta$ v

FIG. 12 - Un exemple d'exécution

6.3 Exemple du déroulement de l'algorithme d'implantation

Nous allons ici "dérouler" l'exécution des protocoles de l'écrivain et du lecteur sur un exemple afin de montrer qu'une lecture peut effectivement être de type (A), (B) ou (C). Nous supposons que la variable locale v et les registres W et R ont tous initialement la valeur 0. L'écrivain exécute l'instruction E^a, qui fait passer la valeur de Y de α à β puis l'instruction E^b, il lit la valeur de R, soit 0, et effectue enfin l'action E^c qui fait passer la valeur du registre W de 0 à 1. Nous allons considérer cinq lectures L_1, L_2, L_3, L_4 et L_5 se déroulant après E^b et concurremment à l'action E^c (ces 5 lectures correspondent à 5 exécutions différentes). Le registre W étant régulier, toute lecture de W contenue dans L_1, L_2, L_3, L_4 ou L_5 peut être valuée arbitrairement par la valeur 0 ou 1. Par contre, toute lecture du registre Y contenue dans L_1, L_2, L_3, L_4 ou L_5 est valuée β. La Figure 12 ci-dessous décrit les cinq lectures L_1, L_2, L_3, L_4 et L_5. Les valeurs indiquées sous les actions sont celles obtenues lors de la lecture de W.

- La lecture L_1

  Elle obtient $W = 0$ lors de la lecture L_1^a, comme $r = \lambda_W(L_1^a)$ on peut terminer là l'ensemble des actions car les autres n'ont aucun effet sur la valuation ou sur le registre R. La lecture L_1 est de type (A).

- La lecture L_2

  Elle obtient $W = 1$ lors de L_2^a, l'instruction L_2^b est valuée β. Ensuite L_2^c obtient $W = 0$ et l'instruction L_2^d ne change pas la valeur de R. Puis L_2 obtient $W = 0$ lors de L_2^f, la lecture L_2 est de type (B) et $\lambda(L_2) = \lambda_Y(L_2^e)$.

- La lecture L_3

Elle se déroule comme la lecture L_2 jusqu'à l'instruction L_3^f qui obtient $W = 1$. La lecture est de type (C) et on a $\lambda(L_3) = \lambda_Y(L_3^b)$.

- La lecture L_4

  Elle obtient $W = 1$ lors de L_4^a. Ensuite L_4 obtient $W = 1$ lors de L_4^c. La valeur du registre R devient alors 1, enfin L_4 obtient $W = 1$ lors de L_4^f, la lecture est de type (B) et $\lambda(L_4) = \lambda_Y(L_4^e)$.

- La lecture L_5

  Elle se déroule comme L_4 sauf en L_5^f où elle obtient $W = 0$. La lecture est de type (C) et $\lambda(L_5) = \lambda_Y(L_5^b)$.

6.4 Preuve de la validité de l'implantation

Preuve. Considérons une exécution $\mathcal{S}$ comportant un ensemble de lectures $\mathcal{L}$ et un ensemble d'écritures $\mathcal{E}$ relatives au registre X et leur implantation par les algorithmes donnés ci-dessus. Nous allons définir une fonction de lecture, c'est à dire une application π de $\mathcal{L}$ dans $\mathcal{E}$ telle que, pour toute lecture L de $\mathcal{L}$, la valeur écrite par $\pi(L)$ est égale à la valuation $\lambda(L)$ de la lecture L.

Pour toute lecture L du registre X on note ρ_L la lecture du registre Y qui a donné sa valeur à $\lambda(L)$. Ainsi si L est de type (B) on a $\rho_L = L^e$, si L est de type (C) alors $\rho_L = L^b$ et si L est de type (A) alors on a $\rho_L = L'^e$ ou $\rho_L = L'^g$ où L' est la dernière lecture qui précède L et qui trouve $R \neq W$ lors de sa première lecture de W. La lecture ρ_L s'effectue sur le registre Y régulier on peut alors définir $\pi_Y(\rho_L)$ qui est nécessairement un E^a pour une certaine écriture E de X on définit alors :

$$\pi(L) = E$$

Nous montrons maintenant que la fonction de lecture π satisfait les 3 conditions $(i), (ii)$ et (iii) de la Proposition 4.6.

Condition (i) : On montre que $\pi(L) \dashrightarrow L$. Si L est de type (B) ou (C) on sait que ρ_L figure dans l'implantation de L. Or puisque le registre Y est régulier on a

$$\pi_Y(\rho_L) \dashrightarrow \rho_L$$

ce qui implique pour les opérations E et L qui contiennent respectivement ces actions dans leur implantation :

$$E \dashrightarrow L$$

Si L est de type (A), ρ_L ne figure pas dans l'implantation de L mais précède toutes les actions de cette implantation on a ainsi :

$$\rho_L \longrightarrow L^a$$

ce qui implique par l'axiome (A_3) :

$$\pi_Y(\rho_L) \dashrightarrow L^a$$

et le résultat.

Condition (ii) : On montre que si L est une lecture de X et E une écriture de X telle que $E \longrightarrow L$ alors on a :

$$\pi(L) = E \text{ ou } E \longrightarrow \pi(L)$$

Si ρ_L est un élément de l'implantation de L on a :

$$E^a \longrightarrow \rho_L$$

puisque le registre Y est régulier ceci implique

$$\pi_Y(\rho_L) = E^a \text{ ou } E^a \longrightarrow \pi_Y(\rho_L)$$

d'où le résultat. Si ρ_L n'est pas un élément de l'implantation de L alors on applique le Lemme 6.3. Puisque l'on a $E^c \longrightarrow L^a$ et que L^a trouve $R = W$ au moment de son action le lemme assure l'existence d'une écriture F de R par le lecteur telle que :

$$E^b \dashrightarrow F \longrightarrow L^a$$

Cette écriture F a modifié la valeur de R, ainsi il existe une lecture L' du registre X telle que $F = L'^d$ et L' est de type (B) ou (C). Ainsi ρ_L est ou bien dans l'implantation de L' ou bien dans celle d'une lecture qui a suivi L'. Dans tous les cas on a :

$$E^a \longrightarrow E^b \dashrightarrow L'^d \longrightarrow \rho_L$$

car ρ_L est égal à L'^e ou L'^g ou bien figure dans une lecture qui les suit. L'axiome (A_4) implique alors :

$$E^a \longrightarrow \rho_L$$

et puisque le registre Y est régulier on obtient :

$$\pi_Y(\rho_L) = E^a \text{ ou } E^a \longrightarrow \pi_Y(\rho_L)$$

ceci donne alors :

$$\pi(L) = E \text{ ou } E \longrightarrow \pi(L)$$

Condition (iii) : Il suffit de montrer que si $L_1 \longrightarrow L_2$ sont deux lectures successives alors on a :

$$\pi(L_1) = \pi(L_2) \text{ ou } \pi(L_1) \longrightarrow \pi(L_2)$$

pour montrer que le résultat est encore vrai si les deux lectures ne sont pas successives, il suffit de procéder par transitivité. De fait on peut se contenter de montrer que si $L_1 \longrightarrow L_2$ alors :

$$\rho_{L_1} = \rho_{L_2} \text{ ou } \rho_{L_1} \longrightarrow \rho_{L_2}$$

La fin de la preuve consiste à considérer tous les types de lectures possibles pour L_1 et L_2 et à utiliser le Lemme 6.1 dans chacun des cas. Si L_2 est de type (A) alors

- si L_1 est aussi de type (A) :$\rho_{L_1} = \rho_{L_2}$
- si L_1 est de type (B) alors on a aussi $\rho_{L_1} = \rho_{L_2}$
- si L_1 est de type (C) alors on a $\rho_{L_1} = L_1^b$ et $\rho_{L_2} = L_1^g$ d'où $\rho_{L_1} \longrightarrow \rho_{L_2}$ on remarque qu'entre ces deux lectures de Y il existe deux lectures de W à savoir L_1^c et L_1^f qui obtiennent des valeurs distinctes en appliquant le Lemme 6.1, il n'y a pas d'inversions.

Si L_2 est de type (B) ou (C) et si L_1 est de type (A), alors ρ_{L_1} précède L_1^a et ρ_{L_2} suit L_2^a, comme $\lambda_W(L_1^a) \neq \lambda_W(L_2^a)$ on peut appliquer le Lemme 6.1 et il n'y a pas d'inversion entre ρ_{L_1} etρ_{L_2}.

Si L_2 est de type (B) ou (C) et si L_1 est de type (B) on a :

$$\rho_{L_1} = L_1^e \text{ et } \rho_{L_2} = L_2^e \text{ ou } \rho_{L_2} = L_2^b$$

De toute façon entre ces deux lectures de Y se trouvent L_1^f et L_2^a, lectures de W telles que :

$$\lambda_W(L_1^f) = r \neq \lambda_W(L_2^a)$$

Une fois de plus le Lemme 6.1 permet de conclure.

Si L_2 est de type (B) ou (C) et si L_1 est de type (C), on a

$$\rho_{L_1} = L_1^b \text{ et } \rho_{L_2} = L_2^e \text{ ou } \rho_{L_2} = L_2^b$$

il y a deux lectures de W qui obtiennent des valeurs différentes entre L_1^b et L_1^g ce qui termine la preuve. $\square$

En conclusion, le registre construit à partir des registres sûrs Y, W et R vérifie les hypothèses de la Proposition 4.6 et est donc atomique. Nous avons ainsi donné une construction de registres atomiques à partir de registres réguliers. Par ailleurs, nous avons vu en Section 4 que des registres réguliers peuvent être construits à partir de registres sûrs. Ainsi nous pouvons énoncer :

Théorème 6.4 *Un registre multi-valeur atomique mono-écrivain et mono-lecteur peut être construit à partir de registres multi-valeurs sûrs.*

La question naturelle qui se pose alors est d'évaluer la complexité d'une telle construction. Dans le cas d'un registre binaire, nous avons vu au début de cette section qu'au moins trois registres booléens réguliers ou sûrs étaient nécessaires. Comme un registre binaire régulier peut être construit à partir d'un registre binaire sûr en changeant simplement l'opération de lecture, la construction proposée dans cette section montre que :

Théorème 6.5 *Il faut et il suffit de trois registres binaires sûrs pour construire un registre binaire atomique mono-écrivain et mono-lecteur.*

Dans le cas de registres multi-valeurs, le nombre de registres binaires sûrs nécessaires à la construction d'un registre atomique dépend du nombre de valeurs que peut prendre ce registre. Pour simplifier, supposons que le registre atomique que nous voulons construire puisse prendre 2^b valeurs pour un certain entier b. La construction donnée plus haut permet alors de construire un tel registre atomique à partir d'un registre régulier pouvant prendre 2^b valeurs et de deux registres binaires réguliers (ou sûrs). De plus, nous avons vu au paragraphe 4.2 qu'un registre régulier pouvant prendre 2^b valeurs pouvait être construit à partir de 2^b registres binaires réguliers (ou sûrs). Ainsi la construction ci-dessus nécessite $2^b + 2$ registres binaires réguliers (ou sûrs). Il est en fait possible d'améliorer considérablement cette complexité puisqu'un nombre linéaire en b de registres binaires est suffisant.

Il existe principalement deux méthodes pour de telles constructions. La première, due à Peterson [Pet] consiste à considérer trois copies du registre multi-valeur et un certain nombre de registres binaires regroupés sous le nom de contrôle. Chaque accès au registre multi-valeur écrit ou lit les trois copies. Le contrôle permet de savoir laquelle des trois lectures a été faite sans interférences avec une écriture. Ainsi, à une constante près, le nombre de registres binaires sûrs utilisés est $3b$ et chaque lecture ou écriture nécessite l'accès à également $3b$ registres binaires.

Dans la seconde méthode, quatre copies du registre multi-valeur sont utilisées. Par contre, chaque lecture ou écriture n'est faite que sur une seule des quatre copies. Le contrôle permet de déterminer laquelle des quatre copies doit être utilisée de façon à rendre atomique ces lectures et écritures. Ce principe est du à Kirousis et al [KKV] et a été repris par Tromp [Tro] qui a diminué la taille du contrôle.

Remerciements

Les auteurs souhaitent remercier chaleureusement Hugues Fauconnier, Béatrice Bérard qui ont participé activement au groupe de travail qui a précédé l'écriture de cet article et Shlomo Moran qui a grandement contribué à la redaction de la preuve d'impossibilité du paragraphe 5.

Bibliographie

[Abr] Abraham, U. *On interprocess communication and the problem of common atomic registers*, Technical Report Ben-gurion University, Israel, (1988).

[And] Anderson, J. H. *Composite registers*, Distributed Computing, **6**, 141-154 (1993).

[Ang] Anger, F. D. *On Lamport's Interprocessor Communication Model*, ACM Trans. on Programming Languages and Systems, **11**, 404-417 (1989).

[Arn] Arnold, A. *An example of use of MEC: verification of Tromp's algorithm* , Technical Report, Université Bordeaux 1, (1993).

[AKKV] Awerbuch, B., Kirousis, L. M., Kranakis, E., Vitányi P. *A proof technique for register atomicity*, Proceedings of Conference on Foundations of Software Technology and Theoretical Computer Science, LNCS, Springer Verlag **338**, 286-303 (1988).

[Blo] Bloom, B. *Constructing Two-writer atomic register*, IEEE Trans. on Computers, **37**, 1506-1514 (1988).

[HS] Haldar, S. , Subramanian, P. S. *Space-optimal conflict free construction of 1-writer 1-reader multivalued atomic register*, International 8^{th} Workshop on Distributed Computing, LNCS, Springer Verlag, **857**, 116-129 (1994).

[Her] Herlihy, M. *Wait free synchronization* ACM Trans. on Programming Languages and Systems, **13**, 124-149 (1991).

[HW] Herlihy, M. , Wing M. *Linearizability : a correctness condition for concurrent objects* ACM Trans. on Programming Languages and Systems, **12**, 463-492 (1990).

[IS] Israeli A. , Shaham A. *Optimal multi-writer atomic register* Proceedings of the 11^{th} ACM Symposium on Principles of Distributed Computing (1992).

[KKV] Kirousis, L. M., Kranakis E., Vitányi P. *Atomic Mulitreader Register*, International 2^{nd} Workshop on Distributed Computing, LNCS, Springer Verlag, **312**, 278-296 (1987).

[KV] Kirousis, L. M. , Veneris A. G. *Efficient algorithms for checking atomicity of a run of read and write operations*, International 7^{th} Workshop on Distributed Computing, LNCS, Springer Verlag, **725**, 54-68 (1993).

[L78] Lamport, L. *Time, Clocks, and the Orderings of Events in a Distributed System*, Communications of the ACM, **21**, 558-564 (1978).

[L85] Lamport, L. *Interprocess communication*, SRI Technical Report, March 1985.

[L86a] Lamport, L. *On interprocess communication: Part I–Basic formalism*, Distributed Computing, **1**, 77-85 (1986) .

[L86b] Lamport, L. *On interprocess communication: Part II–Algorithms*, Distributed Computing, **1**, 86-101 (1986).

[Mis] Misra, J. *Axioms for memory access in asynchronous hardware systems*, ACM Trans. on Programming Languages and Systems, **8**, 142-153 (1986).

[Mor] Möring, R. H. *Algorithmic aspects of comparability graphs and interval graphs* in Graphs and Order (NATO ASI C147), I. Rival ed D. Reidel, 41-101, (1985).

[Ore] Ore, O. *Theory of Graphs*, Amer. Math. Soc. Colloq. Publ. **38**, Providence, (1962).

[Pet] Peterson, G. L. *Concurrent reading while writing*, ACM Trans. on Programming Languages and Systems, **5**, 46-55 (1983).

[Tro] Tromp, J., *How to Construct an Atomic Variable*, International Workshop on Distributed Algorithms, LNCS, Springer Verlag **392**, 292-302 (1989).

[Vid] Vidyasankar, K. *Converting Lamport's regular register to atomic register* , Information Processing Letters **28**, 287-290 (1988).

[VA] Vitányi, P. , Awerbuch, B. *Atomic shared register acces by asynchronous hardware* , Proceedings of the 27^{th} Symposium on Foundations of Computer Science, 223-243 (1986), Errata, Ibid, (1987).

[Wie] Wiener, N., *A contribution to the theory of relative position*, Proc. Camb. Philos. Soc. **17**, 441-449 (1914).