INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

UNE METHODE AXIOMATIQUE DE PREUVE DE PROPRIETES DE FATALITE DE PROGRAMMES PARALLELES AVEC HYPOTHESE D'EXECUTION EQUITABLE

THESE

présentée et soutenue publiquement le 31mai 1983
devant la commission d'examen
à l'Institut National Polytechnique de Lorraine
pour l'obtention du grade de
Docteur de 3e cycle En Informatique
par

DOMINIQUE MERY

Président: Mr P. COUSOT
Examinateurs: Mrs K.R. APT
M. NIVAT
J.P. THOMESSE

Au terme de ce travail de deux années, je veux remercier les différentes personnes qui m'ont aidé directement ou indirectement à réaliser cette thèse.

Tout d'abord, je voudrais remercier mes parents de m'avoir permis d'entreprendre de telles études, ainsi que ma femme de m'avoir supporté pendant cette période pleine de doutes et d'espoir.

Je n'aurais pu envisager une telle expérience sans le dynamisme, la compétence et l'amitié de Monsieur le Professeur Patrick COUSOT, Professeur à l'Université de Metz ; il a su me guider et m'encourager, je n'oublierai pas la confiance qu'il m'a accordée.

Monsieur le Professeur Claude PAIR, Professeur à l'Institut National Polytechnique de Lorraine, m'a autorisé à entrer en DEA d'Informatique ; je le remercie vivement.

Monsieur le Professeur Maurice NIVAT, Professeur à l'Université de Paris VII, s'est intéressé à mon travail. Sa compétence scientifique de réputation internationale m'impressionne et sa participation au jury de cette thèse me touche profondément. Je le remercie d'avoir accepté d'y participer.

Monsieur le Docteur APT, chargé de recherche au CNRS et actuellement au LITP, a montré une attention particulière à ce travail et l'a jugé avec un oeil de logicien et d'informaticien expérimenté. Ses remarques toujours pertinentes ont été très enrichissantes ; je le remercie d'avoir accepté de participer à ce jury.

Monsieur le Professeur THOMESSE, Professeur à l'Institut National Polytechnique de Lorraine, a critiqué les résultats développés dans cette thèse avec un regard pratique et je le remercie d'avoir accepté de participer à ce jury.

Je remercie toutes les personnes qui m'ont encouragé mais que je ne puis citer car je ne voudrais oublier personne, sans oublier Madame DRIQUERT, qui a accepté de dactylographier un manuscrit rébarbatif pour tout un profane.

D. MERY

TABLE DES MATIÈRES

0 - INTRODUCTION

Depuis les articles de FLOYD [FLO] et de HOARE [HOA], les systèmes de preuve et les méthodes de preuve concernant les programmes rencontrent un intérêt considérable dans tous les domaines de l'informatique. En effet, l'élaboration et la validation de programmes sont liées, puisque les algorithmes sont implantés dans l'environnement de l'individu et il est donc indispensable d'offrir à l'informaticien un cadre méthodologique d'études et de conception des programmes en vue d'une plus grande fiabilité. Cette voie tracée par FLOYD [FLO], poursuivie par HOARE [HOA] et BURSTALL [BUR], conduit à la conception de logiques de programme basées sur des logiques anciennes et à première vue adaptées à l'étude des propriétés de programmes. Les propriétés de programme qui sont les plus intéressantes sont classées en deux groupes : les propriétés d'invariance, comme, par exemple, la correction partielle, l'exclusion mutuelle, c'est-à-dire qu'une assertion représentant la propriété est vraie à partir d'un instant et le reste par la suite, et les propriétés de fatalité qui expriment l'accomplissement plus ou moins prompt d'un événement, comme par exemple, la terminaison d'un programme ou l'accessibilité en section critique. La logique modale de MANNA et PNUELI [MP1, MP2, MP3], la logique temporelle de PNUELI [PNU] sont deux logiques permettant d'exprimer et de prouver ces deux propriétés. D'autres logiques modales apparaissent simultanément : il s'agit de la logique dynamique de PRATT [PRA] et de la logique du processus introduite par HAREL, KOZEN et PARIKH [HKP]. L'école polonaise a contribué au développement de la logique algorithmique autour de SALWICKI [SAL]. Ces logiques ont fait l'objet de résultats abondants et intéressants quant à leur expressivité,complétude et décidabilité. Cependant, si nous prenons l'exemple de

la logique temporelle, on ne garantit pas toujours que chaque propriété puisse s'exprimer dans ce formalisme et l'article de WOLPER [WOL] est significatif de ce besoin d'ajouter des axiomes pour exprimer une propriété ; SISTLA et Co [SCFG] montrent l'impossibilité de caractériser les tampons infinis avec la logique temporelle linéaire.

Notre intérêt se porte maintenant sur une distinction que l'on peut faire au niveau des logiques temporelles. Suivant deux hypothèses très distinctes, le flot d'exécution est considéré comme linéaire ou comme arborescent ; c'est-à-dire que le temps de référence devient soit linéaire soit arborescent. LAMPORT [LAM4] fait cette distinction pour montrer que le choix d'un modèle temporel conduit à une classe de programmes donnée, ceci nous paraît surprenant et EMERSON et HALPERN [EMA2] reconsidèrent l'argumentation de LAMPORT [LAM4], pour déduire que la logique temporelle linéaire n'est pas close et peut s'exprimer dans un modèle de temps arborescent plus important.

Ces logiques du temps ont fait l'objet d'une thèse de KAMP [KAM], qui montrait la nécessité d'ajouter un opérateur de durée. Mais nous distinguerons deux politiques d'utilisation de ces logiques temporelles.

Une première politique d'utilisation de ces logiques est la spécification des propriétés de programmes, en vue d'une meilleure adéquation de la solution au problème informellement posé et une seconde concerne l'étude des mécanismes d'induction et de raisonnement, à partir d'un ensemble d'axiomes.

La première politique fait l'objet de nombreux papiers concernant l'expressivité, la complétude et décidabilité des logiques temporelles et on citera les articles de CLARKE et SISTLA [SIC], EMERSON et HALPERN [EMA1], [EMA2], BEN-ARI, MANNA et PNUELI [BMP], CLARKE, EMERSON et SISTLA [CES]. De plus, le formalisme de la logique temporelle est utilisé pour spécifier des applications

au Hardware, BOCHMANN [BOC], les propriétés des programmes distribués, SCHWARTZ et MELLIAR-SMITH [SCM1], les propriétés des processus communiquant, MANNA et WOLPER [MAW], les protocoles, SCHWARTZ et MELLIAR-SMITH [SCM2].

La seconde politique demande l'élaboration de systèmes et de méthodes de preuve. Cette voie est celle suivie au début par FLOYD [FLO], puis HOARE [HOA] et BURSTALL [BUR] donnent des méthodes de preuve pour programmes séquentiels. MANNA [MAN] et PNUELI [PNU] ont poursuivi cette politique, en utilisant la logique temporelle. De nombreux articles concernant des systèmes axiomatiques ont été donnés dans cette voie : MANNA et PNUELI [MAP1], [MAP2], [MAP3] et on citera les logiques de HOARE adaptées pour le cas de systèmes parallèles ou non-déterministes : APT [APT1], APT, FRANCEZ, DE ROEVER [AFD], NISHIMURA [NIS], APT et PLOTKIN [APP], LAMPORT et SCHNEIDER [LAS], SCHLICHTING et SCHNEIDER [SCS]. Mais ce qui nous paraît important c'est la notion de règles de preuve, comme par exemple, COUSOT et COUSOT [COC1, COC2] qui étudient l'invariance et la fatalité, et GRÜMBERG et FRANCEZ [GRF] dans un cas particulier de déduction. Dans toute cette politique, il s'agit de donner un ensemble de règles d'inférences et d'axiomes pour prouver des propriétés dans un cadre particulier.

Ce bref historique et les diverses références à ce qui est fait nous conduit à notre sujet d'étude actuelle. Nous nous intéressons aux programmes parallèles et plus précisément au cas de processus séquentiels non-déterministes partageant des variables communes avec hypothèse d'équité faible. La notion d'équité faible a fait l'objet de développements antérieurs notamment par APT et OLDEROG [APO], qui prennent en compte cette hypothèse par le biais d'une transformation syntaxique et pour des programmes DO, à la DIJKSTRA et pour la notation de HOARE [HOA]. Leur étude se borne aux propriétés de terminaison de tels programmes. L'hypothèse d'équité forte nous semble plus vague dans son

axiomatisation. La méthode proposée par APT, PNUELI, STAVI [APS] introduit explicitement des variables de retard. Nous voulons étudier l'hypothèse d'équité faible et en donner une axiomatique pour les propriétés de fatalité de programmes parallèles. Notre point de départ est l'article d'OWICKI et LAMPORT [OWL] qui prend en compte la notion d'équité faible, de façon axiomatique. Cependant exceptée la logique temporelle, la base formelle nous manque dans ce cas et ceci devient indispensable, si l'on veut s'assurer de la complétude sémantique et de la correction de telles méthodes. Ainsi, le système proposé par OWICKI et LAMPORT [OWL] est incomplet car il ne possède pas de règle d'induction. Ce qui nous pousse à développer un tel système, c'est la notion de treillis de preuve, qui permet de schématiser et de simplifier les preuves. Bon nombre d'imprécisions sont contenues dans cet article et nous pensons qu'une étude ensembliste et opérationnelle est plus adaptée que la notion de logique temporelle qui nous semble manquer de développement. Le problème des assertions est crucial dans ce genre de problème et il nous paraît trop ébauché par OWICKI et LAMPORT [OWL]. De plus, la notion d'équité a été étudiée par KUIPER et DE ROEVER [KUD] pour CSP et leur étude nous paraît mélanger les deux conceptions du temps.

Nous allons donner une base formelle à l'étude des propriétés de fatalité sous hypothèse d'équité faible, en spécifiant la classe des programmes parallèles concernés ; puis nous enrichirons le système formel d'OWICKI et LAMPORT [OWL], afin d'assurer sa correction et sa complétude sémantique. La méthode de treillis de preuve sera ensuite prouvée correcte et complète et une illustration de cette méthode pour un programme complexe montrera son utilisation facile. Il serait intéressant d'étendre cette méthode à des programmes communicants par envoi asynchrone de messages et nous montrerons ce que peut être un système

de preuve adapté pour ce genre de programmes. Mais nous donnons maintenant le détail de notre exposé.

Le chapitre I est une étude de la notion de structures abstraites sur des langages infinitaires et son utilisation pour caractériser les propriétés de fatalité. Le langage d'assertions est spécifié et les notions de point fixe, d'opérateur monotone et d'ordinal de clôture sont introduites. Ainsi on applique les résultats au cas où la structure abstraite considérée est celle que l'on peut associer à un programme considéré comme entité abstraite. La fatalité est définie par l'introduction des traces de $\mathbb{P}$, un programme donné quelconque et on montre que, pour un programme $\mathbb{P}$ sans hypothèse d'équité, on caractérise chaque fatalité par un opérateur monotone sur la structure abstraite associée à $\mathbb{P}$ et on associe un ordinal dit de fatalité à chaque fatalité.

L'étude du chapitre précédent ne permet pas de généraliser facilement au cas de la fatalité sous hypothèse d'équité faible pour des programmes parallèles formés de programmes séquentiels non-déterministes partageant des variables communes. Une étude particulière à ce type d'hypothèse exige de spécifier tout d'abord la syntaxe et la sémantique opérationnelle des programmes parallèles étudiés, puis la notion de traces, d'assertions de contrôle et d'actions atomiques. L'hypothèse d'équité faible est caractérisée à l'aide des actions atomiques, des assertions de contrôle et de la notion de modèle de fatalité. On généralise alors les résultats obtenus dans le chapitre I pour l'étude de la fatalité sous hypothèse d'équité faible. On associe ainsi à toute assertion un opérateur monotone, un ordinal et une suite d'assertions qui conduisent fatalement à Q sous hypothèse d'équité faible. Nous supposons qu'il y a un mécanisme

de synchronisation et d'exclusion mutuelle au niveau des variables du programm

Dans le chapitre III, nous enrichissons le système axiomatique de OWICKI et LAMPORT [OWL] ; le système axiomatique obtenu, noté OL, est prouvé correct et sémantiquement complet à partir des résultats du chapitre II. La preuve de complétude sémantique est très complexe et demande la construction préalable d'une suite d'assertions indicée par les ordinaux et stable à partir de l'ordinal de fatalité.

A partir du système axiomatique OL, on énonce, dans le chapitre IV, la méthode des treillis de preuve. La preuve de correction et la preuve de complétude de cette méthode est faite facilement à partir des résultats précédents sur OL. Nous illustrons notre méthode par la preuve d'une propriété de fatalité relative à l'algorithme du ramasse-miettesproposé par BEN-ARI [BEN2], dans lequel nous avons trouvé une erreur par cette méthode. L'hypothèse d'équité faible avait été caractérisée par un modèle particulier de fatalité et nous donnons une implantation de l'équité faible par l'intermédiaire de structures bien fondées. Enfin nous donnons une approche abstraite à la notion de treillis de preuve, en tentant de caractériser les règles et axiomes par des opérations et objets abstraits.

Le chapitre V est une généralisation de OL pour le cas de programmes distribués communiquant par envoi et réception, de façon asynchrone, de messages dans un tampon. Nous considérons les propriétés d'invariance relatives à ce type de programme et étendons la méthode d'invariance d'OWICKI et GRIES [OWG]. OL est ensuite augmenté d'axiomes prenant en compte les primitives d'envoi et de réception de messages et nous donnons une illustration de ces deux méthodes avec l'algorithme amélioré de RICART et AGRAWALA : on donne les preuves d'exclusion mutuelle et d'accessibilité en section critique.

Notre exposé se termine par une conclusion, suivie de la bibliographie. Quelques directions futures sont données pour la suite de nos recherches.

I - PRÉLIMINAIRES LOGIQUES

I - PRÉLIMINAIRES LOGIQUES

Nous donnons dans cette partie des résultats relatifs aux structures logiques abstraites mises en jeu dans les sémantiques opérationnelles des programmes. MOSCHOVAKIS [MOS] étudie les notions de structures abstraites et d'induction sur ces dernières : ses résultats généralisent ceux obtenus par SPECTOR [SPE] et KLEENE [KLE]. Cependant, il raisonne sur un langage des prédicats du premier ordre simple et, afin de garantir une expressivité à nos langages d'assertion, il nous paraît nécessaire de donner une version "langage infinitaire" de ses résultats et d'adapter ses études à la nôtre.

Notre volonté ici est de généraliser le résultat de APT et PLOTKIN [APO] à des structures abstraites, qui, nous le rappelons, prouvaient que les ordinaux mis en jeu au cours de preuves de terminaison de programmes non-déterministes sont récursifs, si on se restreint au domaine des entiers pour les variables du programme et à des fonctions et relations effectivement calculables. Ce résultat n'illustre pour nous nullement le désir de ses auteurs qui étudient le non-déterminisme dénombrable sur des structures dénombrables mais plutôt le non-déterminisme récursif. La suite de notre exposé comprend une définition des langages infinitaires ainsi que de la notion de structure abstraite. Les résultats de MOSCHOVAKIS [MOS] sont ensuite adaptés à notre étude. De plus, les notions de codage sont examinées.

I.1.- LANGAGE INFINITAIRE ET STRUCTURE ABSTRAITE

Définition [I.1].1 :

Un langage $\mathcal{L}$ est une collection de symboles formée de trois classes

de symboles :

- la classe des symboles de constantes, notée $\mathcal{C}$.
- la classe des symboles de relations, notée $\mathcal{R}$.
- la classe des symboles de fonctions, notée $\mathcal{F}$.

ie : $\mathcal{L} = \{C_i/i \in I\} \cup \{R_j/j \in J\} \cup \{f_k/k \in K\}$.

On étend le langage $\mathcal{L}$ en lui ajoutant les symboles suivants : (,), $\wedge$(conjonction), $\forall$(quantificateur universel), $\sim$(négation), $v_0,\ldots,v_n$ ...(variables).

A l'aide de $\mathcal{L}$, on définit l'ensemble des termes sur $\mathcal{L}$, noté $\mathcal{T}[\mathcal{L}]$ l'ensemble des formules atomiques, noté $\mathcal{A}[\mathcal{L}]$, et l'ensemble des formules sur $\mathcal{L}$, noté $\mathcal{F}[\mathcal{L}]$. On désignera abusivement le quadruplet $(\mathcal{L},\mathcal{T}[\mathcal{L}],\mathcal{A}[\mathcal{L}],\mathcal{F}[\mathcal{L}])$ par $\mathcal{L}$ et on parlera du langage $\mathcal{L}$.

$\Rightarrow$ désigne le symbole de l'implication et $=$ le symbole d'égalité.

Définition [I.1].2 :

Soit $\mathcal{L}$ un langage.

(1) $\mathcal{T}[\mathcal{L}]$ est le plus petit ensemble contenant les variables et les constantes de $\mathcal{L}$ et stable par application finie de la règle suivante :
si $f \in \mathcal{F}$, $t_1,\ldots,t_n \in \mathcal{T}[\mathcal{L}]$, alors $f(t_1,\ldots,t_n) \in \mathcal{T}[\mathcal{L}]$.

(2) $\mathcal{A}[\mathcal{L}]$ est le plus petit ensemble contenant les objets du type $t_1=t_2$ où t_1 et t_2 sont des termes de $\mathcal{L}$ et stable par application finie de la règle suivante :
si $R \in \mathcal{R}$, $t_1,t_2,\ldots,t_n$ des termes de $\mathcal{T}[\mathcal{L}]$, alors
$R(t_1,\ldots,t_n) \in \mathcal{A}[\mathcal{L}]$.

(3) $\mathcal{F}[\mathcal{L}]$ est le plus petit ensemble contenant $\mathcal{A}[\mathcal{L}]$ et stable par application finie des règles suivantes :
- si $\varphi,\psi \in \mathcal{F}[\mathcal{L}]$, alors $\varphi\wedge\psi, \sim\varphi \in \mathcal{F}[\mathcal{L}]$.
- si $\varphi \in \mathcal{F}[\mathcal{L}]$, v une variable de $\mathcal{L}$, alors $\forall v\varphi \in \mathcal{F}[\mathcal{L}]$.

Nous définissons maintenant la notion de langage infinitaire $\mathcal{L}_{\alpha\beta}$, en ajoutant une règle supplémentaire de construction aux formules. On note $|A|$, le cardinal de l'ensemble A.

Définition [I.1].3 :

Un langage infinitaire $\mathcal{L}_{\alpha\beta}$ où α,β sont des cardinaux infinis est un langage tel que :

- $\mathcal{T}[\mathcal{L}_{\alpha\beta}]$ est défini comme ci-dessus.
- $\mathcal{A}[\mathcal{L}_{\alpha\beta}]$ est défini comme ci-dessus.
- $\mathcal{F}[\mathcal{L}_{\alpha\beta}]$ est défini par les mêmes conditions et mêmes règles que dans le cas ci-dessus et on ajoute les deux règles suivantes :
  - si Φ est un ensemble de formules de $\mathcal{L}_{\alpha\beta}$ tel que $|\Phi| < \alpha$, alors $\wedge\Phi \in \mathcal{F}[\mathcal{L}_{\alpha\beta}]$.
  - si $\varphi \in \mathcal{F}[\mathcal{L}_{\alpha\beta}]$ et $\mathcal{V}$ est un ensemble de variables de cardinal $|\mathcal{V}| < \beta$, alors $(\forall\mathcal{V})\varphi \in \mathcal{F}[\mathcal{L}_{\alpha\beta}]$.

Définition [I.1].4 :

Soit un langage $\mathcal{L}$. On appelle langage du second ordre un langage dans lequel on autorise une quantification sur les relations. On notera $\mathcal{L}_2$ un tel langage.

Un exemple de langage infinitaire est $\mathcal{L}_{\omega_1\omega}$ qui autorise les unions

dénombrables et l'utilisation d'un nombre fini de variables quantifiées. Nous ajoutons comme d'habitude les formules $\forall\Phi$, $\varphi\vee\psi$, $\exists x\varphi$.

Soit A un ensemble infini. On associe à A un langage $\mathcal{L}^A$ tel que :

- $\mathcal{L}^A$ possède un ensemble infini de variables.
- $\mathcal{L}^A$ possède une constante C pour chaque élément de A.
- $\mathcal{L}^A$ possède un ensemble infini de variables de relations n-aires, pour chaque $n \geq 1$.
- $\mathcal{L}^A$ possède un symbole de relation constante pour chaque relation R sur A, en particulier les symboles $=$, $\sim$, $\wedge$, $\vee$, $\Rightarrow$, $\exists$, $\forall$.

Les formules de $\mathcal{F}[\mathcal{L}]$ peuvent être interprétées naturellement de la façon suivante :

- chaque constante de $\mathcal{L}$ relative à une constante de A est interprétée comme la constante de A.
- chaque symbole de relation sur A est interprété par la relation A correspondante.
- chaque symbole de fonction sur A est interprété par la fonction sur A correspondante.
- chaque terme (resp. chaque formule atomique, resp. chaque formule) est interprété de façon structurelle en utilisant les 3 points ci-dessus.

Définition [I.1].5 :

Soit A un ensemble infini ; $\mathcal{L}^A$ un langage sur A dont les seules constantes sont $=, R_1,\ldots,R_\ell$.

(a) On appelle structure abstraite $\mathcal{U}$ sur A relativement au langage $\mathcal{L}^A$ le $\ell+1$-uplet $(A,R_1,\ldots,R_\ell)$.

(b) Soit $\mathcal{U}$ une structure abstraite sur A.
φ une formule de $\mathcal{L}^A$.
(la définition reste correcte pour $\mathcal{L}^A$ un langage du second ordre).

(1) On dit que les occurrences de la variable de relation S sont monotones dans $\varphi(S)$ relativement à $\mathcal{U}$, si, dans l'interprétation naturelle sur $\mathcal{U}$,

$$\varphi(S) \wedge \bar{S} \subseteq S' \Rightarrow \varphi(S').$$

(2) On dit qu'une relation n-aire R sur A est $\mathcal{L}^A$-monotone et $\mathcal{L}^A$-inductive sur $\mathcal{U}$, ou inductive sur $\mathcal{U}$, s'il y a des constantes $a_1,\ldots,a_k$ dans A et un opérateur

$$\Gamma : \mathcal{P}(A^{k+n}) \to \mathcal{P}(A^{k+n})$$

élémentaire sur $\mathcal{U}$ tel que $R(\bar{x}) \Leftrightarrow (\bar{a},\bar{x}) \in I_\Gamma$.
$\bar{a}$ est le k-uplet $(a_1,\ldots,a_k)$ et $\bar{x}$ est le n-uplet $(x_1,\ldots,x_n)$.
I_Γ est le sous-ensemble de A^{k+n} désignant la clôture de l'opérateur Γ.

(3) Un opérateur $\Gamma : \mathcal{P}(A^n) \to \mathcal{P}(A^n)$, $n \geq 1$ est élémentaire sur $\mathcal{U}$, s'il y a une formule φ de $\mathcal{L}^A$ telle que :

$$\varphi \simeq \varphi(\bar{x},S) \simeq \varphi(x_1,\ldots,x_n,S,=,R_1,\ldots,R_\ell)$$

(i) Les occurrences des constantes de relation dans $\varphi(\bar{x},S)$ sont parmi $=,R_1,\ldots,R_\ell$.

(ii) les occurrences des symboles de relation $=,R_1,\ldots,R_\ell$ apparaissent dans $\varphi(\bar{x},S)$ de façon monotone.

(iii) La seule variable de relation de $\varphi(\bar{x},S)$ est la variable n-aire S.

(iv) Les variables individuelles libres sont parmi $x_1,\ldots,x_n$.

(v) La formule $\varphi(\bar{x},S)$ définit Γ :

Pour chaque $S \subseteq A^n$, $\Gamma(S) = \{\bar{x} : \varphi(\bar{x},S)\}$.

Dans le cas b-3, on notera $I_\varphi = I_\Gamma$ et on dira que φ est associée à Γ et vice versa. I_Γ est construit de façon inductive par les méthodes de construction de point fixe.

Théorème [I.1].1 :

Soit A un ensemble infini, Γ un opérateur monotone sur $\mathcal{P}(A^n)$, I_Γ^α, I_Γ tels que :

$I_\Gamma^\alpha = \Gamma(\bigcup_{\beta<\alpha} I_\Gamma^\beta)$ et $I_\Gamma = \bigcup_\alpha I_\Gamma^\alpha$; α, α_0, β des ordinaux.

(1) si $\alpha \leqslant \beta$, alors $I_\Gamma^\alpha \subset I_\Gamma^\beta$.

(2) Il existe un plus petit ordinal α_0 tel que :

- $\alpha_0 < |A|^+$ où $|A|^+$ est le plus petit ordinal de cardinalité strictement plus grande que celle de A.
- $I_\Gamma = I_\Gamma^{\alpha_0}$.
- $\forall \alpha \geqslant \alpha_0$, $I_\Gamma^{\alpha_0} = I_\Gamma^\alpha$.

On note $\alpha_0 = \|\Gamma\|$, l'ordinal de clôture de Γ sur A.

(3) L'ensemble I_Γ est le plus petit ensemble tel que :

$\Gamma(I_\Gamma) = I_\Gamma$ et $I_\Gamma = \cap\{S : \Gamma(S) = S\}$.

Preuve : On verra par exemple MOSCHOVAKIS [MOS] ou HINMAN [HIN]. □

On peut donner quelques caractérisations de la notion d'inductivité et ceci est l'objet du théorème suivant qui n'est donné que pour préciser cette notion.

Théorème [I.1].2 :

Soit A un ensemble infini,

$R,Q_1,Q_2,\ldots$ des relations sur A.

$\mathcal{U}_0$ la structure suivante (A).

(i) Les relations $x = y$, $x \neq y$, $x = c$ (c un élément fixé de A), $x \neq c$ sont inductifs (sur $\mathcal{U}_0$).

(ii) Si R est inductive sur $(A,Q_1,\ldots,Q_m)$ et chaque Q_i a une occurrence dans la liste $Q_1' \ldots Q_m'$, alors R est inductive dans $Q_1',\ldots,Q_m'$.

(iii) R est inductive sur (A,R).

(iv) si $P(\bar{x}) \Leftrightarrow R_1(\bar{x}) \bar{\wedge} R_2(\bar{x})$

$Q(\bar{x}) \Leftrightarrow R_1(\bar{x}) \underline{\vee} R_2(\bar{x})$

alors P et Q sont inductifs sur (A,R_1,R_2).

(v) si $P(\bar{x}) \Leftrightarrow \forall y R(y,\bar{x})$

$Q(\bar{x}) \Leftrightarrow \exists y R(y,\bar{x})$

alors P et Q sont inductifs sur (A,R).

Preuve :

On suppose que l'on dispose d'un langage pour la structure abstraite $(A,R,Q_1,\ldots,Q_i,\ldots)$ noté $\mathcal{L}^A$. Les preuves sont triviales. □

Nous poursuivons nos innovations par quelques définitions relatives à la

définissabilité inductive sur une structure abstraite.

Définition [I.1].6 :

Soit $\mathcal{U} = (A,R_1,\ldots,R_\ell)$ une structure abstraite,
$\mathcal{L}^A$ un langage sur $\mathcal{U}$.

(1) Une relation élémentaire sur $\mathcal{U}$ est une relation définie par une formule du langage $\mathcal{L}^A$.

(2) Une relation R sur $\mathcal{U}$ est un point fixe, s'il y a une formule $\varphi \simeq \varphi(\bar{x},S)$ monotone en S dans le langage $\mathcal{L}^A$ telle que $R = I_\varphi$.

(3) Une relation R est $\mathcal{L}^A$-monotone, $\mathcal{L}^A$-inductive et $\mathcal{L}^A$-définissable, ou inductive sur $\mathcal{U}$, s'il y a un point fixe I_φ et des constantes $\bar{a} = (a_1,\ldots,a_k)$ dans A telles que : $R(\bar{x}) \Leftrightarrow (\bar{a},\bar{x}) \in I_\varphi$.

(4) Une relation R est coinductive sur $\mathcal{U}$, si R est inductive sur $\mathcal{U}$.

(5) Une relation R est hyperélémentaire sur $\mathcal{U}$, si R est inductive et coinductive sur $\mathcal{U}$.

(6) Une fonction $f : A^n \to A^m$ est élémentaire (resp. hyperélémentaire), si sa relation de représentation $G_f(\bar{x},\bar{y}) \Leftrightarrow f(\bar{x}) = \bar{y}$ est élémentaire (resp. hyperélémentaire).

Théorème [I.1].3 (de transitivité) :

Soit A un ensemble infini, $R,Q_1,\ldots,Q_m$ des relations sur A. Si R est inductive sur $(A,Q,Q_1,\ldots,Q_m)$ et Q inductive sur $(A,Q_1,\ldots,Q_m)$, alors R est inductive sur $(A,Q_1,\ldots,Q_m)$.

Preuve :

On verra essentiellement MOSCHOVAKIS [MOS]. □

Ce résultat va nous permettre de définir et déduire les propriétés de clôture simples de la classe des relations inductives ou hyperélémentaires.

Une relation $P(\bar{x})$ est définie à partir d'une relation $R(y_1,\ldots,y_m)$ par substitution hyperélémentaire, s'il y a des fonctions hyperélémentaires $f_1(\bar{x}),\ldots,f_m(\bar{x})$ telles que :

$$P(\bar{x}) \Leftrightarrow R(f_1(\bar{x}),\ldots,f_m(\bar{x})).$$

Théorème [I.1].4 :

La classe des relations inductives sur une structure $\mathcal{U}$ est fermée sous les opérations $\wedge,\vee,\exists,\forall$ et les substitutions hyperélémentaires.

La classe des relations hyperélémentaires sur une structure $\mathcal{U}$ comprend toutes les relations élémentaires et est fermée par toutes les opérations $\sim,\wedge,\vee,\Rightarrow,\exists,\forall$ et substitution hyperélémentaire.

Preuve :

Conséquence du théorème précédent. □

Le langage du second ordre $\mathcal{L}_2^A$ est obtenu en autorisant les quantifications sur les variables de relation dans $\mathcal{L}^A$. Le langage du second ordre $\mathcal{L}^A$ pour une structure abstraite $\mathcal{U}$ possède ses formules dans $\mathcal{L}_2^A$ dont les (symboles) constantes de relation sont parmi $=,R_1,\ldots,R_\ell$ ($\mathcal{U} = (A,R_1,\ldots,R_\ell)$). Les relations définissables sur $\mathcal{L}^A$ sont appelées du second ordre.

Définition [I.1].7 :

(1) On dit qu'une relation R sur $\mathcal{U}$ est π^1_1, si
$R(\bar{x}) \Leftrightarrow (\forall S_1)\ldots(\forall S_k)\varphi(S_1,\ldots,S_k,\bar{x})$
où φ est une formule (élémentaire) sur $\mathcal{L}^A$, langage de $\mathcal{U}$.

(2) On dit que R est Σ^1_1, si $\sim R$ est π^1_1.

(3) On dit que R est Δ^1_1, si R est π^1_1 et Σ^1_1.

Nous caractérisons les relations inductives et hyperélémentaires sur une structure abstraite par π^1_1 et Δ^1_1 :

Théorème [I.1].5 :

(1) Toute relation inductive sur une structure abstraite $\mathcal{U}$ est π^1_1.

(2) Toute relation hyperélémentaire sur une structure abstraite $\mathcal{U}$ est Δ^1_1

Preuve :

Si R est inductive, alors il existe une formule $\varphi(\bar{u},\bar{x},S)$ et des constantes $\bar{a}$ telles que :

$$R(\bar{x}) \Leftrightarrow (\bar{a},\bar{x}) \in I_\varphi \ ;$$

I_φ est le point fixe d'un opérateur défini par φ :

$$R(x) \Leftrightarrow (\forall S)[(\forall\bar{u})(\forall\bar{x}')[S(\bar{u},\bar{x}')\varphi(\bar{u},\bar{x}',S)] \Rightarrow S(\bar{a},\bar{x})]$$

CQFD. □

Si A est égal à $\mathbb{N}$, alors réciproquement toute relation π^1_1 [resp. Δ^1_1] est élémentaire [resp. hyperélémentaire] (on verra HINMAN [HIN], SHOENFIELD [SHO], ROGERS [ROG], MOSCHOVAKIS [MOS]). On note parfois $\mathbb{N}$ par ω.

On peut préciser la nature des ordinaux mis en jeu dans les opérateurs définissant des relations inductives ou hyperélémentaires. Nous avons associé à tout opérateur monotone sur A un ordinal de cardinalité au plus celle de A. Si Γ est un opérateur élémentaire sur A, on lui associe une formule φ sur $\mathcal{L}^A$ et on note $\|\varphi\|$ l'ordinal $\|\Gamma\|$, appelé l'ordinal de clôture de Γ défini par φ :

$\|\varphi\| = \mathrm{Sup}(|\bar{x}|_\varphi+1 : \bar{x} \in I_\varphi)$ où $|\bar{x}|_\varphi$ est l'ordinal associé à $\bar{x}$ dans la construction de I_φ.

Etant donnée une structure abstraite $\mathcal{U} = (A,R_1,\ldots,R_\ell)$ et un langage $\mathcal{L}^A$ pour $\mathcal{U}$.

L'ordinal de clôture de $\mathcal{U}$, noté $\kappa^{\mathcal{U}}$, est tel que : $\kappa^{\mathcal{U}} = \mathrm{Sup}\{\|\varphi\|$/les occurrences de S dans φ sur $\mathcal{L}^A$ sont monotones$\}$.

Par exemple, soit $\mathcal{U}$, la structure $(\mathbb{N},\leqslant)$. $\kappa^{\mathcal{U}} = |\mathbb{N}|^+ = |\omega|^+$, le premier ordinal non-dénombrable encore noté ω_1. ω_1 n'est pas le premier ordinal non-récursif noté ω_1^{CK} ou κ par APT et PLOTKIN [APP].

Théorème [I.1].6 (de clôture) :

Soit $\mathcal{U}$ une structure abstraite ; $\mathcal{L}^A$ un langage pour $\mathcal{U}$, $\varphi \simeq \varphi(\bar{x},S)$ une formule où les occurrences de S sont monotones ; I_φ, le point fixe de φ, est hyperélémentaire sur $\mathcal{U}$ ssi $\|\varphi\| < \kappa^{\mathcal{U}}$.

Preuve :

On consultera MOSCHOVAKIS [MOS]. □

Ca résultat étend les résultats de l'arithmétique donnés par SPECTOR [SPE] et KLEENE [KLE] où hyperélémentaire est en fait hyperarithmétique. Mais notre intérêt se portera sur les opérations inductives et pour cela on introduit la notion de norme.

Définition [I.1].8 :

Soit A un ensemble et α un ordinal.

(1) On appelle norme sur un ensemble A une fonction $\sigma : A \to \alpha$; α est la longueur de σ.

(2) Une norme $\sigma : A \to \alpha$ est inductive sur $\mathcal{U}$, s'il y a des relations $J_\sigma(\bar{x},\bar{y})$, $\bar{J}_\sigma(\bar{x},\bar{y})$ telles que :

(1) J_σ et $\sim\bar{J}_\sigma$ sont inductives,

(2) Si $\bar{y} \in A$, alors $(\forall\bar{x})\{[\bar{x} \in A \wedge \sigma(\bar{x}) < \sigma(\bar{y})] \leftrightarrow J_\sigma(\bar{x},\bar{y}) \leftrightarrow \bar{J}_\sigma(\bar{x},\bar{y})\}$.

Théorème [I.1].7 :

Soit $\mathcal{U}$ une structure abstraite.

$\mathcal{L}^A$ un langage pour $\mathcal{U}$.

Toute relation inductive sur $\mathcal{U}$ admet une norme inductive.

Preuve :

Il suffit d'utiliser le résultat suivant : en affectant un ordinal à chaque élément de I_φ on définit une norme inductive (cf. MOSCHOVAKIS [MOS]).

Théorème [I.1].8 :

Soit $\mathcal{U}$ une structure abstraite,

P une relation inductive sur $\mathcal{U}$,

$\kappa = \kappa^{\mathcal{U}}$,

$\sigma : P \to \alpha$ une norme inductive sur P.

(1) $\alpha \leqslant \kappa$.

(2) $\alpha < \kappa$ ssi P est hyperélémentaire sur $\mathcal{U}$.

Preuve : cf MOSCHOVAKIS [MOS]. □

Les relations inductives sur une structure abstraite $\mathcal{U}$ sont notre centre d'intérêt et l'étude des propriétés de fatalité se ramène à l'étude d'opérateurs et de relations sur des structures abstraites.

Nous n'avons pas mentionné explicitement les résultats du second ordre car nous n'en avons point besoin.

Cette notion abstraite nous semble intéressante car elle ignore la structure véritable et nos résultats peuvent être énoncés de façon plus formelle et plus générale.

Si nous appliquons notre étude à l'arithmétique IN, les relations élémentaires sont arithmétiques et les relations hyperélémentaires sont hyperarithmétiques. On a en particulier le théorème de Souslin-Kleene.

Théorème [I.1].9 :

(1) Toute relation hyperarithmétique est Δ^1_1 et réciproquement.

(2) Toute relation arithmétique est π^1_1 et réciproquement.

Ces théorèmes sont dûs à SPECTOR [SPE], KLEENE [KLE], SOUSLIN .

La notion de codage est nécessaire, on peut consulter la preuve dans SHOENFIELD [SHO], HINMAN [HIN], MOSCHOVAKIS [MOS] ou ROGERS [ROG].

I.2.- STRUCTURE ABSTRAITE DE PROGRAMME

Nous définissons une structure abstraite adaptée à l'étude des programmes ; cela revient à formaliser et abstraire la notion de sémantique opérationnelle. Mais il faut faire attention à la notion de variables de programme.

Nous considèrerons les variables de programme comme des constantes pour un certain ensemble de base et la valeur d'une variable sera en fait l'image de cette variable pour une certaine fonction.

Nous supposons qu'il existe un ensemble dont les objets sont des programmes, sans préciser la notion de programme.

Cependant, à chaque programme on associe un ensemble de variables de programme $\mathbb{P}$ $\mathcal{V}[\mathbb{P}]$, un ensemble domaine de $\mathbb{P}$ noté $\mathcal{D}[\mathbb{P}]$, un ensemble de relations $\mathcal{R}[\mathbb{P}]$ ainsi qu'un ensemble de fonctions notée $\mathcal{F}[\mathbb{P}]$ sur $\mathcal{D}[\mathbb{P}] \cup \mathcal{V}[\mathbb{P}]$. Nous verrons dans le paragraphe suivant comment justifier de telles hypothèses.

Définition [I.2].1 :

Soit $\mathbb{P}$ un programme, $\mathcal{V}[\mathbb{P}]$, $\mathcal{D}[\mathbb{P}]$, $\mathcal{F}[\mathbb{P}]$ comme ci-dessus. On appelle structure abstraite de $\mathbb{P}$ et on note $\mathcal{U}[\mathbb{P}]$, la structure abstraite $<\mathcal{V}[\mathbb{P}] \cup \mathcal{D}[\mathbb{P}], R_1,\ldots,R_\ell>$ où $R_1,\ldots,R_\ell$ sont des éléments de $\mathcal{R}[\mathbb{P}]$.

A l'aide de $\mathcal{U}[\mathbb{P}]$, on définit la notion d'états de $\mathbb{P}$ et on notera $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$, l'ensemble des états de $\mathbb{P}$ de nombre n.

On supposera que $\mathcal{D}[\mathbb{P}]$ contient un sous-ensemble $\mathcal{L}[\mathbb{P}]$ d'éléments appelés des étiquettes dont le nombre est dénombrable. On distinguera, de plus, les fonctions de $\mathcal{V}[\mathbb{P}]$ dans $\mathcal{D}[\mathbb{P}] \setminus \mathcal{L}[\mathbb{P}]$ et on les notera m.

Définition [I.2].2 :

On appelle état de $\mathbb{P}$ d'indice n, tout $n+1$-uplet noté s, tel que $s = (s_1,\ldots,s_{n+1})$ où les s_i sont tels que :

$$[\forall i \in \{1,\ldots,n\}, s_i \in \mathcal{L}[\mathbb{P}]] \wedge [s_{n+1} \in (\mathcal{V}[\mathbb{P}] \to (\mathcal{D}[\mathbb{P}] \setminus \mathcal{L}[\mathbb{P}]))] \wedge$$

$$[(\forall i \in \{1,\ldots,n\}, s_i = s) \vee (\forall i \in \{1,\ldots,n\}, \forall j \in \{1,\ldots,n\}, (i \neq j) \Rightarrow (s_i \neq s_j))]$$

On note $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$, l'ensemble des états de P d'indice n.

$S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ est un sous-ensemble de $(\mathcal{V}[\mathbb{P}] \cup \mathcal{D}[\mathbb{P}])^{n+1}$.

Théorème [I.2].1 :

Soit n un entier naturel, $\mathcal{U}[\mathbb{P}]$ une structure abstraite de $\mathbb{P}$.

$S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ est inductif sur $\mathcal{U}[\mathbb{P}]$ et est un point fixe sur $\mathcal{U}[\mathbb{P}]$.

Preuve :

Notons $A = \mathcal{V}[\mathbb{P}] \cup \mathcal{D}[\mathbb{P}]$. Soit $\mathcal{L}^A$ un langage sur $\mathcal{U}[\mathbb{P}]$, et $\varphi(\bar{x},S)$ la formule suivante :

$$\varphi(\bar{x},S) \equiv \left[\bigvee_{i\in\{1,\ldots,n\}}\left[\bigvee_{s\in\mathcal{L}[\mathbb{P}]}(s_i=s)\right]\right] \wedge \Big[(s_1=s_2=\ldots=s_n) \vee$$

$$(\forall(i,j)\in[n]^2, s_i \neq s_j)\Big] \wedge [s_{n+1} = f(\bar{x})]$$

où f est une fonction de $\mathcal{V}[\mathbb{P}]$ dans $\mathcal{D}[\mathbb{P}] \setminus \mathcal{L}[\mathbb{P}]$.

Les occurrences de S dans φ sont monotones et il suffit de considérer comme $\bar{a}$ l'ensemble des variables de $\mathbb{P}$ que nous supposons en nombre fini.

On vérifie alors d'après la définition de la notion d'états que $\forall s \in S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$, $(\bar{v},s) \in I_\varphi$. □

Nous allons associer à $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ une relation de transition notée $t_n[\mathbb{P}]$ et elle nous permettra de modéliser l'exécution abstraite de $\mathbb{P}$ sur $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ pour l'indice n.

Définition [I.2].3 :

Soient s,s' deux éléments de $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$. On suppose qu'il existe des relations binaires sur $\mathcal{L}[\mathbb{P}]$, notées suc_j, $j \in \{1,\ldots,n\}$, $t_n[\mathbb{P}](s,s')$ ssi

$[(s_1=\ldots=s_n)\wedge(s_{n+1}=f(\bar{v}))\wedge suc_1(s_1,s_1')\wedge\ldots\wedge suc_n(s_n,s_n')\wedge(\forall(i,j)\in[n]^2, s_i\neq s_$

$\wedge(s_{n+1}'=f'(\bar{v}))\wedge(f'(\bar{v})=f(\bar{v}))]$

$\vee$

$[(s_1'=\ldots=s_n')\wedge(s_{n+1}'=f'(\bar{v}))\wedge suc_1(s_1,s_1')\wedge\ldots\wedge suc_n(s_n,s_n')\wedge(\forall(i,j)\in[n]^2,$

$s_i\neq s_j)\wedge(s_{n+1}=f(\bar{v}))\wedge(f'(\bar{v})=f(\bar{v}))]$

$\vee$

$[(\forall(i,j)\in[n]^2,(s_i\neq s_j)\wedge(s_i'\neq s_j'))\wedge suc_1(s_1,s_1')\wedge\ldots\wedge suc_n(s_n,s_n')\wedge$

$(r^*_{n+1}(s_{n+1},s'_{n+1}))]$.

r^*_{n+1} désigne la transformation de la fonction s_{n+1} en s'_{n+1}.

Dans cette définition, on distingue trois parties entre crochets correspondant respectivement à l'initialisation de $\mathbb{P}$, l'action accédant à la fin de $\mathbb{P}$ et l'exécution d'un pas de $\mathbb{P}$. Nous aurions pu utiliser une disjonction infinitaire dénombrable pour les définitions relatives à $suc_1,\ldots,suc_n$ et r^*_{n+1} mais nous supposerons que le nombre des fonctions et des relations est au plus dénombrable. Ceci nous permet de nous limiter à un langage $\mathcal{L}_{\omega_1\omega}$.

Théorème [I.2].2 :

Soit $\mathbb{P}$ un programme, $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ un ensemble d'états de $\mathbb{P}$,
$t_n[\mathbb{P}]$ une relation de transition comme ci-dessus et
$t^*_n[\mathbb{P}]$ la fermeture transitive de $t_n[\mathbb{P}]$.
$t^*_n[\mathbb{P}]$ est inductive sur $\mathcal{U}[\mathbb{P}]$.

Preuve :

$S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ est inductive sur $\mathcal{U}[\mathbb{P}]$. La définition de $t_n[\mathbb{P}]$ ne

met en jeu que des relations (inductives) élémentaires de $\mathcal{U}[\mathbb{P}]$. La transition $t_n[\mathbb{P}](s,s')$ peut s'écrire par une formule $\varphi(\bar{x},S)$ où φ est élémentaire sur $\mathcal{U}[\mathbb{P}] \cup S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$. $t^*_n[\mathbb{P}](s,s')$ est la plus petite relation transitive contenant $t_n[\mathbb{P}]$.

Soit Γ tel que $\Gamma(S) = S\vee\varphi(\bar{v},S)$. Alors I_Γ définit
$\left(S \subset S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}] \times S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]\right)$

$t^*_n[\mathbb{P}]$ sur $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$. Puisque $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ est inductif sur $\mathcal{U}[\mathbb{P}]$, par transitivité (cf MOSCHOVAKIS [MOS]), $t^*_n[\mathbb{P}]$ est inductive sur $\mathcal{U}[\mathbb{P}]$. □

Théorème [I.2].3 :

Soit $\mathbb{P}$ un programme, $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ un ensemble d'états de $\mathbb{P}$.
$\mathcal{U}[\mathbb{P}]$ une structure abstraite de $\mathbb{P}$ possédant un nombre dénombrable de relations et de fonctions.
Alors $\left\{s' \in S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]/t_n[\mathbb{P}](s,s')\right\}$ est dénombrable, pour tout s de $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$.

Preuve :

Trivialement, le nombre des fonctions et relations sur $\mathcal{U}[\mathbb{P}]$ est dénombrable. Si $\left|\left\{s' \in S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]/t_n[\mathbb{P}](s,s')\right\}\right|$ est plus grand que ω, alors il existe un nombre non dénombrable de relations et fonctions dans $\mathcal{U}[\mathbb{P}]$, puisque la définition de $t_n[\mathbb{P}]$ met en jeu un nombre non dénombrable de ces individus. Ceci est contraire à l'hypothèse. □

Nous avons besoin d'étudier l'ensemble formé par $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ et $t_n[\mathbb{P}]$ et pour cela il nous faut définir la notion de langage d'assertions de $\mathbb{P}$.

Définition [I.2].4 :

Soit $\mathbb{P}$ un programme ; $\mathcal{U}[\mathbb{P}]$ une structure abstraite de $\mathbb{P}$;

$S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ comme ci-dessus.

On appelle langage d'assertions de $\mathbb{P}$ relativement à $\mathcal{U}[\mathbb{P}]$ et on note $\mathcal{A}[\mathbb{P}]$, tout langage tel qu'il existe deux fonctions d'abstractions $\rho[\mathbb{P}]$ et de concrétisation $\tilde{\rho}[\mathbb{P}]$

$$\mathcal{A}[\mathbb{P}] \underset{\rho[\mathbb{P}]}{\overset{\tilde{\rho}[\mathbb{P}]}{\rightleftarrows}} \mathcal{P}(S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}])$$ vérifiant les hypothèses suivantes

(1) $\forall E \in \mathcal{P}(S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}])$,
$\underline{\text{True}} \in \mathcal{A}[\mathbb{P}]$ et $\rho[\mathbb{P}](E) \Rightarrow \text{True}$.

(2) $\forall P \in \mathcal{A}[\mathbb{P}]$,
$\tilde{\rho}[\mathbb{P}](P) \subset S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$.

(3) $\tilde{\rho}[\mathbb{P}] \circ \rho[\mathbb{P}] \subset 1$

(4) $1 \Rightarrow \rho[\mathbb{P}] \circ \tilde{\rho}[\mathbb{P}]$.

(5) Pour toute famille $(A_\alpha)_{\alpha \leq \alpha_0}$ de $\mathcal{P}(S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}])$ telle que $\alpha_0 < \omega_1$,
$\rho[\mathbb{P}](\bigcup_{\alpha \leq \alpha_0} A_\alpha) \Rightarrow \bigvee_{\alpha \leq \alpha_0} \rho[\mathbb{P}](A_\alpha)$

(6) Pour toute famille $(P_\alpha)_{\alpha \leq \alpha_0}$ de $\mathcal{A}[\mathbb{P}]$ telle que $\alpha_0 < \omega_1$,
$\tilde{\rho}[\mathbb{P}](\bigvee_{\alpha \leq \alpha_0} P_\alpha) \subset \bigcup_{\alpha \leq \alpha_0} \tilde{\rho}[\mathbb{P}](P_\alpha)$.

On remarquera que $\rho[\mathbb{P}]$ et $\tilde{\rho}[\mathbb{P}]$ sont croissantes toutes deux pour les ordres partiels $\subset$ et $\Rightarrow$.

Ceci revient à exiger une certaine condition d'expressivité du langage $\mathcal{A}[\mathbb{P}]$. On définit la notion d'assertions de $\mathcal{A}[\mathbb{P}]$ comme celle de formules. La notion de validité change quelque peu.

Définition [I.2].5 :

Soit $\mathbb{P}$ un programme, $\mathcal{A}[\mathbb{P}]$ un langage d'assertions.

On dit que A est vraie en s de $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$, si

$s \in \tilde{\rho}[\mathbb{P}](A)$ et on note $A(s)$.

Les fonctions sont supposées toujours exister mais une question intéressante serait de savoir si de telles fonctions et donc de tels langages existent toujours.

Nous supposerons disposer de tels langages dans le cadre de notre étude.

Soit $T^n_{\mathcal{U}[\mathbb{P}]}$ l'ensemble des suites finies d'éléments de $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ tels que :

$\langle s_1 \ldots s_p \rangle \in T^n_{\mathcal{U}[\mathbb{P}]}$ ssi $t_n[\mathbb{P}](s_1,s_2) \wedge \ldots \wedge t_n[\mathbb{P}](s_{p-1},s_p)$.

Théorème [I.2].4 :

Soit $\mathbb{P}$ un programme, $\mathcal{U}[\mathbb{P}]$ une structure abstraite de $\mathbb{P}$,

$S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ un ensemble d'états de $\mathbb{P}$.

$T^n_{\mathcal{U}[\mathbb{P}]}$ est inductif sur $\mathcal{U}[\mathbb{P}]$.

Preuve :

$S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ est inductif sur $\mathcal{U}[\mathbb{P}]$,

$t^*_n[\mathbb{P}]$ est inductif sur $\mathcal{U}[\mathbb{P}]$.

Soit Γ l'opérateur défini par la formule :

$\varphi(\bar{x},S) \equiv Sv(\exists s \in S, S(s) \wedge t_n^*[\mathbb{P}](s,\bar{x}))$

$\bar{x} \in S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$.

On définit ainsi $I_\varphi = I_\Gamma$; on en déduit que $T^n_{\mathcal{U}[\mathbb{P}]}$ est inductif sur $\left(\mathcal{U}[\mathbb{P}], S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}], t_n^*[\mathbb{P}]\right)$.

Soit sur $\mathcal{U}[\mathbb{P}]$, par transitivité. □

Nous donnons quelques rappels relatifs aux mots sur un alphabet donné. Ces rappels seront nécessaires pour la suite de notre exposé dans ce chapitre et dans les prochains.

Définition [I.2].6 :

Soit E un ensemble non-vide.

1 - On appelle mot sur E toute application w de $\mathbb{N}^+$ dans E telle que $dom(w) = \mathbb{N}^+$ ou $dom(w) = \{1,\ldots,i\}$ pour un i de $\mathbb{N}^+$. $dom(w)$ est appelé le domaine de w et i la longueur de w.

2 - $|w|$ est la longueur de w. Si $|w|$ est finie, on dit que w est un mot fini et sinon on dit que w est infinie.

3 - L'ensemble de tous les mots sur E est noté, E^∞. L'ensemble des mots finis est noté E^* et celui des mots infinis E^ω : $E^\infty = E^\omega \cup E^*$.

4 - Etant donnés deux mots w et w' sur E.

$w \leqslant w'$ ssi $(dom(w) \subset dom(w')) \wedge (\forall i \leqslant |w|,\ w(i) = w'(i))$.

5 - Si w est un mot sur E et $i \in dom(w)$, $j \in dom(w)$, $i < j$, $w[i,j[$ est le mot sur E tel que $dom(w[i,j[) = [j-i+1]$ et

$\forall k \in [j-i+1]$, $(w[i,j[)(k) = w(i+k-1)$.

On note ε le mot vide.

$T^n_{\mathcal{U}[\mathbb{P}]}$ est un sous-ensemble de l'ensemble des mots sur $S^n_{\mathcal{U}[\mathbb{P}]}$: c'est-à-dire $T^n_{\mathcal{U}[\mathbb{P}]} \subset \left(S^n_{\mathcal{U}[\mathbb{P}]}\right)^\infty$. La relation d'ordre partiel $\leqslant$ sur $T^n_{\mathcal{U}[\mathbb{P}]}$ est élémentaire sur $T^n_{\mathcal{U}[\mathbb{P}]}$ et inductive sur $\mathcal{U}[\mathbb{P}]$. On note T^n, un sous-ensemble quelconque de $T^n_{\mathcal{U}[\mathbb{P}]}$ et T^n_s le sous-ensemble de $T^n_{\mathcal{U}[\mathbb{P}]}$ dont les mots commencent par s où s est élément de $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$. Nous considèrons un langage d'assertions $\mathcal{A}[\mathbb{P}]$ et, à l'aide des notions ci-dessus, nous définissons la relation de fatalité relativement à un sous-ensemble T^n de $T^n_{\mathcal{U}[\mathbb{P}]}$.

Définition [I.2].7 :

Soient P et Q deux assertions de $\mathcal{A}[\mathbb{P}]$;

$S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ l'ensemble des états de $\mathbb{P}$ de nombre n ;

T^n un ensemble de traces de $\mathbb{P}$ de nombre n.

On dit que P conduit fatalement à Q sur T^n, si l'énoncé suivant est vrai :

$$\Big\{\forall s \in S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}], P(s) \Rightarrow \Big[(\forall w)(w \in T^n_s), (\exists w')(w' \in T^n_s), [(w \leqslant w') \wedge (\exists i)\{(i \in dom(w')) \wedge (Q(w'(i)))\}]\Big].$$

On note : $T^n \models P \rightsquigarrow Q$.

Etant donné une assertion Q de $\mathcal{A}[\mathbb{P}]$, on définit la classe de toutes les assertions P de $\mathcal{A}[\mathbb{P}]$ conduisant fatalement à Q sur T^n et on la note $[T^n. \rightsquigarrow Q]$. On munit cette classe de l'ordre partiel $\Rightarrow$.

Théorème [I.2].5 :

Soit $\mathbb{P}$ un programme, $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ un ensemble d'états de $\mathbb{P}$ sur $\mathcal{U}[\mathbb{P}]$ une structure abstraite de $\mathbb{P}$, T^n tout l'ensemble $T^n_{\mathcal{U}[\mathbb{P}]}$, $\mathcal{A}[\mathbb{P}]$ un langage d'assertions pour $\mathbb{P}$.

(1) $([T^n.\rightsquigarrow Q], \Rightarrow)$ a un sup noté $\bar{Q}$.

(2) Si le langage de $\mathcal{U}[\mathbb{P}]$ est de type $\mathcal{L}_{\alpha\beta}$, $\alpha \geqslant \kappa^{\mathcal{U}[\mathbb{P}]}$, $\beta \geqslant \omega$, pour toute assertion P de $[T^n.\rightsquigarrow Q]$, il existe un ordinal $0(P,Q)$ plus petit que $\kappa^{\mathcal{U}[\mathbb{P}]}$ et un opérateur inductif $\Gamma_{T^n}[Q]$ sur $S^n_{\mathcal{U}[\mathbb{P}]}$ tels que :

- $I_{\Gamma_{T^n}[Q]} = \tilde{\rho}[\mathbb{P}]\ (\underset{\Rightarrow}{\mathrm{Sup}}([T^n.\rightsquigarrow Q]))$
- $P \Rightarrow \rho[\mathbb{P}]((I_{\Gamma_{T^n}[Q]})^{0(P,Q)})$.
- $0(P,Q) \leqslant \|\Gamma_{T^n}[Q]\| \leqslant \kappa^{\mathcal{U}}$.
- si on suppose que $t_n[\mathbb{P}]$ est dénombrable, alors quel que soit le langage $\mathcal{L}_{\alpha\beta}$, $\|\Gamma_{T^n}[Q]\| < \omega_1$, pourvu que $\alpha \geqslant \omega_1$.

Preuve :

Soit $\Gamma_{T^n}[Q]$ l'opérateur sur $S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}]$ suivant :

$$\Gamma_{T^n}[Q](S) = S \cup \{s \in S^n_{\mathcal{U}[\mathbb{P}]}[\mathbb{P}] / [\forall w \in T^n_s,$$
$$\exists w' \in T^n_s, (w \leqslant w') \wedge (w'(2) \in S)] \vee Q(s)\}.$$

Tous les éléments de $\Gamma_{T^n}[Q]$ sont inductifs sur $\mathcal{U}[\mathbb{P}]$; par transitivité, on déduit que $\Gamma_{T^n}[Q]$ est un opérateur inductif sur $\mathcal{U}[\mathbb{P}]$; Γ_{T^n} est donc tel que $\|\Gamma_{T^n}\| < \kappa^{\mathcal{U}[\mathbb{P}]}$, d'après le théorème [I.1].8.

Soit $P_0 = \rho[\mathbb{P}]\ (I_{\Gamma_{T^n}[Q]})$. $P_0 \in \mathcal{A}[\mathbb{P}]$ et par construction de $I_{\Gamma_{T^n}[Q]}$, $P_0 \in [T^n.\rightsquigarrow Q]$.

Supposons qu'il existe P_1 dans $[T^n.\rightsquigarrow Q]$ tel que $P_0 \Rightarrow P_1$ et $P_1 \not\Rightarrow P_0$. Ceci revient à écrire :

$\tilde{P}_0 \subset \tilde{P}_1$ et $\tilde{P}_1 \not\subset \tilde{P}_0$ où $\tilde{P}_i = \tilde{\rho}[\mathbb{P}](P_i)$, $i=0,1$.

Soit $s_1 \in \tilde{P}_1$ tel que $s_1 \notin \tilde{P}_0$.

$\forall w \in T^n_{s_1}$, $\exists w' \in T^n_{s_1}$, $(w \leqslant w') \wedge (\exists i \in \mathrm{dom}(w'), Q(w'(i)))$.

Soit $w \in T^n_{s_1}$ et $w' \in T^n_{s_1}$ tel que $w \leqslant w'$ et $\exists i \in \mathrm{dom}(w'), Q(w'(i))$.

Supposons que $Q(w'(i_1))$. Pour tout état $w'(j)$ tel que $j \in \{1,\ldots,i\}$, $w'(j) \notin \tilde{P}_0$, pour tout $i \leqslant i_1$.

sinon $w(1) \in \tilde{P}_0$.

On a prouvé qu'il existe $s \in Q$ tel que $s \notin \tilde{P}_0$.

Or, $\{s\} \rightsquigarrow Q$, ce qui est en contradiction avec l'hypothèse. $I_{\Gamma_{T^n}[Q]}$ est maximal.

Soit $P \in [T^n.\rightsquigarrow Q]$. Pour tout état s de $I_{\Gamma_{T^n}[Q]}$, il existe un plus petit ordinal $\alpha(s)$ tel que

$$s \in \left(I_{\Gamma_{T^n}[Q]}\right)^{\alpha(s)} \text{ et } s \notin \left(I_{\Gamma_{T^n}[Q]}\right)^{\beta}, \forall \beta < \alpha(s).$$

Posons $0(P,Q) = \mathrm{Sup}(\alpha(s)/P(s))$.

Par contradiction, $0(P,Q) \leqslant \|\Gamma_{T^n}[Q]\|$.

Supposons que $t[\mathbb{P}]$ soit dénombrable :

$|\{s' \in S^n_{[\mathbb{P}]}[\mathbb{P}] / t_n[\mathbb{P}](s,s')\}| \leqslant \omega$.

Soit $\|\Gamma_{T^n}[Q]\| \geqslant \omega_1$. Il existe $u \subset \left(I_{\Gamma_{T^n}[Q]}\right)^{\omega_1}$ tel que $\forall \alpha < \omega_1$, $u \notin \left(I_{\Gamma_{T^n}[Q]}\right)^{\alpha}$. Par construction de $I_{\Gamma_{T^n}[Q]}$,

$$\left\{\begin{array}{l} t_n[\mathbb{P}](u,u') \\ u' \in I_{\Gamma_{T^n}[Q]} \end{array}\right\} \Rightarrow \{u' \in I_{\Gamma_{T^n}[Q]}{}^{\beta} \wedge (\beta < \omega_1)\}.$$

Soit γ_u l'ordinal suivant :

$\gamma_u = \mathrm{Sup}(\beta / t_n[\mathbb{P}](u,u') \wedge (u' \in \Gamma^{\beta}_{T^n}[Q]))$.

- $\gamma_u < \omega_1$ car β est dénombrable et $t_n[\mathbb{P}]$ est aussi dénombrable.

Or par définition des ordinaux dénombrables (cf ROGERS [ROG]) γ_u est dénombrable.

On en déduit que $\gamma_u = \omega_1$, puisque ω_1 était le plus petit ordinal pour u. Donc $\omega_1 < \omega_1$, ce qui est absurde. On en déduit le résultat suivant

$$\| \Gamma_{T^n}[Q] \| < \omega_1 . \ \square$$

Ce théorème ci-dessus est à rapprocher des résultats obtenus par APT et PLOTKIN [APP] au sujet du non-déterminisme non borné et, en particulier, dénombrable. Nous pensons que notre résultat est plus adéquat au problème d non-déterminisme dénombrable que ceux de APT et PLOTKIN [APP] qui se restrei à des structures abstraites telles que l'ordinal est récursif (ie : noté κ par APT et PLOTKIN [APP]) : toutes les relations et fonctions sont récursives et les domaines sont dénombrables. Notre résultat ne suppose rien à p du nombre cardinal de successeurs de chaque élément pour cette relation de transition. Cependant, les résultats de APT et PLOTKIN [APP] ne concernant qu'une certaine classe de propriétés de fatalité qui sont la terminaison de programmes séquentiels non-déterministes, alors que nous considérons une pr priété quelconque de fatalité. Dans un premier temps, nous voulions abstrai au maximum la notion de fatalité sous hypothèse d'équité faible, en ne mentionnant l'hypothèse d'équité faible que par le seul choix d'un ensemble, maximal pour cette propriété, de traces d'exécution et en donnant une version généralisée du théorème précédent pour toute partie de $T^n_{\mathcal{U}[\mathbb{P}]}$.

Une remarque critique et pertinente de APT, me permit de comprendre que le problème de la fatalité sous hypothèse d'équité faible exigeait une abstraction moins forte au niveau de la notion de programme : la notion d'action atomique n'est pas prise en compte explicitement par la notion de structure abstraite de $\mathbb{P}$, notée $\mathcal{U}[\mathbb{P}]$ et les assertions de contrôle associées à chaque action atomique ne sont pas spécifiées dans le langage d'assertions $\mathcal{A}[\mathbb{P}]$. Nous avons préféré donc reporter le résultat relatif à la fatalité sous hypothèse d'équité faible et ces quelques préliminaires au chapitre suivant. Nous noterons dans le théorème précédent le rôle joué par le langage $\mathcal{L}_{\alpha\beta}$.

Nous n'avons pu construire un opérateur monotone pour chaque assertion d'un langage d'assertions d'un programme donné sous hypothèse d'équité faible.

Ceci nous contraint à spécifier la classe des programmes concernés et la notion d'action atomique ; notre prochaine étude consistera à généraliser le théorème précédent pour le cas où l'ensemble des traces est restreint à celles obéissant à l'hypothèse d'équité faible. Nous considèrerons des relations de transition dénombrables par la suite et ceci nous imposera un langage infinitaire de type $\mathcal{L}_{\omega_1\omega}$ mais nous reviendrons sur ce problème en temps opportun.

II - PROPRIÉTÉS DE FATALITÉ DES PROGRAMMES PARALLÈLES

II - PROPRIÉTÉS DE FATALITÉ DES PROGRAMMES PARALLÈLES

Les résultats précédents sont maintenant généralisés à l'étude des programmes parallèles. Dans le chapitre précédent les relations et les notions abstraites avaient un sens lié à l'exécution du programme ; cependant il convient de présenter formellement la classe des programmes. De plus certaines notions comme action atomique, assertion de contrôle nous seront, par la suite, très utiles.

Cette étude veut définir la notion d'équité faible, en utilisant la notion de structure sémantique de fatalité qui sera explicitée. Le point de vue adopté par OWICKI et LAMPORT [OWL] exprimait par la logique temporelle linéaire les notions de fatalité et d'équité faible ; ils utilisaient comme modèle d'interprétation, un ensemble de suites d'exécution noté Σ ; les notions d'assertions d'états et d'assertions de suite d'états nous paraissent confuses et cette confusion si claire dans leur esprit aboutit dans l'article de KUIPER et DE ROEVER [KUD] a une méconnaissance dangereuse des deux notions : ainsi, ils expriment la notion d'équité faible pour une affectation atomique $x:=t$ par $\Box(\underline{at}\ x:=t) \Rightarrow \Diamond(\text{after } x:=t)$ et ceci n'est pas correct car le $\Box$ porte sur les suites et $\Diamond$ sur les états, sinon cela signifie que $\underline{at}\ x:=t$ et $\underline{after}\ x:=t$ peuvent se produire en même temps. On peut corriger cette notation en utilisant $\Box((\underline{at}\ x:=t) \Rightarrow \Diamond(\underline{after}\ x:=t))$. Nous préférons oublier ces notations modales et étudier les propriétés de fatalité sous hypothèse d'équité faible par le biais d'opérateurs et d'ordinaux.

II.1.- SYNTAXE DES PROGRAMMES PARALLELES

On désigne par $\mathcal{V}$, un ensemble dont les éléments sont appelés variables, $\mathcal{B}$, un ensemble dont les éléments sont appelés expressions booléennes, $\mathcal{E}$, un ensemble dont les éléments sont appelés expressions, $\mathcal{D}$, un ensemble dont les éléments sont appelés valeurs. Nous supposons que $\mathcal{V}$ est un ensemble dénombrable, ce qui nous paraît être suffisant et réaliste pour notre étude.

Dans la suite on désignera un élément de $\mathcal{V}$ par les lettres $v, v_1, v_2, \ldots$, un élément de $\mathcal{B}$ par $B, B_1, B_2, \ldots$, un élément de $\mathcal{E}$ par $e, e_1, \ldots$, un élément de $\mathcal{D}$ par $d, d_1, \ldots$.

Intuitivement, un programme parallèle est un ensemble de programmes séquentiels non-déterministes partageant des variables communes. Nous donnons formellement la définition des programmes séquentiels non-déterministes et ensuite des programmes parallèles.

Définition [II.1].1 :

La classe des programmes séquentiels non-déterministes, notée $\mathcal{P}$, est la plus petite classe telle que :

- pour tout élément v de $\mathcal{V}$, pour chaque élément e de $\mathcal{E}$, $[v:=e]$ et $[v:=?]$ sont des éléments de $\mathcal{P}$.
- $[\underline{skip}]$ est élément de $\mathcal{P}$.

et $\mathcal{P}$ est stable par application finie des règles suivantes :

- Règle 1 : si $\mathbb{P}_1 \in \mathcal{P}$, $\mathbb{P}_2 \in \mathcal{P}$, $B \in \mathcal{B}$,
  alors $[\underline{if}\ B\ \underline{then}\ \mathbb{P}_1\ \underline{else}\ \mathbb{P}_2\ \underline{fi}] \in \mathcal{P}$.
- Règle 2 : si $\mathbb{P} \in \mathcal{P}$, $B \in \mathcal{B}$,
  alors $[\underline{while}\ B\ \underline{do}\ \mathbb{P}\ \underline{od}] \in \mathcal{P}$.

- Règle 3 : si $\mathbb{P}_1 \in \mathcal{P}$, $\mathbb{P}_2 \in \mathcal{P}$,
  alors $[\mathbb{P}_1 ; \mathbb{P}_2] \in \mathcal{P}$.

Définition [I.1].2 :

La classe des programmes parallèles, notée $\mathcal{C}$, est la plus petite classe contenant $\mathcal{P}$ et stable par application de la règle suivante :

- Règle 4 : si $\mathbb{P}_1, \ldots, \mathbb{P}_n \in \mathcal{P}$, $n \in \mathbb{N}$, $n \geq 1$, alors
  $[\underline{Cobegin}\ \mathbb{P}_1 \| \ldots \| \mathbb{P}_n\ \underline{coend}] \in \mathcal{C}$.

Afin de localiser les différentes parties de chaque objet de $\mathcal{P}$ ou $\mathcal{C}$ ie le contrôle, nous définissons une relation d'étiquetage notée lab et l'ensemble des étiquettes associées à $\mathcal{P}$ ou $\mathcal{C}$: on note $\mathcal{L}$ un ensemble d'étiquettes, dénombrable.

On notera l'équivalence syntaxique de deux objets O et O' par l'expression $O \equiv O'$.

(1) si $\ell, \ell' \in \mathcal{L}$, $\ell \neq \ell'$, $\mathbb{P} \equiv [Inst]$ où Inst est v:=e, v:=?, skip,
alors $\underline{lab}(\mathbb{P}) \equiv [\ell:Inst;\ell':]$, $\mathcal{L}[\mathbb{P}] = \{\ell, \ell'\}$, $end(\mathbb{P}) = \ell'$.

(2) si $\ell, \ell' \in \mathcal{L}$, $\ell \neq \ell'$, $\mathbb{P} \equiv [\underline{if}\ B\ \underline{then}\ \mathbb{P}_1\ \underline{else}\ \mathbb{P}_2\ \underline{fi}]$, $\mathbb{P}_1 \in \mathcal{P}$, $\mathbb{P}_2 \in \mathcal{P}$,
$\ell \notin \mathcal{L}[\mathbb{P}_1] \cup \mathcal{L}[\mathbb{P}_2]$, $\ell' \notin \mathcal{L}[\mathbb{P}_1] \cup \mathcal{L}[\mathbb{P}_2]$, $\mathcal{L}[\mathbb{P}_1] \cap \mathcal{L}[\mathbb{P}_2] = \emptyset$,
alors $lab(\mathbb{P}) \equiv [\ell:\underline{if}\ B\ \underline{then}\ \underline{lab}(\mathbb{P}_1)\underline{else}\ \underline{lab}(\mathbb{P}_2)\underline{fi};\ell']$,
$\mathcal{L}[\mathbb{P}] = \{\ell, \ell'\} \cup \mathcal{L}[\mathbb{P}_1] \cup \mathcal{L}[\mathbb{P}_2]$,
$end(\mathbb{P}) = \ell'$.

(3) si $\ell, \ell' \in \mathcal{L}$, $\ell \neq \ell'$, $\mathbb{P} \equiv [\underline{while}\ B\ \underline{do}\ \mathbb{P}'\ \underline{od}]$, $\mathbb{P}' \in \mathcal{P}$,
$\ell \notin \mathcal{L}[\mathbb{P}']$, $\ell' \notin \mathcal{L}[\mathbb{P}']$,
alors $\underline{lab}(\mathbb{P}) \equiv [\ell:\underline{while}\ B\ \underline{do}\ \underline{lab}(\mathbb{P}')\underline{od};\ell':]$,
$\mathcal{L}[\mathbb{P}] = \{\ell, \ell'\} \cup \mathcal{L}[\mathbb{P}']$,
$end(\mathbb{P}) = \ell'$.

④ si $\mathbb{P} \equiv [\mathbb{P}_1;\mathbb{P}_2], \mathbb{P}_1 \in \mathcal{P}$, $\mathbb{P}_2 \in \mathcal{P}$, $\mathcal{L}[\mathbb{P}_1] \cap \mathcal{L}[\mathbb{P}_2] = \emptyset$,
alors $\underline{lab}(\mathbb{P}) \equiv \alpha\ lab(\mathbb{P}_2)$ où $(\alpha;\ell' :\equiv lab(\mathbb{P}_1))$ et $\ell' = end(\mathbb{P}_1)$,
$\mathcal{L}[\mathbb{P}] = \mathcal{L}[\mathbb{P}_1] \setminus \{end(\mathbb{P}_1)\} \cup \mathcal{L}[\mathbb{P}_2]$,
$end(\mathbb{P}) = end(\mathbb{P}_2)$.

⑤ si $\mathbb{P} \equiv [\underline{Cobegin}\ \mathbb{P}_1 \| \ldots \| \mathbb{P}_n\ \underline{coend}], (\forall i \in \{1,\ldots,n\}, \mathbb{P}_i \in \mathcal{P})$,
$\ell, \ell' \in \mathcal{L}, \ell \neq \ell', \ell \in \mathcal{L} \setminus \bigcup_{i=1}^{n} \mathcal{L}[\mathbb{P}_i], \ell' \in \mathcal{L} \setminus \bigcup_{i=1}^{n} \mathcal{L}[\mathbb{P}_i]$,
$\forall (i,j) \in \{1,\ldots,n\}^2, ((i \neq j) \Rightarrow \mathcal{L}[\mathbb{P}_i] \cap \mathcal{L}[\mathbb{P}_j] = \emptyset)$,
alors $\underline{lab}(\mathbb{P}) \equiv [\ell:\underline{Cobegin}\ \underline{lab}(\mathbb{P}_1) \| \ldots \| \underline{lab}(\mathbb{P}_n)\underline{coend};\ell':]$,
$\mathcal{L}[\mathbb{P}] = \{\ell,\ell'\} \cup \bigcup_{i=1}^{n} \mathcal{L}[\mathbb{P}_i]$,
$end(\mathbb{P}) = \ell'$.

Définition [II.1].3 :

Soit $\mathbb{P}$ un objet de la classe $\mathcal{C}$. $\mathcal{L}[\mathbb{P}]$ est l'ensemble des étiquet associées à $\mathbb{P}$ avec $\underline{lab}$.

Comme $\mathbb{P}$ est fini, $\mathcal{L}[\mathbb{P}]$ et $\underline{lab}(\mathbb{P})$ satisfaisant les règles 1, 2, 4, 5 existent toujours. La condition faite sur $\mathcal{L}$ est toujours satisfaite, on suppose que $\mathcal{L}$ est infini dénombrable ; on peut toujours choisir une éti quette dans la partie restante de $\mathcal{L}$ car elle est infinie.

Définition [II.1].4 :

On appelle programme parallèle tout objet de la classe $\mathcal{C}$. Pour chaque programme parallèle $\mathbb{P}$ de $\mathcal{C}$, $\mathcal{V}[\mathbb{P}]$ est l'ensemble des variables de $\mathbb{P}$.

II.2.- SEMANTIQUE OPERATIONNELLE DES PROGRAMMES PARALLELES

Dans ce paragraphe nous noterons $C\mathbb{P}$ un programme parallèle quelconqu

et $\mathbb{P}S$ un programme séquentiel non déterministe quelconque. Nous associons une sémantique opérationnelle d'abord à $\mathbb{P}S$ puis nous l'utilisons pour en associer à $\mathbb{P}$. $\mathcal{D}[\mathbb{P}S]$ désigne le domaine des variables de $\mathbb{P}S$.

Définition [II.2].1 :

Un état s de $\mathbb{P}S$ est un couple formé d'une étiquette ℓ de $\mathcal{L}[\mathbb{P}S]$ et d'une application de $\mathcal{V}[\mathbb{P}S]$ dans $\mathcal{D}[\mathbb{P}S]$ ie :

$s = (\ell,m)$ et $\ell \in \mathcal{L}[\mathbb{P}S]$, $m \in [\mathcal{V}[\mathbb{P}S] \to \mathcal{D}[\mathbb{P}S]]$.

On note $S[\mathbb{P}S]$ l'ensemble des états de $\mathbb{P}S$ et il est équivalent à $\mathcal{L}[\mathbb{P}S] \times [\mathcal{V}[\mathbb{P}S] \to \mathcal{D}[\mathbb{P}S]]$. Il nous reste à définir une relation de transition $t[\mathbb{P}S]$ sur $S[\mathbb{P}S]$ qui va modéliser l'exécution de $\mathbb{P}S$.

Avant de donner la relation de transition $t[\mathbb{P}S]$, nous donnons quelques notions préliminaires. Pour chaque variable v de $\mathbb{P}S$ on définit D_v comme étant le domaine des valeurs de v et dans ce cas $\mathcal{D}[\mathbb{P}S] = \bigcup_{v \in \mathcal{V}[\mathbb{P}S]} D_v$.

Pour chaque application m de $\mathcal{V}[\mathbb{P}S]$ dans $\mathcal{D}[\mathbb{P}S]$ on définit $m[e/v]$ comme étant l'application de $\mathcal{V}[\mathbb{P}S]$ dans $\mathcal{D}[\mathbb{P}S]$ telle que :

$\forall v' \in \mathcal{V}[\mathbb{P}S], (v' \neq v) \Rightarrow m[e/v](v') = m(v')$

$(v' = v) \Rightarrow m[e/v](v') = e$

Si B est une expression booléenne de $\mathbb{P}S$, alors, $B[m]$ désigne la valeur booléenne de B pour m où m évalue les valeurs des variables v de $\mathbb{P}S$: $B[m] \in [\mathcal{D}[\mathbb{P}S]] \to \{T,F\}$ où $\{T,F\}$ est l'ensemble des valeurs booléennes $\underline{true}$ et $\underline{false}$.

Définition [II.2].2 :

Soient s,s' deux états quelconques de $S[\mathbb{P}S]$.

$s = (\ell,m)$, $s' = (\ell',m')$ où $\ell,\ell' \in \mathcal{L}[\mathbb{P}S]$, $m,m' \in \mathcal{V}[\mathbb{P}S] \to \mathcal{D}[\mathbb{P}S]$.

Alors $t[\mathbb{PS}]$ est une relation binaire définie par :

$t[\mathbb{PS}](s,s')$ si, et seulement si,

$[(\underline{lab}(\mathbb{PS})\equiv\alpha\ell:v:=e;\ell':\beta)\wedge(m'=m[e/_v])]\vee$

$[(\underline{lab}(\mathbb{PS})\equiv\alpha\ell:v:=?;\ell':\beta)\wedge(\bigvee_{x\in D_v}(m'=m[x/_v]))]\vee$

$[(\underline{lab}(\mathbb{PS})\equiv\alpha\ell:\underline{skip};\ell':\beta)\wedge(m=m')]\vee$

$[(\underline{lab}(\mathbb{PS})\equiv\alpha\ell;\underline{if}\ B\ \underline{then}\ \ell_1:\alpha_1\underline{else}\ \ell_2:\alpha_2\ \underline{fi};\beta)\wedge(m=m')\wedge$

$\{(B[m]\wedge(\ell'=\ell_1))\vee(\sim B[m]\wedge(\ell'=\ell_2))\}]\vee$

$[\{(\underline{lab}(\mathbb{PS})\equiv\alpha:\underline{if}\ B\ \underline{then}\ \alpha_1;\ell:\underline{else}\ \alpha_2;\ell_2:\underline{fi};\ell':\beta)\vee$

$(\underline{lab}(\mathbb{PS})\equiv\alpha:\underline{if}\ B\ \underline{then}\ \alpha_1;\ell_1:\underline{else}\ \alpha_2;\ell:\underline{fi};\ell':\beta)\}\wedge(m=m')]\vee$

$[(\underline{lab}(\mathbb{PS})\equiv\alpha\ell:\underline{while}\ B\ \underline{do}\ \ell_1:\alpha_1\ \underline{od};\ell_2:\beta)\wedge(m=m')\wedge$

$\{(B[m]\wedge(\ell'=\ell_1))\vee(\sim B[m]\ (\ell'=\ell_2))\}]\vee$

$[(\underline{lab}(\mathbb{PS})\equiv\alpha:\underline{while}\ B\ \underline{do}\ \ell_1:\alpha_1;\ell:\underline{od};\ell_2:\beta)\wedge(m=m')\wedge$

$\{(B[m]\wedge(\ell'=\ell_1))\vee(\sim B[m]\wedge(\ell'=\ell_2))\}]$

Définition [II.2].3 :

Le couple $(S[\mathbb{PS}], t[\mathbb{PS}])$ est appelé sémantique opérationnelle de $\mathbb{P}$

Nous étendons cette notion aux cas de programmes parallèles. Ainsi nous modélisons l'exécution de tout programme parallèle CIP en mélangeant de façon non déterministe les transitions.

Modélisant l'exécution de chaque programme séquentiel composant CIP en supposant qu'à tout instant de l'exécution de CIP un seul programme séquentiel peut être activé, nous supposons que les problèmes de synchronisation et d'exclusion mutuelle sont résolus par programmation, et non pas d'après la sémantique du programme. Cependant aucune hypothèse d'équité n'est faite à ce niveau.

Définition [II.2].4 :

Un état de CIP $(CIP \in \mathcal{C})$ est un couple formé d'un état de contrôle $\bar{\ell}$ et d'un état des variables m tel que :

- $\bar{\ell} \in \{\underline{\ell}_1,\bar{\ell}_2\}\cup\prod_{i=1}^{n}\mathcal{L}[\mathbb{PS}_i]$ où $\underline{\ell}_1,\bar{\ell}_2 \in \mathcal{L}[CIP]\setminus\bigcup_{i=1}^{n}\mathcal{L}[\mathbb{PS}_i]$.

- m est une application de $\mathcal{V}[CIP]$ dans $\mathcal{D}[CIP]$ telle que $\mathcal{V}[CIP] = \mathcal{V}[\mathbb{PS}_i]$, $\forall i \in \{1,\ldots,n\}$.

On note $S[CIP]$ l'ensemble des états et il est équivalent à

$$[\{\underline{\ell}_1,\bar{\ell}_2\}\cup\prod_{i=1}^{n}\mathcal{L}[\mathbb{PS}_i]]\times[\mathcal{V}[CIP]\to\mathcal{D}[CIP]].$$

Si nous avons fait l'hypothèse $\mathcal{V}[CIP] = \mathcal{V}[\mathbb{PS}_i]$, c'était pour exprimer le fait que $\mathbb{PS}_i$ partagent ses variables avec les autres $\mathbb{PS}_j$: ceci revient à étendre l'ensemble des variables de chaque $\mathbb{PS}_i$ de façon artificielle. On peut maintenant définir une relation de transition $t[CIP]$ sur $S[CIP]$.

Définition [II.2].5 :

Soient s,s' deux éléments quelconques de $S[CIP]$:

$s=(\bar{\ell},m)$, $s'=(\bar{\ell}',m')$.

$t[CIP](s,s')$ si, et seulement si,

$[(\underline{lab}(CIP)\equiv[\underline{\ell}_1:\underline{Cobegin}\ \ell_1:\alpha_1[]\ldots[]\ell_n:\alpha_n\ \underline{coend};\bar{\ell}_2:])\wedge$

$(\bar{\ell}=\underline{\ell}_1)\wedge(m=m')\wedge(\bar{\ell}'=(\ell_1,\ldots,\ell_n))]\vee$

$[(\underline{lab}(CIP)\equiv[\underline{\ell}_1:\underline{Cobegin}\ \alpha_1;\ell^1:[]\ldots[]\alpha_n;\ell^n:\underline{coend};\bar{\ell}_2])\wedge$

$(\bar{\ell}=(\ell',\ldots,\ell^n))\wedge(m=m')\wedge(\bar{\ell}_2=\bar{\ell}')]\vee$

$[(\underline{lab}(CIP)\equiv[\underline{\ell}_1:\underline{Cobegin}\ \alpha_1;\ell_1:\beta_1;\ell_1':[]\ldots[]\alpha_n;\ell_n:\beta_n;\ell_n':\underline{coend};\bar{\ell}_2:])\wedge$

$(\bar{\ell}\neq\underline{\ell}_1)\wedge(\bar{\ell}\neq\bar{\ell}_2)\wedge(\bar{\ell}\neq(\ell_1',\ldots,\ell_n'))\wedge(\bar{\ell}=(\ell_1,\ldots,\ell_n))\wedge$

$(\exists i\in\{1,\ldots,n\},\{t[\mathbb{PS}_i]((\ell_i,m),(\ell_i'',m'))\wedge(\forall j\in[1,\ldots,n]\setminus\{i\},(\ell_j=\ell_j''))\wedge$

$(\bar{\ell}'=(\ell_1'',\ldots,\ell_n''))\})]$.

Définition [II.2].6 :

Le couple (S[CIP], t[CIP]) est appelé sémantique opérationnelle de

L'hypothèse d'équité ne peut être exprimée directement à l'aide de la sémantique opérationnelle ; ceci nécessite l'introduction d'une notion permettant de définir rigoureusement la fatalité. Dans cette définition, on suppose que les variables ne sont pas modifiées par effet de bord.

II.3.- ACTIONS ATOMIQUES DE CIP

Nous précisons dans cette partie le "grain" de l'action que nous utilis La sémantique opérationnelle permet de discrétiser l'exécution de CIP. Intuitivement, les actions atomiques de CIP sont les tests atomiques et les affectations atomiques. Nous ajoutons à cet ensemble quatre autres types d'actions atomiques : Start, Finish, Out-if1, Out-if2. Les deux premières permettent respectivement de simuler le passage du contrôle depuis le début de CIP ver le début de chaque programme séquentiel composant CIP et le passage du contrôle depuis la fin de chaque programme séquentiel composant CIP vers la fi du cobegin. Les deux suivantes simulent la fin d'une partie du if suivant q l'on se trouve dans la branche B positive ou B négative. La définition de la sémantique opérationnelle aurait pu explicitement donner l'action atomique exécutée lors d'une transition simple entre deux états de CIP.

Formellement nous définissons l'ensemble des actions atomiques de CIP, noté A[CIP], par induction sur la syntaxe.

Définition [II.3].1

L'ensemble des actions atomiques de CIP, A[CIP], est construit à partir des règles suivantes :

(1) si CIP∈ {[v:=e],[skip],[v:=?]},then A[CIP] = {CIP}.

(2) si CIP ≡ [if B then CIP_1 else CIP_2 fi], then
A[CIP] = {<B>,Out-if1, Out-if2}∪A[CIP_1]∪A[CIP_2]

(3) si CIP ≡ [CIP_1 ;CIP_2], alors A[CIP] = A[CIP_1]∪A[CIP_2].

(4) si CIP ≡ [while B do CIP ' od], alors A[CIP]={<B>}∪A[CIP '].

(5) si CIP ≡ [Cobegin $CIP_1 \| \ldots \| CIP_n$ coend], alors
$A[CIP]=\{Start, Finish\} \cup \bigcup_{i=1}^{n} A[CIP_i]$.

On note <B> pour signifier que le test B est atomique.

Les actions atomiques seront nécessaires pour définir plus loin la notion d'équité faible.

Définition [II.3].2 :

Soit S une partie quelconque de CIP.

AV(S) est le sous-ensemble de S[CIP] défini par

$(s \in AV(S))$ ssi $(s=((\ell_1,\ldots,\ell_n),m)) \wedge (\underline{lab}(CIP)\equiv \alpha;\ell:S\beta) \wedge (\exists i\in[n], \ell=\ell_i)$.

AP(S) est le sous-ensemble de S[CIP] défini par

$(s \in AP(S))$ ssi $(s=((\ell_1,\ldots,\ell_n),m)) \wedge (\underline{lab}(CIP)\equiv \alpha;S;\ell:\beta) \wedge (\exists i\in[n], \ell=\ell_i)$.

En utilisant les ensembles AV(S) et AP(S), nous définissons pour chaque action atomique a de A[CIP] les sous-ensembles de S[CIP], notés AV(a) et AP(a), par :

$s \in AV(a)$ ssi $[\{(S=a)\wedge(a\in\{v:=e;v:=?,skip\})\wedge(C\mathbb{P}\equiv\alpha S\beta)\} \Rightarrow \{s\in AV(S)\}]\wedge$

$[\{(a=Start)\wedge(C\mathbb{P}\equiv \underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend})\} \Rightarrow \{s\in AV(C\mathbb{P})\}]\wedge$

$[\{(a=Finish)\wedge(C\mathbb{P}\equiv \underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend})\} \Rightarrow$
$\{s\in AP(\mathbb{P}S_1)\cap\ldots\cap AP(\mathbb{P}S_n)\}]\wedge$
$[\{(a=Out\text{-}if1)\wedge(C\mathbb{P}\equiv\alpha \underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}\ \beta)\wedge$
$(a\in A[\underline{if}]\setminus(A[S_1]\cup A[S_2]))\} \Rightarrow \{s\in AP(S_1)\}]\wedge$

$[\{(a=Out\text{-}if2)\wedge(C\mathbb{P}\equiv\alpha\ \underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}\ \beta)\wedge$
$(a\in A[\underline{if}]\setminus(A[S_1]\cup A[S_2]))\} \Rightarrow \{s\in AP(S_2)\}]\wedge$

$[\{(a=<B>)\wedge(S\equiv \underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi})\wedge(C\mathbb{P}\equiv\alpha S\beta)\}\Rightarrow\{s\in AV(S)\}]\wedge$

$[\{(a=<B>)\wedge(S\equiv \underline{while}\ B\ \underline{do}\ S'\ \underline{od})\wedge(C\mathbb{P}\equiv\alpha S\beta)\}\Rightarrow\{s\in AV(S)\cup AP(S')\}]$.

Si $s \in AV(a)$, cela signifie intuitivement que le contrôle est juste av
a et que a peut être exécuté ultérieurement au cours de l'exécution. De la
même façon, on peut exprimer le fait d'être après une action atomique et ceci
est l'objet de la définition suivante.

$s \in AP(a)$ ssi $[\{(S=a)\wedge(a\in\{v:=e;v:=?,\underline{skip}\})\wedge(C\mathbb{P}\equiv\alpha S\beta)\} \Rightarrow \{s\in AP(S)\}]\wedge$

$[\{(a=Start)\wedge(C\mathbb{P}\equiv \underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend})\} \Rightarrow$
$\{s\in AV(\mathbb{P}S_1)\cap\ldots\cap AV(\mathbb{P}S_n)\}]\wedge$
$[\{(a=Finish)\wedge(C\mathbb{P}\equiv \underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend})\} \Rightarrow \{s\in AP(C\mathbb{P})\}]\wedge$

$[\{(a\in\{Out\text{-}if1,Out\text{-}if2\})\wedge\ (C\mathbb{P}\equiv\alpha\ \underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}\ \beta)\wedge$

$(a\in A[\underline{if}]\setminus(A[S_1]\cup A[S_2]))\} \Rightarrow \{s\in AP(\underline{if})\}]\wedge$

$[\{(a=<B>)\wedge(S\equiv\underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2)\wedge(C\mathbb{P}\equiv\alpha S\beta)\} \Rightarrow$

$\{s\in AV(S_1)\cup AV(S_2)\}]\wedge$
$[\{(a=<B>)\wedge(S\equiv\underline{while}\ B\ \underline{do}\ S'\ \underline{od})\wedge\ (C\mathbb{P}\equiv\alpha S\beta)\} \Rightarrow$

$\{s\in AV(S')\cup AP(S)\}]$.

Nous pouvons alors donner une relation entre la sémantique opérationnelle et la sémantique de chaque action atomique a de $C\mathbb{P}$:

Pour tout a de $A[C\mathbb{P}]$, pour tout état s,s' de $S[C\mathbb{P}]$,

$t[a](s,s')$ ssi $(s \in AV(a))\wedge(s' \in AP(a))\wedge t[C\mathbb{P}](s,s')$.

Les actions atomiques caractérisent $t[C\mathbb{P}]$ par :

$t[C\mathbb{P}](s,s')$ ssi $\exists a \in A[C\mathbb{P}]$, $t[a](s,s')\wedge(s \in AV(a))\wedge(s' \in AP(a))$.

On note t[a] la relation de transition définie pour a et elle est à comparer au modèle utilisé par MANNA et PNUELI [MAP1, MAP2, MAP3] notamment. L'approche structurée nous paraît la plus intéressante mais nous aurons besoin

par la suite de faire référence à une action atomique. Notons que, au niveau des ensembles AV, AP, il y a certaines relations d'inclusion qui permettront de passer d'une approche à l'autre modulo la sémantique opérationnelle. La preuve du dernier énoncé nous paraît simple et laborieuse et nous préférons la laisser de côté.

II.4.- STRUCTURE SEMANTIQUE DE FATALITE

A l'aide de la sémantique opérationnelle de CIP et des notions relatives aux mots sur un alphabet, nous introduisons un cadre capable de prendre en compte la notion de fatalité sous certaine(s) hypothèse(s).

Définition [II.4].1 :

Soient E un ensemble non-vide et r une relation binaire définie sur E. L'ensemble des traces sur E engendré par r est le sous-ensemble de E^∞ noté T(E,r), défini par

$\forall w \in E^\infty$, $(w \in T(E,r)) \Leftrightarrow$

$[(|w| = 0) \wedge (w = \varepsilon)] \vee$

$[(|w| = 1) \wedge (w \in E)] \vee$

$[(\forall i < |w|, r(w(i), w(i+1))) \wedge (|w| \geq 2)]$.

Définition [II.4].2 :

Soit CIP un programme parallèle ($CIP \in \mathcal{C}$),

(S[CIP],t[CIP]) sa sémantique opérationnelle.

On appelle structure sémantique de fatalité de CIP tout triplet (S[CIP],t[CIP],T) où T est une partie de T(S[CIP],t[CIP]), noté T[CIP].

Notre définition est restreinte au cas de programmes parallèles mais reste valable pour d'autres couples (E,r). La notion de trace est utilisée pour définir correctement le caractère arborescent du temps ; cette notion est considérée par BEN-ARI, MANNA et PNUELI [BMP] sous le nom de chemin et une axiomatique complète est donnée. Nous ne considérons pas la notion de possibilité qui est plus faible que la fatalité, son étude a un intérêt philosophique respectable mais n'a que peu d'importance pour le cas des programmes.

II.5.- PROPRIETES DE FATALITE

Notre but ici est de lier les notions du I.1 et les notions des paragraphes précédents, afin de formuler correctement la notion d'équité et de fatalité.

Etant donné un programme parallèle CIP de $\mathcal{C}$. On lui associe une structure abstraite $\mathcal{U}$[CIP] telle que toute transition élémentaire est présente sous forme d'une relation. Ceci revient à associer à toute action atomique de CIP une relation. On suppose de plus avoir construit de façon inductive la structure sémantique de fatalité associée à CIP comme il a été fait au chapitre I. Ces opérations ont été réalisées à l'aide d'un langage que nous noterons $\mathcal{L}^{CIP}$.

Soient $\mathcal{A}$[CIP] et $\tilde{\mathcal{A}}$[CIP] tels qu'il existe deux fonctions d'abstraction (*) ρ[CIP] et $\tilde{\rho}$[CIP] entre ces deux langages d'assertion pour CIP.

On suppose que $\tilde{\mathcal{A}}$[CIP] est l'ensemble des parties de S[CIP].

(*) et de concrétisation.

Définition [II.5].1 :

On appelle modèle de fatalité pour $\mathcal{A}[CIP]$ un objet $\mathcal{M}$ tel que :

{pour toutes assertions P,Q de $\mathcal{A}[CIP]$,

$\mathcal{M} \models P \rightsquigarrow Q$ (ie : $\mathcal{M}$ valide $P \rightsquigarrow Q$)}

si, et seulement si,

{pour toutes assertions $\tilde{P}, \tilde{Q}$ de $\tilde{\mathcal{A}}[CIP]$,

$(S[CIP],t[CIP],T) \models \tilde{P} \rightsquigarrow \tilde{Q}$ où $T \subset T[CIP]$}.

Dans cette définition, $(S[CIP],t[CIP],T) \models \tilde{P} \rightsquigarrow \tilde{Q}$ est équivalente à $T \models \tilde{P} \rightsquigarrow \tilde{Q}$ définie au chapitre I. Ainsi, si $\mathcal{M}$ est un modèle de fatalité pour $\mathcal{A}[CIP]$, on étend les fonctions d'abstraction et de concrétisation aux traces de CIP et nous définissons maintenant la notion de fatalité relative à un modèle de fatalité $\mathcal{M}$.

Définition [II.5].2 :

Soit CIP un programme parallèle (ie : CIP $\in \mathcal{C}$),

$\mathcal{A}[CIP]$ un langage d'assertions pour CIP,

$\mathcal{M}$ un modèle de fatalité pour $\mathcal{A}[CIP]$,

$\rho[CIP],\tilde{\rho}[CIP]$ les deux fonctions comme ci-dessus,

$\mathcal{U}[CIP]$ la structure abstraite associée à CIP,

P,Q deux assertions de $\mathcal{A}[CIP]$,

$\mathcal{M}$ un modèle de fatalité pour CIP.

On dit que $\mathcal{M}$ valide $P \rightsquigarrow Q$ ou que P conduit fatalement à Q pour $\mathcal{M}$, si $\tilde{\rho}[CIP](\mathcal{M}) \models \tilde{P} \rightsquigarrow \tilde{Q}$ où $\tilde{P} = \tilde{\rho}[CIP](P)$ et $\tilde{Q} = \tilde{\rho}[CIP](Q)$.

Remarque : La structure sémantique de fatalité est un modèle pour le langage $\mathcal{P}(S[CIP])$.

Cette définition est à rapprocher de la logique temporelle du temps arborescent proposée par BEN-ARI, MANNA et PNUELI [BMP] et en fait nous avons "caché" l'opérateur "$\forall\exists$". La définition donnée par OWICKI et LAMPORT [OWL] est équivalente car elle porte sur les suites d'exécution. Nous pensons que la logique temporelle est un outil intéressant pour spécifier de telle relation mais elle nous semble manquer d'expressivité et nous ne citerons que l'exemple de WOLPER [WOL] qui augmente l'expressivité de cette logique, afin de prendre en compte plus de phénomènes. Notre langage est suffisamment général pour ne pas souffrir des mêmes inconvénients mais nous allons le spécifier davantage notamment par les assertions de contrôle.

Remarquons que les assertions de $\mathcal{A}[CIP]$ sont interprétées naturellement par les fonctions d'abstraction et de concrétisation,

$P \in \mathcal{A}[CIP]$: $s \in S[CIP]$,

$P(s)$ ssi $s \in \tilde{\rho}[CIP](P)$.

Définition [II.5].3 :

Soit CIP un programme parallèle,

S une partie de CIP ie une instruction,

a une action atomique de CIP

1 - at $S \equiv \rho[CIP](AV(S))$.

2 - after $S \equiv \rho[CIP](AP(S))$.

3 - at $a \equiv \rho[CIP](AV(a))$.

4 - after $a \equiv \rho[CIP](AP(a))$.

5 - jafter $a \equiv \rho[CIP](\{s \in S[CIP] / \exists \underline{s} \in S[CIP], (\text{at } a)(\underline{s}) \wedge$
$(\forall s' \in S[CIP], (t^*[CIP](\underline{s},s') \wedge t^+[CIP](s',s))$
$\Rightarrow (\text{at } a)(s')) \wedge (\text{after } a)(s)\})$.

6 - $\underline{jat}\ a \equiv \rho[CIP](\{s \in S[CIP]/[(a=Start) \Rightarrow (\underline{at}\ a)(s)] \wedge$

$[\{(a \neq Start)\} \Rightarrow \{(\exists b \in A[CIP], (b \neq a)$

$\wedge(\underline{jafter}\ b)(s) \wedge (\underline{at}\ a)($

$(\underline{at}\ a \equiv \underline{after}\ b)(s))]$

A l'aide des assertions ci-dessus, nous définissons les assertions jat et jafter pour le cas d'une instruction de CIP.

Définition [II.5].4 et Propriété :

1 - $S \in \{[v:=e, v:=?, skip\}$ et $a \equiv S$

$\underline{jat}\ S \equiv \underline{jat}\ a$, $\underline{jafter}\ S \equiv \underline{jafter}\ a$

2 - $S \equiv S_1;S_2$

$\underline{jat}\ S \equiv \underline{jat}\ S_1$, $\underline{jafter}\ S_1 \equiv \underline{jat}\ S_2$, $\underline{jafter}\ S \equiv \underline{jafter}\ S_2$.

3 - $S \equiv \underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}$

$\underline{jat}\ S \equiv \underline{jat}<B>$, $\underline{jat}\ S_1 \equiv \underline{jafter}<B> \wedge \underline{at}\ S_1$,

$\underline{jat}\ S_2 \equiv \underline{jafter}<B> \wedge \underline{at}\ S_2$,

$\underline{jafter}\ S^{(1)} \equiv (\underline{jafter}\ S) \wedge \underline{jafter}\ S_1$,

$\underline{jafter}\ S^{(2)} \equiv (\underline{jafter}\ S) \wedge \underline{jafter}\ S_2$

4 - $S \equiv \underline{while}\ B\ \underline{do}\ S'\ \underline{od}$

$\underline{jat}\ S \equiv \underline{jat}<B> \wedge \underline{at}\ S$,

$\underline{jafter}\ S \equiv \underline{jafter}<B> \wedge \underline{after}\ S$.

$\underline{jat}\ S' \equiv \underline{jafter}<B> \wedge \underline{at}\ S'$,

$\underline{jafter}\ S' \equiv \underline{jat}<B> \wedge \underline{after}\ S'$,

5 - $S \equiv \underline{Cobegin}\ \mathbb{P}S_1 [] \ldots [] \mathbb{P}S_n\ \underline{coend}$

$\underline{jat}\ S \equiv \underline{at}\ S$

$\underline{jafter}\ S \equiv \underline{after}\ S$

$\underline{jat}\ \mathbb{P}S_1 \wedge \ldots \wedge \underline{jat}\ \mathbb{P}S_n \equiv \underline{at}\ \mathbb{P}S_1 \wedge \ldots \wedge \underline{at}\ \mathbb{P}S_n \wedge \underline{jafter}$ Start.

Les assertions définies ci-dessus sont appelées assertions de contrôle de CIP ; OWICKI et LAMPORT [OWL] ne disposaient que des assertions de contrôle $\underline{at}\ S$ et $\underline{after}\ S$ et nous avons ajouté $\underline{jat}$ et $\underline{jafter}$, afin de pouvoir exprimer le fait que l'on venait d'exécuter à l'instant précédent telle ou telle autre instruction atomique. Ces deux assertions supplémentaires nous sont indispensables pour garantir la complétude sémantique de notre système de preuve et nous noterons au passage la complémentarité de ces deux assertions. Une illustration ultérieure de l'utilité de ces assertions sera faite plus tard. Nous allons maintenant appliquer les résultats du chapitre précédent à notre cas particulier de programme.

Théorème [II.5].1 :

Soit CIP un programme parallèle,

$\mathcal{A}[CIP]$ un langage d'assertions pour CIP,

$\mathcal{M}$ un modèle de fatalité sur $\mathcal{A}[CIP]$ tel que

$\mathcal{M} \underset{\rho[CIP]}{\overset{\tilde{\rho}[CIP]}{\rightleftarrows}} (S[CIP], t[CIP], T[CIP])$

$\mathcal{U}[CIP]$ une structure abstraite pour CIP

Q une assertion de $\mathcal{A}[CIP]$.

Pour toute assertion P de $\mathcal{A}[CIP]$ telle que $\mathcal{M} \models P \rightsquigarrow Q$, il existe un opérateur monotone Γ sur $S[CIP]$ tel qu'il existe un ordinal, noté $O(P,Q)$, vérifiant :

- $O(P,Q) \leqslant \|\Gamma\| \leqslant \kappa^{\mathcal{U}[C\mathbb{P}]}$
- $\forall s \in S[C\mathbb{P}], \left(P \Rightarrow \rho[C\mathbb{P}]\left(\Gamma^{O(P,Q)}\right)\right)(s)$.
- $\forall s \in S[C\mathbb{P}], (Q \Leftrightarrow \rho[C\mathbb{P}](\Gamma^{\circ}))(s)$.
- $\forall \beta < O(P,Q), \exists s \in S[C\mathbb{P}], P(s) \wedge \sim \rho[C\mathbb{P}](\Gamma^{\beta})$.

Preuve :

Elle est immédiate en appliquant le théorème [I.2].5 et la définition de la fatalité selon un modèle de fatalité $\mathcal{M}$. En fait Γ dépend de $\mathcal{M}$ et non pas de P. On en déduit de même le résultat suivant :

Théorème [II.5].2 :

On suppose les mêmes hypothèses que le théorème [I.2].1. On ajoute l'hypothèse suivante : $t[C\mathbb{P}]$ est dénombrable.

Alors, pour toute assertion P de $\mathcal{A}[C\mathbb{P}]$ telle que $\mathcal{M} \models P \rightsquigarrow Q$, il existe un opérateur monotone Γ sur $S[C\mathbb{P}]$ tel qu'il existe un ordinal $O(P,Q)$ vérifiant :

- $O(P,Q) = \|\Gamma\|$.
- $O(P,Q) < \omega_1$.
- $\forall s \in S[C\mathbb{P}], \left(P \Rightarrow \rho[C\mathbb{P}]\left(\Gamma^{O(P,Q)}\right)\right)(s)$,
- $\forall s \in S[C\mathbb{P}], \left(Q \Leftrightarrow \rho[C\mathbb{P}](\Gamma^{\circ})\right)(s)$.
- $\forall \beta < O(P,Q), \exists s \in S[C\mathbb{P}], P(s) \wedge \sim \rho[C\mathbb{P}](\Gamma^{\beta})$.

Preuve :

On applique ici le théorème [I.2].5 qui précise que l'ordinal de fermeture est dénombrable dans ce cas. □

Notre étude se limite à l'étude du non-déterminisme dénombrable et l'hypothèse faite sur $t[C\mathbb{P}]$ est suffisante dans le cadre de notre étude. Il serait intéressant de généraliser à des cardinalités régulières comme $\omega_2, \omega_3, \ldots$ (cf COUSOT-COUSOT [COC2]). Nous pensons, come nous l'avons déjà précisé, que l'aspect dénombrable du non-déterminisme est mis en évidence, alors que APT et PLOTKIN [APP], qui donnent un résultat plus fin avec d'autres hypothèses, montrent que l'on peut se suffire des ordinaux récursifs (ie : $<\kappa<\omega_1$). Cependant, la notion de récursivité implique le dénombrable et dans ce cas il s'agit de non-déterminisme récursif. Ces deux derniers théorèmes sont de simples application du théorème [I.2].5 et donnent une caractérisation des règles d'induction et principes d'induction que l'on utilise au cours de preuves de propriétés de fatalité sans hypothèse. Ces résultats reprennent les méthodes de terminaison de FLOYD [FLO] et de HOARE [HOA] et il nous faut envisager intuitivement que, si on ajoute l'hypothèse d'équité faible, l'utilisation d'ensembles bien fondés ou d'ordinaux est nécessaire, ce qui revient à construire un opérateur monotone Γ comme nous l'avons fait ci-dessus.

II.6.- HYPOTHESE D'EQUITE FAIBLE

A l'aide de la notion de modèle de fatalité, nous caractérisons l'hypothèse d'équité faible : toute action atomique prête à être déclenchée attendra un temps fini arbitraire sa prise en compte par le "scheduler". APT et OLDEROG [APO] caractérisent la notion d'équité faible au niveau de la validité par le biais de transformation syntaxique ; nous pensons que cette approche est intéressante mais préconise plus ou moins une implantation. Intuitivement, ils considèrent une propriété de terminaison à la HOARE comme $\{P\}\ \mathbb{P}\ \{Q\}$ sous hypothèse d'équité faible et montrent que ceci est équivalent à

$\{P\}T_{WF}(\mathbb{P})\{Q\}$ sans hypothèse où $T_{WF}(\mathbb{P})$ est la transformation faiblement équitable de $\mathbb{P}$. Nous choisissons une autre solution plus sémantique qui est la restriction de l'ensemble des traces aux traces faiblement équitables ce qui revient à définir un modèle de fatalité faiblement équitable.

Définition [II.6].1 :

Soit $C\mathbb{P}$ un programme parallèle,

$\mathcal{A}[C\mathbb{P}]$ un langage d'assertions pour $C\mathbb{P}$,

$A[C\mathbb{P}]$ l'ensemble des actions atomiques de $C\mathbb{P}$,

$\mathcal{M}$ un modèle de fatalité pour $C\mathbb{P}$.

On dit que $\mathcal{M}$ est faiblement équitable, si, pour toute action a de $A[C\mathbb{P}]$,

$$\mathcal{M} \models \underline{\text{at}}\ a \rightsquigarrow \underline{\text{jafter}}\ a.$$

Parmi les modèles de fatalité faiblement équitables, on note $\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable, le plus grand au sens de l'inclusion modulo les fonctions $\rho[C\mathbb{P}]$ et $\tilde{\rho}[C\mathbb{P}]$ pour les ensembles de traces. Ce choix revient à considérer le plus grand ensemble de traces vérifiant l'hypothèse de la définition [II.6].1. On notera T, le plus grand ensemble de traces, associé à $\mathcal{M}_{WF}$ par les fonctions $\rho[C\mathbb{P}]$ et $\tilde{\rho}[C\mathbb{P}]$.

A l'aide de $\mathcal{M}_{WF}$, nous pouvons définir formellement la notion de fatalité sous hypothèse d'équité faible.

Définition [II.6].2 :

Soient $C\mathbb{P}$ un programme parallèle,

$\mathcal{A}[C\mathbb{P}]$ un langage d'assertions pour $C\mathbb{P}$,

P,Q deux assertions de $\mathcal{A}[C\mathbb{P}]$.

On dit que P conduit fatalement à Q sous hypothèse d'équité faible dans $C\mathbb{P}$, si $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

Nous étudions maintenant la propriété de fatalité sous hypothèse d'équité faible à l'aide d'opérateur monotone approprié, afin de généraliser le théorème [I.2].5 pour le cas où le modèle de fatalité est $\mathcal{M}_{WF}$.

Théorème [II.6].1 :

Soient $C\mathbb{P}$ un programme parallèle,

$\mathcal{A}[C\mathbb{P}]$ un langage d'assertions pour $C\mathbb{P}$,

$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable pour $C\mathbb{P}$,

Q une assertion de $\mathcal{A}[C\mathbb{P}]$.

a - Il existe un opérateur monotone Γ_Q sur $S[C\mathbb{P}]$ tel que :

1 - Pour toute assertion P telle que $\tilde{\rho}[C\mathbb{P}](P) \subset I_{\Gamma_Q}$, $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

2 - Pour toute assertion P telle que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, $\tilde{\rho}[C\mathbb{P}](P) \subset I_{\Gamma_Q}$.

b - Pour toute assertion P de $\mathcal{A}[C\mathbb{P}]$ telle que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, il existe une famille de multiensembles M_P telle que

1 - $M_P = \{a(s)/P(s)$ est vraie$\}$ et a(s) est un multiensemble formé d'actions atomiques de $C\mathbb{P}$.

2 - $\forall s \in S[C\mathbb{P}], (P(s) \wedge (a(s) \neq \emptyset)) \Rightarrow (\exists a \in a(s), (\underline{\text{at}}\ a)(s))$

et $\forall s \in S[C\mathbb{P}], (P(s) \wedge (a(s) = \emptyset)) \Rightarrow Q(s)$

3 - $\forall s \in S[C\!P], \forall a \in a(s)$,

$[(\underline{at}\ a)(s) \wedge t^*[C\!P](s,s') \wedge (\underline{at}\ a)([s,s'[) \wedge (\underline{jafter}\ a)(s')] \Rightarrow$

$[(a(s') \subsetneq a(s)) \wedge (\forall w \in T_{s'}, \exists w \in T_{s'}, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i))))]$

Avant de commencer la preuve, nous remarquerons que les actions atomiques de $a(s)$ sont critiques car elles permettent de faire progresser la fatalité faiblement équitable. La relation $\subset$ est une relation bien fondée, ce que nous recherchons en fait dans ce type de problème (cf. [FLO]). L'opérateur Γ est un opérateur d'élagage des branches non équitables de l'arbre d'exécution. Nous donnons maintenant la preuve du théorème.

Preuve :

Notre preuve est en quatre points, qui sont les suivants :

(1) Construction d'un opérateur monotone Γ_Q.

(2) Preuve du point a-1 du théorème.

(3) Preuve du point a-2 du théorème.

(4) Preuve du point b du théorème.

(1) Construction de Γ_Q

L'opérateur Γ_Q est défini sur $S[C\!P]$ et nous supposons tout au long de la preuve que deux fonctions d'abstraction et de concrétisation permettent de raisonner tantôt sur les assertions, tantôt sur des parties d'ensembles, considérées comme des assertions mais plus intuitives. Nous notons Z la variable de Γ_Q utilisée. Soit Γ_Q l'opérateur suivant $Z \in \mathcal{P}(S[C\!P])$:

$$\Gamma_Q(Z) = \{s \in S[C\!P] / [Q(s)]$$

$$\vee \Big(\exists a \in A[C\!P], \{\exists s' \in S[C\!P], (\underline{at}\ a)(s) \wedge t[C\!P](s,s') \wedge (\underline{jafter}\ a)(s') \wedge (s' \in Z)\}$$

$$\wedge \Big[\forall s'' \in S[C\!P], \big[t^*[C\!P](s,s'') \wedge (\underline{at}\ a)([s,s"]) \big] \Rightarrow \big[\exists s' \in S[C\!P], (\underline{at}\ a)(s") \wedge t[C\!P](s",s') \wedge (\underline{jafter}\ a)(s') \wedge (s' \in Z) \big] \Big] \Big)$$

$$\wedge\ \forall s' \in S[C\!P], \{t[C\!P](s,s')\}$$

$$\Rightarrow$$

$$\Big[\exists a \in A[C\!P], \Big[\exists s'' \in S[C\!P], (\underline{at}\ a)(s') \wedge t[C\!P](s',s") \wedge (\underline{jafter}\ a)(s") \wedge (s" \in Z) \Big]$$

$$\wedge \Big[\forall s'' \in S[C\!P], \big\{ t^*[C\!P](s',s") \wedge (\underline{at}\ a)([s',s"] \big\} \Rightarrow \big\{ \exists u \in S[C\!P], (\underline{at}\ a)(s") \wedge t[C\!P](s",u) \wedge (\underline{jafter}\ a)(u) \wedge (u \in Z) \big\} \Big]$$

$$\wedge \Big[\forall a \in A[C\!P], (\underline{at}\ a)(s') \Rightarrow (\forall w \in T_{s'}, \exists w' \in T_{s'}, (w \leq w') \wedge (\exists i \in dom(w'), (\underline{jafter}\ a)(w'(i)))) \Big]$$

$$\vee\ [Q(s')]$$

$$\wedge\ (\forall a \in A[C\!P], (\underline{at}\ a)(s) \Rightarrow (\forall w \in T_s, \exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), (\underline{jafter}\ a)(w'(i)))))$$

Dans la définition de Γ_Q, nous avons utilisé une notation permettant d'exprimer le fait que le contrôle reste devant une action atomique entre deux états inclus :

$$(\underline{at}\ a)([s,s'])\wedge t^*[s,s'] \text{ ssi } \forall s'' \in S[CI\!P], \quad (t^*[CI\!P](s,s'')\wedge t^*[CI\!P](s'',s')) \Rightarrow (\underline{at}\ a)(s'')$$

T_s désigne l'ensemble des traces faiblement équitables commençant par s où $s \in S[CI\!P]$.

Soient X,Y deux parties quelconques de $S[CI\!P]$ telles que $X \subseteq Y$.

L'opérateur Γ_Q peut s'écrire plus simplement sous la forme :

$$\forall Z \in \mathcal{P}(S[CI\!P]),\ \Gamma_Q(Z) = \{s \in S[CI\!P]/Q(s) \vee C(Z)(s)\}$$

où $C(Z)(s)$ est la relation vérifiée par s dans Γ_Q.

Soit $s \in \Gamma_Q(X)$; alors $Q(s)$ ou $C(X)(s)$ sont vraies.

Si $Q(s)$ est vraie, alors $s \in \Gamma_Q(Y)$.

Si $C(X)(s)$ est vraie, alors, par la forme de $C(X)$, $C(Y)(s)$ est vraie puisque $X \subseteq Y$.

On en déduit : $s \in \Gamma_Q(Y)$.

Γ_Q est un opérateur monotone sur $S[CI\!P]$.

Mais avant de l'utiliser, il faut montrer les points (2) et (3), afin de prouver que Γ_Q vérifient les hypothèses du théorème à démontrer.

(2) Soit P une assertion de $\mathcal{A}[CI\!P]$ telle que $\tilde{\rho}[CI\!P](P) \subset I_{\Gamma_Q}$.

Puisque Γ_Q est un opérateur monotone sur $S[CI\!P]$, il existe un ordinal dit de clôture, noté $\Gamma_Q^{\|\Gamma_Q\|}$, tel que :

(1) $- \forall \alpha > \|\Gamma_Q\|,\ \Gamma_Q^{\alpha} = \Gamma_Q^{\|\Gamma_Q\|} = I_{\Gamma_Q}$.

(2) $- \|\Gamma_Q\| \leqslant \kappa^{\mathcal{U}[CI\!P]}$ où $\mathcal{U}[CI\!P]$ est la structure abstraite de référence.

(3) $- \forall s \in I_{\Gamma_Q},\ \exists \alpha(s) \leqslant \|\Gamma_Q\|,\ \{\forall \beta < \alpha(s), (s \notin \Gamma_Q^{\beta})\}\wedge\{\forall \beta \geqslant \alpha(s), (s \in \Gamma_Q^{\beta})\}$

a) Soit s un état de $\tilde{\rho}[CI\!P](P)$.

Par hypothèse, $s \in I_{\Gamma_Q}$ et par (3), on en déduit qu'il existe $\alpha(s)$ tel que $\alpha(s) \leqslant \|\Gamma_Q\|$ et $(\forall \beta < \alpha(s), (s \notin \Gamma_Q^{\beta}))\wedge(\forall \beta \geqslant \alpha(s), (s \in \Gamma_Q^{\beta}))$.

On fait une preuve par induction transfinie, pour montrer que s conduit fatalement à Q sous hypothèse d'équité faible.

a-1.- Cas où $\alpha(s) = 0$

Par définition de $\alpha(s)$, cela signifie que s valide Q.

On en déduit que s conduit fatalement à Q.

a-2.- Cas où $\alpha(s) > 0$

Dans ce cas, $\alpha(s) \leqslant \|\Gamma_Q\|$ par convergence de Γ_Q et on suppose que :

$$\forall s' \in S[CI\!P], [\alpha(s') < \alpha(s)] \Rightarrow [\forall w \in T_{s'}, \exists w' \in T_{s'}, (w \leqslant w')\wedge(\exists i \in dom(w'), Q(w'(i)))]$$

Par définition de $\alpha(s)$, $s \in \Gamma_Q^{\alpha(s)}$ et par construction de Γ_Q, s ne valide pas Q, puisque $\alpha(s) > 0$ et

$$\exists a \in A[CI\!P], [\exists s' \in S[CI\!P], (\underline{at}\ a)(s)\wedge t[CI\!P](s,s')\wedge(\underline{jafter}\ a)(s')\wedge(s' \in \Gamma_Q^{\beta})\wedge(\beta\wedge\alpha) \wedge[\forall s'' \in S[CI\!P], \{t^*[CI\!P](s,s'')\wedge(\underline{at}\ a)([s,s'])\} \Rightarrow \{\exists s' \in S[CI\!P], t[CI\!P](s'',s')\wedge(\underline{at}\ a)(s'')\wedge(\underline{jafter}\ a)(s')\wedge (s' \in \Gamma_Q^{\beta})\wedge(\beta<\alpha)\}]$$

où $\alpha = \alpha(s)$.

Soit w une trace commençante par s et faiblement équitable pour $CI\!P$.

Cas 1 : $|w| = 1$

Par hypothèse faite sur s, toute action atomique prête à être exécutée le sera fatalement.

On en déduit qu'il existe w' de T_s tel que : pour toute action atomique a de $C\!I\!P$,

$(w \leq w') \wedge \exists i \in dom(w'), \underline{at}\ a(w'[1,i[) \wedge (\underline{jafter}\ a)(w'(i))$.

Puisque $s \in \Gamma_Q^\alpha$, il existe $a \in A[C\!I\!P]$ qui est prête en s et qui vérifie :

$\forall s'' \in S[C\!I\!P], \{t^*[C\!I\!P](s,s'') \wedge (\underline{at}\ a)([s,s'[) \wedge (\underline{jafter}\ a)(s'')\} \Rightarrow \{(s'' \in \Gamma_\beta) \wedge (\beta < \alpha)$

Il existe une trace w' commençant par s tel que, pour cette action atomique a de $C\!I\!P$:

$(w \leq w') \wedge \Big(\exists i \in dom(w'), (\underline{at}\ a)([w'[1,i-1]) \wedge (\underline{jafter}\ a)(w'(i)) \wedge (w'(i) \in \Gamma_\beta) \wedge (\beta <$

On a prouvé que :

(i) $\forall w \in T_s, (|w|=1) \Rightarrow \Big(\exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), (w'(i) \in \Gamma^\beta) \wedge (\beta < \alpha)\Big)$

Par hypothèse, si $w'(i) \in \Gamma^\beta$, pour $\beta < \alpha$, alors, $\alpha(w'(i)) < \alpha$, et

(j) $\forall v \in T_{w'(i)}, \exists v' \in T_{w'(i)}, (v \leq v') \wedge (\exists j \in dom(v'), Q(v'(j)))$.

On en déduit en combinant (i) et (j) que :

$\forall w \in T_s, (|w|=1) \Rightarrow (\exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i))))$.

Cas 2 : $1 < |w| < +\infty$

Cas 2.1 : $\exists i \in dom(w)$, $Q(w(i))$.

Cas 2.2 : $\forall i \in dom(w), \sim Q(w(i))$.

Par hypothèse, $w(1) \in \Gamma_Q^\alpha$.

Cas 2.2.1 : $\exists i \in dom(w)$, $w(i) \in \Gamma_Q^\beta$.

On utilise alors l'hypothèse :

$w(i)...w(|w|)$ est un mot commençant par $w(i)$ et élément de T (ensemble des traces faiblement équitables) car sinon w ne serait pas élément de T.

$\exists w' \in T_{w(i)}, (w(i)...w(|w|) \leq w') \wedge (\exists i \in dom(w'), Q(w'(i)))$.

On considère alors : $w'' = w(1)...w(i-1).w'$, w'' vérifie l'hypothèse suivante :

$(w \leq w'') \wedge (\exists j \in dom(w''), Q(w''(j)))$.

Cas 2.2.2 : $\forall i \in dom(w), w(i) \notin \Gamma_Q^\beta$.

Dans ce cas, chaque $w(i)$ est un élément de Γ_Q^α par construction de Γ_Q : en effet, à partir de s, toute transition conduit à un état où les actions atomiques activables en s et non activées en s, le sont encore en le successeur de s par $t[C\!I\!P]$ et dans notre cas $w(2)$ n'est pas dans Γ_Q^β pour $\beta < \alpha$, donc une action atomique reste encore activable en $w(2)$. On poursuit ce raisonnement pour chaque i de $dom(w)$.

Puisque $w(|w|)$ est un élément de Γ_Q^α, on applique le cas 1 :

$\exists w' \in T_{w(|w|)}, ((w(|w|) \leq w'') \wedge (\exists i \in dom(w''), Q(w''(i)))$.

Soit $w' \in T_s$ tel que $w' = w.w''$. On vérifie que

$(\exists i \in dom(w'), Q(w'(i))) \wedge (w \leq w'))$.

Cas 3 : $|w| = +\infty$.

Cas 3.1 : $\exists i \in dom(w)$, $Q(w(i))$.

Dans ce cas, $w \leq w$ et w vérifie la propriété.

Cas 3.2 : $\forall i \in dom(w), \sim Q(w(i))$

Or par définition de mot sur un alphabet donné, le seul mot plus grand que w est w. Puisque $s \in \Gamma^{\alpha}$, il existe $a \in A[CIP]$ vérifiant la propriété décrite au début de (2) et :

$(\underline{at}\ a)(s)$ est vraie, ainsi que $\exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), (\underline{jafter}\ a)($

On en déduit que $w' = w$ et que

$\exists i \in dom(w), (\underline{jafter}\ a)(w(i))$.

Par propriété de l'action a, on déduit la propriété :

$\exists i \in dom(w), (\underline{jafter}\ a)(w(i)) \wedge (w(i) \in \Gamma^{\beta}) \wedge (\beta < \alpha)$.

On déduit alors par hypothèse de récurrence que

$\exists i \in dom(w), (w(i) \in Q)$.

Ce qui contredit l'hypothèse faite sur w.

On a prouvé que s conduit à Q sous hypothèse d'équité faible pour chaque $\alpha(s) > 0$, en supposant cela pour s' tel que $\alpha(s') < \alpha(s)$, donc, pour chaque s de I_{Γ_Q}, s conduit fatalement à Q sous hypothèse d'équité faible.

b) Soit P de $\mathcal{A}[CIP]$ vérifiant $\tilde{\rho}[CIP](P) \subset I_{\Gamma_Q}$.

La propriété démontrée dans a) nous permet de généraliser pour les assertions P :

$\forall s \in S[CIP], P(s) \Leftrightarrow s \in \tilde{\rho}[CIP](P)$ (par définition de la validité).

$\forall s \in S[CIP], P(s) \Rightarrow (s \in I_{\Gamma_Q})$ (par hypothèse sur P).

$\forall s \in S[CIP], (s \in I_{\Gamma_Q}) \Rightarrow \forall w \in T_s, \exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i)))$ (par a)).

On en déduit :

$\forall s \in S[CIP], P(s) \Rightarrow (\forall w \in T_s, \exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i))))$.

Ce que l'on écrit encore :

$\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

(3) Soit P une assertion telle que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

Il suffit de raisonner sur un état s tel que

$\forall w \in T_s, \exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i)))$.

L'opérateur Γ_Q permet de construire l'arbre d'exécution de CIP sous hypothèse d'équité faible par rapport à Q : en fait, nous avons coupé les branches non équitables.

Par construction de Γ_Q et de I_{Γ_Q}, s appartient nécessairement à I_{Γ_Q}, sinon cela signifierait qu'il existe s' de Q tel que $s' \notin I_{\Gamma_Q}$ ce qui est absurde.

En effet, au cours de la construction de Γ_Q, nous n'avons considéré que les traces faiblement équitables de CIP.

Soit $P \in \mathcal{A}[CIP]$ telle que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

Alors

$\forall s \in S[CIP], P(s) \Rightarrow [\forall w \in T_s, \exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i)))]$

$\forall s \in S[CIP], P(s) \Leftrightarrow s \in \tilde{\rho}[CIP](P)$.

$\forall s \in S[CIP], (s \in \tilde{\rho}[CIP](P)) \Rightarrow [\forall w \in T_s, \exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i)))]$

$\forall s \in S[CIP], (s \in \tilde{\rho}[CIP](P)) \Rightarrow (s \in I_{\Gamma_Q})$

D'où : $\tilde{\rho}[CIP](P) \subseteq I_{\Gamma_Q}$.

④ Soit P une assertion telle que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

1) Nous raisonnons sur un élément quelconque s de $\tilde{\rho}[CIP](P)$.

On définit pour chaque état de $\tilde{\rho}[CIP](P)$ un multiensemble d'actions de CIP appelé multiensemble d'actions critiques de CIP relativement à Q noté a(s).

Pour cela, nous utilisons une induction structurelle pour construire a(s) :

Pas 1 : $\forall s \in \tilde{\rho}[CIP](Q)$, $a(s) = \emptyset$.

Pas général : $s \in \tilde{\rho}[CIP](P)$ et $s \notin \tilde{\rho}[CIP](Q)$.

$$a(s)=[\bigcup_{t[CIP](s,u')} a(u')]\cup\{a\in A[CIP] / (\underline{at}\ a)(s)\wedge\sim Q(s)\wedge$$
$$(\forall u'\in S[CIP],[(\underline{at}\ a)([s,u'[)\wedge t^*[CIP](s,u'$$
$$\sim Q([s,u'[)\wedge(\underline{jafter}\ a)(u')]$$
$$\Rightarrow$$
$$[(a(u') \subseteq_{\neq} a(s))]\wedge$$
$$(\forall w\in T_{u'},\exists w'\in T_{u'},(w\leqslant w')\wedge$$
$$(\exists i\in dom(w'),Q(w'(i))))]$$

Cette construction est basée sur Γ_Q dont la définition est telle qu'une action atomique est prête à être activée ou on est dans un état validant Q. Nous avons considéré des multiensembles car une action atomique peut être critique plusieurs fois.

Donc on considère $M_p = \{a(s)/P(s)\}$ pour 1- du b).

Nous prouvons maintenant le point 2 - b).

2) Soit s un état tel que P(s) et $Q(s) \neq \emptyset$.

Cas 1 : $a(s) = \{a\}$.

Par construction de a(s), alors (at a)(s) est vrai et $\sim Q(s)$ est vrai.

Cas 2 : On suppose que

$$\forall s'\in S[CIP][(s\neq s')\wedge(\forall w\in T_s,\exists w'\in T_s,(w\leqslant w')\wedge(\exists i\in dom(w'),Q(w'(i)))]$$
$$\Rightarrow$$
$$[(a(s')\subset a(s))\Rightarrow(\exists a\in a(s'),(\underline{at}\ a)(s'))].$$

Soit $s' \in S[CIP]$ quelconque tel que $t[CIP](s,s')$. Puisque P(s) est vraie, alors :

$$\forall w\in T_s,\exists w'\in T_s,(w\leqslant w')\wedge(\exists i\in dom(w'),Q(w'(i))).$$

Nécessairement $a(s') \subset a(s)$ et, par hypothèse, il existe une action atomique a de a(s') telle que (at a)(s') est vraie. On distingue deux cas :

Cas 2.1 :

Au cours de la transition $t[CIP](s,s')$, a n'a pas été exécutée, alors (at a)(s) est vraie et $a \in a(s)$.

Cas 2.2 :

Au cours de la transition $t[CIP](s,s')$, b a été exécutée et b est distincte de a. On distingue deux sous-cas : ou bien, b est une action critique pour s et dans ce cas

$(\underline{jafter}\ b)(s') \equiv (\underline{at}\ a)(s)$ et $b \in a(s), (\underline{at}\ b)(s)$.

ou bien, b n'est pas critique et puisque $a(s') \subset a(s)$ aucune action crit
de $a(s)$ n'a été exécutée et donc il existe c de $a(s)$ telle que (at c)

On remarquera l'utilisation des assertions jafter, qui nous paraissent indispensables pour les preuves de fatalité.

3) Cette propriété est due essentiellement à la forme de $a(s)$ et à s construction. □

Corollaire [II.6].1 :

Soit CIP un programme parallèle,
$\mathfrak{A}$[CIP] un langage d'assertions pour CIP,
$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable pour CIP,
Q une assertion de $\mathfrak{A}$[CIP].

Pour toute assertion P de $\mathfrak{A}$[CIP] telle que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, il existe un plus petit ordinal noté $O_{WF}(P,Q)$, tel que :

$\tilde{\rho}$[CIP](P) $\subset I_{\Gamma_Q}$ où Γ_Q est l'opérateur relatif à Q et

$O_{WF}(P,Q) \leqslant \|\Gamma_Q\|$.

Preuve :

Elle est immédiate, du fait de l'opérateur Γ_Q monotone et du théorème précédent :

$O_{WF}(P,Q) = \sup(\alpha(s)/s \in \tilde{\rho}[CIP](P))$. □

Définition [II.6].3 :

Soit CIP un programme parallèle,
$\mathfrak{A}$[CIP] un langage d'assertions pour CIP,
$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable pour CIP,
Q une assertion de $\mathfrak{A}$[CIP],
P une assertion de $\mathfrak{A}$[CIP] telle que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

On appelle ordinal de fatalité de P relativement à Q pour $\mathcal{M}_{WF}$ l'ordinal $O_{WF}(P,Q)$.

Nous n'avons pas voulu surcharger le théorème [II.6].1 d'hypothèses supplémentaires restrictives mais nous donnons maintenant une restriction aux ordinaux dénombrables comme pour le cas du modèle de fatalité sans hypothèse cité au chapitre I.

Théorème [II.6].2 :

Soit CIP un programme parallèle,
$\mathfrak{A}$[CIP] un langage d'assertions pour CIP,
$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable pour CIP,
Q une assertion de $\mathfrak{A}$[CIP].

Il existe un opérateur monotone Γ_Q tel que :
- $\|\Gamma_Q\| \leqslant \kappa^{\mathcal{U}[CIP]}$ où $\mathcal{U}$[CIP] est la structure abstraite de CIP.
- si t[CIP] est dénombrable, alors $\|\Gamma_Q\| < \omega_1$.

Preuve :

La preuve du premier tiret est une conséquence de la monotonie de Γ_Q et ce point a été largement utilisé au cours de la preuve du théorème [II.6].1.

Supposons que $t[CIP]$ soit dénombrable : pour tout s de $S[CIP]$, $|\{s' \in S[CIP]/t[CIP](s,s')\}| \leqslant \omega$.

Supposons que $\|\Gamma_Q\| \geqslant \omega_1$. Alors soit $s \in \Gamma_Q$.

Puisque $t[CIP]$ est dénombrable,

$|\{s' \in S[CIP]/t^*[CIP](s,s')\}| \leqslant \omega$. Soit $a \in A[CIP]$ tel que $(\underline{at}\ a)(s)$ et a vérifie :

$$[(\underline{at}\ a)(s) \wedge t[CIP](s,s') \wedge (\underline{jafter}\ a)(s')]$$

$$\left[\wedge[\forall s' \in S[CIP], (\underline{at}\ a)(s) \wedge t^*[CIP](s,s') \wedge (\underline{at}\ a)([s,s'[) \wedge (\underline{jafter}\ a)(s') \Rightarrow (s' \in \Gamma^{\alpha}) \wedge (\alpha < \omega_1)\right]$$

$$\wedge[\forall s' \in S[CIP], t[CIP](s,s') \Rightarrow [\ ]].$$

$$\forall s' \in S[CIP], (\underline{at}\ a)(s) \wedge t^*[CIP](s,s') \wedge (\underline{at}\ a)([s,s'[) \wedge (\underline{jafter}\ a)(s') \Rightarrow (s' \in \Gamma_Q^{\alpha}) \wedge (\alpha < \omega_1)$$

Pour chaque état s' de $S[CIP]$ vérifiant $(s' \in \Gamma_Q^{\alpha}) \wedge (\alpha < \omega_1)$ on note $\alpha_{s'}$, cet ordinal.

$\alpha(s) = \mathrm{Sup}(\alpha_{s'}/s'$ comme ci-dessus) et puisque $t[CIP]$ est dénombrable $\alpha(s)$ est dénombrable. Or $\alpha(s) = \omega_1$ et donc $\alpha(s) = \omega_1 < \omega_1$. Ce qui est absurde. □

Ces deux théorèmes constituent une généralisation utile et constructive pour la méthode d'OWICKI et LAMPORT. On remarquera notamment la notion de multiensemble d'actions critiques qui est en fait un ensemble muni d'un ordre bien fondé. La preuve est constructive en ce sens qu'elle donne une suite d'assertions intermédiaires utiles pour la preuve de complétude sémantique.

Afin de donner une illustration, considérons l'exemple standard :

CIP_1 ≡ Cobegin b:p:=false ∥c:while p do d:skip od coend

Précondition : p

Sous hypothèse d'équité faible, b est une action critique intuitivement et formellement. En effet si s_0 est un état d'entrée, $a(s_0) = \{b,c,d\}$ mais c et d ne sont critiques que par rapport à b.

Nous avons utilisé des multiensembles d'actions critiques car une action critique peut être critique plusieurs fois au cours du rapprochement de Q. Considérons l'exemple suivant noté CIP_1'.

```
CIP1' ≡ Cobegin
    a1 :while p do
          a2:q:=false
        od
        ∥
    b1:while q do
          b2:Skip
        od;
    b3:q:=true
        ∥
    c1:while q do
          c2:Skip
        od;
    c3:p:=false
      coend
```

Préconditions d'entrée : p∧q

a_2 est une action critique car d'elle dépend la terminaison des boucles

b_1 et c_1 ; mais si on exécute a_2, puis b_1 alors a_2 devient à nouveau critique si b_3 est exécutée.

a_2 a donc un degré critique d'ordre 2 pour certains états d'entrée.

Donc on associe à chaque assertion d'entrée une famille de multiensembl d'actions critiques qui décrivent en fait l'exécution à partir des états val dant l'assertion d'entrée.

Théorème [II.6].3 : (de décomposition)

Soit $CI\!P$ un programme parallèle,

$\mathcal{A}[CI\!P]$ un langage d'assertions pour $CI\!P$,

$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable pour $CI\!P$,

P,Q deux assertions quelconques de $\mathcal{A}[CI\!P]$.

On suppose que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

Alors il existe une suite d'assertions de $\mathcal{A}[CI\!P]$, notée $(R(\alpha))_{\alpha \leqslant \kappa^{\mathcal{U}}}$ où $\mathcal{U}[CI\!P]$ est la structure abstraite de $CI\!P$, telle que :

(1) $\mathcal{M}_{WF} \models P \rightsquigarrow \exists \alpha R(\alpha)$

(2) $\forall \alpha > 0$, $\alpha \leqslant \kappa^{\mathcal{U}[CI\!P]}$,

$\mathcal{M}_{WF} \models R(\alpha) \rightsquigarrow \exists \beta < \alpha, R(\beta)$

(3) $\mathcal{M}_{WF} \models R(0) \rightsquigarrow Q$.

Preuve :

On considère l'opérateur monotone Γ_Q qui permet de construire la suit $R(\alpha)$ de la façon suivante :

A chaque élément s de I_{Γ_Q}, on associe un ordinal noté $\alpha(s)$. On a

montré que $\tilde{\rho}[CI\!P](P) \subset I_{\Gamma_Q}$ et qu'il existe un ordinal $O_{WF}(P,Q)$, appelé ordinal de fatalité de P relativement à Q.

Cas $\alpha = 0$

$$\tilde{\rho}[CI\!P](R(0)) = \{s \in S[CI\!P]/(s \in I_{\Gamma_Q}) \wedge (\alpha(s) = 0)\}$$

$$\tilde{\rho}[CI\!P](R(0)) = \tilde{\rho}[CI\!P](Q) = \Gamma_Q^0.$$

Posons : $R(0) \equiv Q$.

Cas général 1 : $\alpha > 0$ et $\alpha \leqslant \|\Gamma_Q\|$

$$\tilde{\rho}[CI\!P](R(\alpha)) = \{s \in S[CI\!P]/(s \in I_{\Gamma_Q}) \wedge (\alpha(s) = \alpha)\}.$$

Cas général 2 : $\alpha > \|\Gamma_Q\|$ et $\alpha \leqslant \kappa^{\mathcal{U}[CI\!P]}$

On pose : $R(\alpha) = R(\|\Gamma_Q\|)$.

Par construction de Γ_Q et par le théorème [II.6].1, les assertions ci-dessus vérifient trivialement les points (1), (2), (3).

(1) Puisque $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, alors par le théorème [II.6].1,

$$\tilde{\rho}[CI\!P](P) \subset I_{\Gamma_Q} \quad \text{et} \quad I_{\Gamma_Q} = \bigcup_{\alpha \leqslant \|\Gamma_Q\|} \{s \in S[CI\!P]/\alpha = \alpha(s)\}$$

en supposant que $\alpha(s)$ est défini pour $s \in I_{\Gamma_Q}$.

On en déduit :

$$I_{\Gamma_Q} = \bigcup_{\alpha \leqslant \kappa} \mathcal{U}[CI\!P]\, \tilde{\rho}[CI\!P](R(\alpha)).$$

D'où : $\tilde{\rho}[CI\!P](P) \subset \bigcup_{\alpha \leqslant \kappa} \mathcal{U}[CI\!P]\, \tilde{\rho}[CI\!P](R(\alpha))$.

On en déduit : $\rho[C\mathbb{P}](\tilde{\rho}[C\mathbb{P}])(P) \Rightarrow \rho[C\mathbb{P}](\bigcup_{\alpha \leq \kappa} \mathcal{U}[C\mathbb{P}]^{\tilde{\rho}[C\mathbb{P}](R(\alpha))}$

D'où par définition de $\tilde{\rho}[C\mathbb{P}], \rho[C\mathbb{P}]$ et de la validité

$\forall s \in S[C\mathbb{P}], P(s) \Rightarrow (\exists \alpha R(\alpha))(s)$; on en déduit trivialement :

$\mathcal{M}_{WF} \models P \rightsquigarrow \exists \alpha R(\alpha)$.

(2) Soit $\alpha > 0$ et $\alpha \leq \kappa^{\mathcal{U}[C\mathbb{P}]}$.

Soit s tel que $R(\alpha)(s)$ soit vrai.

Alors $s \in \Gamma_Q^\alpha$; or

$\forall w \in T_s, \exists w' \in T_s, (w \leq w') \wedge (\exists i \in dom(w'), Q(w'(i)))$.

On en déduit que s conduit à un Γ_Q^β où $\beta < \alpha$ sous hypothèse d'équ faible :

$\mathcal{M}_{WF} \models R(\alpha) \rightsquigarrow \exists \beta < \alpha, R(\beta)$ car $R(0) \equiv Q$.

(3) $R(0) \equiv Q$ et donc $R(0) \Rightarrow Q$ est vrai.

D'où trivialement :

$\mathcal{M}_{WF} \models R(0) \rightsquigarrow Q$. □

Ce théorème permet de donner une suite d'assertions intermédiaires que l'on a besoin pour les preuves de telles propriétés mais nous n'avons donné qu'une décomposition sémantique et pour le cas d'une preuve il faut que le système de preuve contienne des règles et axiomes pour prouver chaque point et la réciproque pour assurer la correction mais ceci est l'objet du prochain chapitre. Mais revenons sur les résultats de ce chapitre liés à cet opérateur monotone Γ_Q ; la preuve du théorème [II.6].1 met en évidence les conditions que doit remplir un système de preuve pour qu'il soit correct et sémantiquement complet. De plus, la preuve donne une stratégie de preuve : trouver

les actions critiques. Notre étude est liée aux ordinaux et aux structures bien fondées, ce qui nous paraît normal du fait des études antérieures [LPS], [FLO], [HOA], [P], [MAP1], [MAP2], [MAP3].

Nous nous limitons aux ordinaux dénombrables (ie $< \omega_1$) c'est-à-dire, à une relation de transition dénombrable.

L'utilisation des ordinaux revient à linéariser le temps d'exécution en le représentant par une suite d'ordinaux ; en fait, on utilise une relation d'équivalence et on raisonne sur le quotient dont le temps est linéaire. L'approche complémentaire revient à considérer le temps comme arborescent ; ces deux approches sont équivalentes et EMERSON et HALPERN [EMA2] en donnent une étude intéressante.

Le raisonnement est d'autant plus simple, s'il est linéaire et la méthode d'OWICKI et LAMPORT [OWL] permet de mélanger les deux conceptions du temps, ce qui est apparu avec les multiensembles $a(s)$ et les ordinaux $\alpha(s)$. Nous devons ajouter quelques règles pour compléter le système d'OWICKI et LAMPORT [OWL].

Nous ajouterons enfin que le fait de se limiter au cas où la relation de transition est dénombrable est un choix de cadre d'étude et que nous aurions pu choisir de ne pas faire de telles hypothèses mais nous aurions dû alors considérer un langage infinitaire pouvant prendre en compte de telles cardinaux :

- pour notre cas, $\mathcal{L}_{\omega_1 \omega}$ suffit amplement.
- dans un autre cas, $\mathcal{L}_{\alpha\omega}$ où $\alpha = |\kappa^{\mathcal{U}[C\mathbb{P}]}|$ est le cardinal de $\kappa^{\mathcal{U}[C\mathbb{P}]}$ aurait dû être utilisé pour garantir l'expressivité des disjonctions de α éléments.

III - SYSTÈME AXIOMATIQUE DE PREUVE OL

III - SYSTÈME AXIOMATIQUE DE PREUVE OL

III.1.- PRESENTATION DE OL

A partir du système axiomatique proposé par OWICKI et LAMPORT [OWL], nous construisons un système axiomatique enrichi de règles d'inférence supplémentaires et en nombre minimal.

Ce système nous permettra de prouver des propriétés de fatalité sous hypothèse d'équité faible. Notre objectif dans l'enrichissement était la correction et la complétude sémantique.

OL est composé de trois parties que l'on décrit brièvement comme suit :

- une première partie formée d'axiomes permettant de spécifier l'hypothèse d'équité faible.
- une seconde partie composée de quatre règles d'inférence dont l'une d'elles autorise l'induction ordinale dans les preuves. OWICKI et LAMPORT [OWL] ne donnent aucune règle d'induction et la preuve de terminaison devient impossible ; leur système est incomplet.
- une dernière partie permettant de prouver certains prémisses de règles de la deuxième partie. En effet, nous aurons besoin de déduire des propriétés d'invariance et des implications pour établir certaines propriétés. Ainsi, nous utilisons un système de preuve d'invariance sûr et sémantiquement complet et ceci n'était pas réalisé par OWICKI et LAMPORT [OWL].

Il nous reste maintenant à énoncer le système axiomatique OL.

Soit CP un programme parallèle,

$\mathcal{A}$[CP] un langage d'assertions pour CP,

Ord, l'ensemble des ordinaux dénombrables.

On suppose que $t[C\mathbb{P}]$ est dénombrable.

P, Q, R, R(α), R(β), I, S désignent des assertions de $C\mathbb{P}$: ie

$P, Q, R, R(\alpha), R(\beta), I, S \in \mathcal{A}[C\mathbb{P}]$.

Partie 1 : Axiomatique de l'équité faible

F1 : $(\underline{at}[v:=e]) \rightsquigarrow (\underline{jafter}[v:=e])$ où $v \in \mathcal{V}[C\mathbb{P}]$, $e \in \mathcal{E}$.

F2 : $(\underline{at}[v:=?]) \rightsquigarrow (\underline{jafter}[v:=?])$ où $v \in \mathcal{V}[C\mathbb{P}]$.

F3 : $(\underline{at}[\underline{skip}]) \rightsquigarrow (\underline{jafter}[\underline{skip}])$.

F4 : $(\underline{at}\ i) \rightsquigarrow (((\underline{jat}\ i_1)\wedge B)\vee((\underline{jat}\ i_2)\wedge\sim B))$

où $i \equiv [\underline{if}\ B\ \underline{then}\ i_1\ \underline{else}\ i\ \underline{fi}]$, $B \in \mathcal{B}$, $i_1, i_2 \in \mathcal{P}$.

F5 : $(\underline{after}\ i_1) \rightsquigarrow (\underline{jafter}\ i)^{(1)}$ où i_1, i comme ci-dessus.

F6 : $(\underline{after}\ i_2) \rightsquigarrow (\underline{jafter}\ i)^{(2)}$ où i_2, i comme ci-dessus.

F7 : $(\underline{at}[\underline{while}\ B\ \underline{do}\ i'\ \underline{od}]) \rightsquigarrow ((\underline{jat}\ i')\wedge B)\vee(\underline{jafter}[\underline{while}\ B\ \underline{do}\ i'\ \underline{od}]\wedge\sim B)$

où i' est un élément de $\mathcal{P}$.

F8 : $(\underline{after}\ i') \rightsquigarrow (((\underline{jat}\ i')\wedge B)\vee(\underline{jafter}[\underline{while}\ B\ \underline{do}\ i'\ \underline{od}])\wedge\sim B))$

où i' est un élément de $\mathcal{P}$.

F9 : $((\underline{at}[\underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend}])\wedge P') \rightsquigarrow (\underline{jat}\ \mathbb{P}S_1\wedge\ldots\wedge\underline{jat}\ \mathbb{P}S_n\wedge P)$

où $P' \equiv P[\underline{at}[\underline{Cobegin}\ldots\underline{coend}]/\underline{jat}\ \mathbb{P}S_1,\ldots,\underline{jat}\ \mathbb{P}S_n], P, P' \in \mathcal{A}[C\mathbb{P}]$.

F10 : Pour tout j de $\{1,\ldots,n\}$,

$(\underline{after}\ \mathbb{P}S_1\wedge\ldots\wedge\underline{after}\ \mathbb{P}S_{j-1}\wedge\underline{jafter}\ \mathbb{P}S_j\wedge\underline{after}\ \mathbb{P}S_{j+1}\wedge\ldots\wedge\underline{after}\ \mathbb{P}S_n\wedge P'_j)$

$\rightsquigarrow((\underline{jafter}[\underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend}])\wedge P_j)$

où $P'_j, P_j \in [C\mathbb{P}], P'_j \equiv P_j[\underline{after}\ \mathbb{P}S_1,\ldots,\underline{after}\ \mathbb{P}S_n/\underline{jafter}[\underline{Cobegin}\ldots\underline{coend}]$

On note $P_j[u/u']$ la substitution dans P_j de u à u' et en fait on nous permet de parler d'assertions de variables libres de tout contrôle.

On notera, dans la suite, "$_{OL}\vdash$" pour signifier que la partie droite a été déduite à partir de OL.

On écrira, "$I \Rightarrow OI$ en $s \in S[C\mathbb{P}]$", pour exprimer l'assertion suivante :

$\forall s,s' \in S[C\mathbb{P}], I(s)\wedge t[C\mathbb{P}](s,s') \Rightarrow I(s')$.

Partie 2 : Règles d'inférence

Règle 1 : Si $_{OL}\vdash P \Rightarrow Q$, alors $_{OL}\vdash P \rightsquigarrow Q$.

Règle 2 : Si $_{OL}\vdash P \rightsquigarrow Q$,

$$_{OL}\vdash \boxed{\begin{array}{l}\exists I \in \mathcal{A}[C\mathbb{P}], P\wedge R \Rightarrow I,\\ \qquad\qquad I \Rightarrow OI,\\ \qquad\qquad I\wedge Q \Rightarrow S\end{array}},$$

alors $_{OL}\vdash (P\wedge R) \rightsquigarrow (Q\wedge S)$.

Règle 3 : Si $_{OL}\vdash P \rightsquigarrow \exists\alpha R(\alpha)$,

$\forall\alpha \in Ord, \alpha > 0, {}_{OL}\vdash R(\alpha) \rightsquigarrow \exists\beta < \alpha, R(\alpha)$,

$_{OL}\vdash R(0) \rightsquigarrow Q$,

alors $_{OL}\vdash P \rightsquigarrow Q$

Règle 4 : Si $_{OL}\vdash P \rightsquigarrow Q$, $_{OL}\vdash Q \Rightarrow R$,

alors $_{OL}\vdash P \rightsquigarrow R$

Partie 3 : Règles auxiliaires

Règle 1A : Si $P \Rightarrow Q$ est vraie pour CIP, c'est-à-dire,

$\forall s \in S[CIP], P(s) \Rightarrow Q(s)$,

alors $_{OL}\vdash P \Rightarrow Q$.

Règle 2A : Si S est invariant par rapport à $P \wedge R$ et Q, c'est-à-dire

$\forall s,s' \in S[CIP], (P \wedge R)(s) \wedge t^*[CIP](s,s') \wedge Q(s') \Rightarrow S(s')$, alors

$$_{OL}\vdash \left[\begin{array}{l} \exists I \in \mathfrak{A}[CIP], P \wedge R \Rightarrow I \\ \qquad\qquad\qquad\quad I \Rightarrow OI \\ \qquad\qquad\qquad\quad I \wedge Q \Rightarrow S \end{array}\right]$$

Règle 3A : Si $P \rightsquigarrow Q$ est une hypothèse de travail, alors $_{OL}\vdash P \rightsquigarrow Q$.

Nous allons maintenant donner quelques explications au sujet des différentes règles et axiomes donnés ci-dessus. Les axiomes formant la première partie spécifient l'hypothèse d'équité faible faite au niveau de la structure sémantique de fatalité. Les axiomes d'équité faible au début et à la fin du Cobegin sont corrects indépendamment de l'hypothèse d'équité faible et nous avons, en fait, distingué le début du programme parallèle et le début de chaque programme séquentiel ainsi que la fin des programmes séquentiels et la fin du programme parallèle ; pour cette raison, nous avons dû ajouter ces deux axiomes contrairement à OWICKI et LAMPORT [OWL] qui ne distinguaient pas ces deux points de contrôle. Cette remarque vaut aussi pour la fin d'une instruction conditionnelle : en effet, nous distinguons 3 points : la fin de l'instruction correspondant à la condition vraie, la fin de l'instruction correspondant à la condition fausse et la fin de l'instruction conditionnelle. Ce choix est dû au choix de la sémantique opérationnelle, qui distingue ces trois points de contrôle. On voit ici que la notion d'action atomique joue un rôle très conséquent pour la notion de fatalité.

La deuxième partie permet de définir, dans un cadre général, la relation $\rightsquigarrow$ sur les formules ou assertions d'un langage $\mathfrak{A}[CIP]$ pour CIP. La règle 1 établit que l'implication est incluse dans la relation de fatalité et on notera que cela est vrai quelle que soit l'hypothèse faite sur la structure sémantique. La règle 2 permet d'enrichir les assertions de fatalité $P \rightsquigarrow Q$, en utilisant une assertion invariante pour CIP et cette assertion lie de façon explicite les deux notions de fatalité et d'invariance. De même ici encore, l'hypothèse d'équité faible n'est pas nécessaire pour la correction de cette règle. La règle 3 est une règle d'induction sur les ordinaux et on pourrait la remplacer par une règle sur une structure bien ordonnée, voire même bien fondée. Les ordinaux nous sont apparus nécessaires et ceci conduit à l'utilisation de variables auxiliaires dans les preuves de fatalité. De même, pour cette règle, aucune hypothèse n'est nécessaire sur la structure sémantique de fatalité. Enfin, la règle 4 fut ajoutée car nous avions besoin de prouver la transitivité de $\rightsquigarrow$ et ceci n'était pas possible sans cette règle.

Les règles de la troisième partie nous sont très utiles pour déduire les prémisses des règles 1, 2, 3, 4 de la partie 2. En effet, nul n'ignore que toute extension axiomatisée de l'arithmétique est incomplète (voir, par exemple, SHOENFIELD [SHO]) et ce résultat est connu sous le nom de "théorème d'incomplétude de Gödel" ; devant cet obstacle, nous avons préféré utiliser une sorte d'oracle pour y remédier. Nous supposons donc pouvoir dire si une

implication est vraie ou fausse et, dans ce cas, on déduit cette assertion dans OL.

Le deuxième obstacle, que nous pouvions rencontrer, concernait le langage d'assertions, qui devait être suffisamment expressif au sens de COOK [COO], afin d'assurer la complétude sémantique de la méthode de preuve d'invariance. Notre règle est correcte modulo le choix d'une méthode de preuve d'invariance correcte et sémantiquement complète et un choix raisonnable est la méthode axiomatique d'OWICKI et GRIES [OWG], dont le seul grief, que l'on puisse lui porter, est de ne pas axiomatiser la notion d'interférence. Enfin, une bien curieuse règle semble être la règle 3 ; celle-ci permet d'ajouter des hypothèses supplémentaires pour faire la preuve. En fait, elle nous est apparue comme nécessaire dans les preuves pratiques que nous avons réalisées, en particulier pour des programmes incomplètement spécifiés. Dans un algorithme destiné à réaliser l'exclusion mutuelle, si on veut prouver que toute partie du système désirant entrer en section critique, y entrera fatalement un jour, il faut supposer que toute partie en section critique en sort un jour ; cette hypothèse est à considérer à l'aide de la règle 3.

Après cette brève justification du système OL, nous voulons expliciter les raisons qui nous poussent à utiliser OL pour les preuves de fatalité sous hypothèse d'équité faible. Ce système nous permet de développer formellement des preuves de fatalité de programmes parallèles sous hypothèse d'équité faible. APT et OLDEROG [APO] traitent le cas de programmes DO sous hypothèse d'équité faible mais leur méthode demande une transformation du programme considéré. De même, APT, PNUELI et STAVI [APS] donnent une méthode qui utilise explicitement des variables dites de délai pour prendre en compte cette hypothèse d'équité. Cette approche nous paraît intéressante formellement mais est difficile à appliquer en pratique. La méthode utilisant OL permet implicitement d'implanter des variables de délai, sans s'y référer. Enfin, toutes les propriétés de fatalité sont ainsi traitées dans ce cadre.

Avant d'examiner les questions de correction et de complétude sémantique de OL, nous donnons des exemples de preuve avec OL de propriétés de fatalité sous hypothèse d'équité faible.

Exemple 1

CP_1 ≡ Cobegin[p:=false] ‖while p do[Skip]od coend

Propriété : $_{OL}\vdash ((\underline{at}\ CP_1)\wedge p) \rightsquigarrow ((\underline{after}\ CP_1)\wedge \sim p)$

Preuve :

$\underline{at}\ CP_1 \wedge p \equiv \underline{at}(\underline{Cobegin}...\underline{coend})\wedge p$ (par définition de cette assertion de contrôle)

(1) $_{OL}\vdash (\underline{at}(\underline{Cobegin}...\underline{coend})\wedge p) \rightsquigarrow (\underline{jat}[p:=false]\wedge \underline{jat}[\underline{while}\ p\ \underline{do}[\underline{Skip}]od]\wedge p)$
(par l'axiome F9, de la partie 1 de OL)

(2) $(\underline{jat}[p:=false]\wedge \underline{jat}[\underline{while}\ p\ \underline{do}[\underline{Skip}]\underline{od}]\wedge p)\Rightarrow(\underline{jat}[p:=false]\wedge p)$
(par propriété de ⇒ et ∧)

(3) $_{OL}\vdash (\underline{jat}[p:=false]\wedge \underline{jat}[\underline{while}\ p\ \underline{do}[\underline{Skip}]od]\wedge p)\Rightarrow(\underline{jat}[p:=false]\wedge p)$
(par la règle 1A avec (2))

(4) $_{OL}\vdash (\underline{jat}[p:=false]\wedge \underline{jat}[\underline{while}\ p\ \underline{do}[\underline{Skip}]\underline{od}]\wedge p) \rightsquigarrow (\underline{jat}[p:=false]\wedge p)$
(par la règle 1 et avec (3)

(5) $(\underline{jat}[p:=false]\wedge p)\Rightarrow(\underline{at}[p:=false]\wedge p)$
(par définition de jat et at)

(6) $_{OL}\vdash$ (jat[p:=false]$\wedge$p)$\Rightarrow$(at[p:=false]$\wedge$p)

(par la règle 1A avec (5))

(7) $_{OL}\vdash$ (jat[p:=false]$\wedge$jat[while p do[Skip]od]$\wedge$p) $\rightsquigarrow$ (at[p:=false]$\wedge$p)

(par la règle 4 avec (4) et (6))

Nous démontrerons ultérieurement la règle de transitivité de $\rightsquigarrow$ déd
à partir de OL :

Règle 5 : Si $_{OL}\vdash P \rightsquigarrow Q$ et $_{OL}\vdash Q \rightsquigarrow R$, alors $_{OL}\vdash P \rightsquigarrow R$.

(8) $_{OL}\vdash$ ((at CIP_1)$\wedge$p) $\rightsquigarrow$ (at[p:=false]$\wedge$p)

(par la règle 5 avec (1) et (7))

(9) (at[p:=false]$\wedge$p)$\Rightarrow$at[p:=false]

(par propriété de $\Rightarrow$ et $\wedge$)

(10) $_{OL}\vdash$ (at[p:=false]$\wedge$p)$\Rightarrow$at[p:=false]

(par la règle 1A avec (9))

(11) $_{OL}\vdash$ (at CIP_1 $\wedge$p) $\rightsquigarrow$ at[p:=false]

(par la règle 4 avec (8) et (10))

(12) $_{OL}\vdash$ (at[p:=false]) $\rightsquigarrow$ (jafter[p:=false])

(axiome F1)

(13) (jafter[p:=false]) $\Rightarrow$ (after[p:=false])

(par définition de jafter et after)

(14) $_{OL}\vdash$ (jafter[p:=false]) $\Rightarrow$ (after[p:=false])

(par la règle 1A avec (13))

(15) $_{OL}\vdash$ (at[p:=false]) $\rightsquigarrow$ (after[p:=false])

(par la règle 4 avec (12) et (14))

L'assertion suivante : [after[p:=false]$\Rightarrow$ $\sim$ p] est invariante pour CIP_1 ; on utilise la règle 2A et la règle 2 pour prouver

(16) $_{OL}\vdash$ at[p:=false] $\rightsquigarrow$(after(p:=false)$\wedge\sim$p)

en effet,
- (at[p:=false]) $\Rightarrow$[after[p:=false]$\Rightarrow\sim$p],
- [after[p:=false]$\Rightarrow\sim$p] $\Rightarrow$ O[after[p:=false]$\Rightarrow\sim$p],
- [after[p:=false]$\Rightarrow\sim$p]$\wedge$after[p:=false]$\Rightarrow\sim$p
- (15)

(17) $_{OL}\vdash$ (at CIP_1 $\wedge$p) $\rightsquigarrow$ [after[p:=false]$\wedge\sim$p]

(par la règle 5 avec (11) et (16))

L'assertion suivante : [(at[while p do skip]$\Rightarrow$[after[p:=false]$\Rightarrow\sim$p])
$\wedge$(at skip $\Rightarrow$ [after[p:=false]$\wedge\sim$p])
$\wedge$(after skip $\Rightarrow$ [after[p:=false]$\wedge\sim$p])
$\wedge$(after[while p do skip od]$\Rightarrow$[after[p:=false]$\wedge\sim$p])]
est invariante pour CIP_1 ; on la note P.

- at $CIP_1\wedge$p $\Rightarrow$ P,
- P $\Rightarrow$ OP

(18) $((\underline{\text{after}}[p:=\text{false}])\wedge\sim p)\Rightarrow[(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{at}}\ \underline{\text{while}})$
$\vee(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{at}}\ \text{skip})$
$\vee(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{after}}\ \text{skip})$
$\vee(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{after}}\ \text{while})]$

(par analyse par cas)

(19) $_{OL}\vdash((\underline{\text{after}}[p:=\text{false}])\wedge\sim p)\Rightarrow[(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{at}}\ \underline{\text{while}})$
$\vee(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{at}}\ \underline{\text{skip}})$
$\vee(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{after}}\ \underline{\text{skip}})$
$\vee(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{after}}\ \text{while})]$

$(\underline{\text{after}}[p:=\text{false}]\wedge\sim p\wedge\underline{\text{at}}\ \underline{\text{while}})\Rightarrow P$

$P\Rightarrow OP$

$[(\underline{\text{jat}}\ \text{Skip}\wedge\underline{\text{after}}[p:=\text{false}]\wedge p)\Rightarrow\text{false}]$

$(\underline{\text{jafter}}\ \text{while}\wedge\underline{\text{after}}[p:=\text{false}]\wedge\sim p)\Rightarrow\sim p$

(20) $_{OL}\vdash([\underline{\text{after}}[p:=\text{false}]]\wedge[\underline{\text{at}}\ \text{while}]\wedge\sim p)\rightsquigarrow[\underline{\text{after}}[p:=\text{false}]\wedge$
$\underline{\text{after}}[\text{while}]\wedge\sim p]$

(par la règle 2 avec les axiomes F7

$_{OL}\vdash\text{false}\rightsquigarrow R$ où R est quelcon

(21) $_{OL}\vdash(\underline{\text{at}}\ \text{skip})\rightsquigarrow(\underline{\text{after}}\ \text{skip})$

(par la règle 4 et par définition d
after et jafter)

(22) $_{OL}\vdash(\underline{\text{at}}\wedge\text{skip}\wedge\underline{\text{after}}[p:=\text{false}]\wedge\sim p)\rightsquigarrow(\underline{\text{after}}\ \text{skip}\wedge\sim p\wedge\underline{\text{after}}[p:=\text{false}]$

(par la règle 2 et en utilisant l'in
variant P)

(23) $_{OL}\vdash((\underline{\text{after}}\ \text{skip})\wedge\underline{\text{after}}[p:=\text{false}]\wedge p)\rightsquigarrow(\underline{\text{after}}\ \text{while})\wedge$
$(\underline{\text{after}}[p:=\text{false}]\wedge\sim p)$

(par la règle 2 et l'axiome F8
comme pour (20))

Nous démontrons la règle suivante :

Règle 6 : Si $_{OL}\vdash P\rightsquigarrow Q_1\vee\ldots\vee Q_i$ et $\forall j\in\{1,\ldots,1\}$,
$_{OL}\vdash Q_j\rightsquigarrow R$,

alors $_{OL}\vdash P\rightsquigarrow R$.

On la déduit à l'aide de la règle 3..

(24) $_{OL}\vdash(\underline{\text{at}}\ \text{skip}\wedge\underline{\text{after}}[p:=\text{false}]\wedge\sim p)\rightsquigarrow(\underline{\text{after}}\ \text{while}\wedge\underline{\text{after}}[p:=\text{false}]\wedge\sim p)$

(par la règle 5 avec (22) et (23))

(25) $_{OL}\vdash(\underline{\text{after}}[p:=\text{false}]\wedge\sim p)\rightsquigarrow(\underline{\text{after}}[p:=\text{false}]\wedge\underline{\text{after}}\ \text{while}\wedge\sim p)$

(par la règle 6 avec (20), (24), (23))

(26) $_{OL}\vdash(\underline{\text{at}}\ CP_1\wedge\sim p)\rightsquigarrow(\underline{\text{after}}[p:=\text{false}]\wedge\underline{\text{after}}\ \text{while}\wedge\sim p)$

(par la règle 5 avec (25) et (17))

(27) $_{OL}\vdash(\underline{\text{after}}[p:=\text{false}]\wedge\underline{\text{after}}\ \text{while}\wedge\sim p)\rightsquigarrow(\underline{\text{after}}\ CP_1\wedge\sim p)$

(par l'axiome F10)

(28) $_{OL}\vdash(\underline{\text{at}}\ CP_1\wedge p)\rightsquigarrow(\underline{\text{after}}\ CP_1\wedge\sim p)$

(par la règle 5 et (27) et (26))

CQFD.

Cet exemple illustre le fait que l'hypothèse d'équité faible est contenue dans l'axiomatique de OL. Nous n'avons fait que reprendre l'exemple d'OWICKI et LAMPORT [OWL] en en donnant une preuve détaillée : CIP_1 termine sous hypothèse d'équité faible.

Exemple 2

CIP_2 ≡ Cobegin[p:=false] [] [x:=7] [] while p do[x:=x+1]od coend

Propriété : $_{OL} \vdash (at\ CIP_2 \wedge p) \rightsquigarrow (x=7)$

La propriété de fatalité relative à CIP_2 exprime que la variable x contient la valeur 7 au cours de l'exécution de CIP_2. Le système proposé par OWICKI et LAMPORT [OWL] utilise des assertions de contrôle qui portent sur les points de contrôle physiques du programme et ces assertions sont utilisées pour montrer que l'affectation [x:=7] est exécutée mais on ne peut prouver que la valeur de x est 7 à ce moment car le fait d'être après [x:=7] ne précise pas si on est juste après ou si d'autres actions ont été exécutées, alors que le contrôle est juste après [x:=7]. Cet aspect du système d'OWICKI et LAMPORT [OWL] est à l'origine de l'incomplétude de leur système ; il nous a fallu introduire deux autres types d'assertions de contrôle portant sur l'exécution du programme et spécifiant qu'une action désignée vient d'être exécutée.

L'assertion jafter[x:=7] signifie que, dans chaque état où elle est vraie, x vaut 7 ; l'assertion after[x:=7] signifie que le contrôle est après [x:=7] mais ne donne aucun renseignement sur la valeur de x car, si p est vrai, on a pu exécuter l'affectation [x:=x+1] et changer la valeur de x.

La preuve de la propriété est déduite à partir des règles et axiomes de notre système OL.

(1) $_{OL} \vdash (at\ CIP_2 \wedge p) \rightsquigarrow (jat[p:=false] \wedge jat[x:=7] \wedge jat[while...od] \wedge p)$

(par l'axiome F9)

(2) $(jat[p:=false] \wedge jat[x:=7] \wedge jat[while...od] \wedge p) \Rightarrow jat[x:=7]$

(par propriété de l'implication et la conjonction)

(3) $jat[x:=7] \Rightarrow at[x:=7]$

(par définition de jat)

(4) $_{OL} \vdash (at\ CIP_2 \wedge p) \rightsquigarrow at[x:=7]$

(par application de la règle 1A à (2) et (3), puis de la règle 4 avec (1))

L'assertion suivante [jafter[x:=7] ⇒ (x=7)] est invariante pour CIP_2 et on l'utilise dans la règle 2 pour prouver :

(5) $_{OL} \vdash at[x:=7] \rightsquigarrow jafter[x:=7] \wedge (x=7)$

(par la règle 2 et axiome F1)

(6) $_{OL} \vdash at\ CIP_2 \wedge p \rightsquigarrow jafter[x:=7] \wedge (x=7)$

(par la règle 5)

(7) $_{OL} \vdash jafter[x:=7] \wedge (x=7) \Rightarrow (x=7)$

(par les propriétés de l'implication et la règle 1A)

(8) $_{OL} \vdash (\underline{at}\ CIP_2 \wedge p) \rightsquigarrow (x=7)$

(par la règle 4 avec (6) et (8)

Ceci termine la preuve de la propriété et nous noterons que le fait que x vaut 7 juste après l'exécution de l'affectation est crucial pour la "bonne marche" de la preuve. Rappelons aussi que le système d'OWICKI et LAMPORT [OWL] ne repose pas sur une méthode de preuve d'invariance sure et sémantiquement complète.

Les assertions de contrôle jat et jafter sont à utiliser dans les assertions intermédiaires de la méthode d'OWICKI et GRIES [OWG]. Nous poursuivons notre étude par la correction et la complétude sémantique de OL.

III.2.- CORRECTION DU SYSTEME AXIOMATIQUE OL

Théorème [III.2].1 :

Soit CIP un programme parallèle,

$\mathcal{A}[CIP]$ un langage d'assertions pour CIP,

$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable,

P,Q deux assertions de $\mathcal{A}[CIP]$ quelconques.

Si $_{OL} \vdash P \rightsquigarrow Q$, alors $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

La preuve de ce théorème occupera la suite de ce paragraphe et se décomposera comme suit :

- notion de validité.
- validation des axiomes pour le modèle $\mathcal{M}_{WF}$.
- correction des règles d'inférence pour $\mathcal{M}_{WF}$.

- justification des règles auxiliaires ajoutées au système.

III.2.1.- Validité des assertions

Cette partie permettra de préciser certaines notions de validité utilisées par la suite et permettra de considérer de façon unifiée une interprétation des formules d'états ou des formules de traces.

Soit CIP un programme parallèle quelconque (ie : CIP $\in \mathcal{C}$),

$\mathcal{A}[CIP]$ un langage d'assertions pour CIP,

$\mathcal{M}$ un modèle de fatalité pour CIP.

Etant donnée une assertion quelconque P de $\mathcal{A}[CIP]$.

On suppose qu'il existe un couple de fonctions

$(\rho[CIP], \tilde{\rho}[CIP])$ telles que $\tilde{\mathcal{M}} = (S[CIP], t[CIP], T)$.

$\mathcal{M} \models P$ ssi $\tilde{\rho}[CIP](\mathcal{M}) \models \tilde{\rho}[CIP](P)$
ie $\forall s \in S[CIP], \tilde{\rho}[CIP](P)(s)$

On a ainsi étendu $\mathcal{M}$ en une interprétation des assertions de $\mathcal{A}[CIP]$ et de fatalité de CIP. On remarquera que le sous-ensemble de S[CIP] formant la troisième composante de $\tilde{\rho}[CIP](\mathcal{M})$ n'a aucun effet sur la validité de P, assertions de CIP. Ces précisions sont nécessaires pour la suite car nous allons considérer l'implication et son interprétation sur $\mathcal{M}$.

Nous poursuivons notre preuve en trois parties et nous procédons comme d'habitude en prouvant la validité des axiomes et la correction des règles d'inférence : ie à partir d'une assertion vraie, on déduit une assertion vraie. On parle d'assertions valides pour $\mathcal{M}$ ou vraies pour $\mathcal{M}$.

III.2.2.- Validité des axiomes

Soit $\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable pour CIP.

Cas des axiomes F1, F2, F3

Les parties entre crochets sont dans chacun des cas une action atomique pour CIP ; puisque notre modèle de fatalité est $\mathcal{M}_{WF}$, il valide, par définition, les axiomes F1, F2, F3. On en déduit les expressions :

$\mathcal{M}_{WF} \models$ at[v:=e] $\rightsquigarrow$ jafter[v:=e] où $v \in \mathcal{V}$[CIP], $e \in \mathcal{E}$.

$\mathcal{M}_{WF} \models$ at[v:=?] $\rightsquigarrow$ jafter[v:=?] où $v \in \mathcal{V}$[CIP].

$\mathcal{M}_{WF} \models$ at[skip] $\rightsquigarrow$ jafter[skip].

Cas des axiomes F4, F5, F6

Il s'agit ici des axiomes relatifs à l'instruction conditionnelle ; on distingue trois actions atomiques :

$a_1 \equiv$ <B>, $a_2 \equiv$ Out-if1, $a_3 \equiv$ Out-if2 et

$i \equiv$ [if B then P_1 else P_2 fi].

Dans le modèle $\mathcal{M}_{WF}$, $\mathcal{M}_{WF} \models$ at $a_i \rightsquigarrow$ jafter a_i, $i \in \{1,2,3\}$.

D'après la définition II.5 .4, on déduit les équivalences suivantes :

at $i \equiv$ at<B>;jafter<B>$\equiv$(jat $P_1 \wedge B) \vee$(jat $P_2 \wedge \sim B$) ;

jafter $i^{(1)} \equiv$ (jafter Out-if1)$\wedge$after $i \equiv$ jafter Out-if1,

jafter $i^{(2)} \equiv$ (jafter Out-if2)$\wedge$after $i \equiv$ jafter Out-if2,

jafter Out-if1$\vee$jafter Out-if2 $\equiv$ jafter i

at Out-if1 $\equiv$ after P_1 ;

at Out-if2 $\equiv$ after P_2 ;

① at<B> $\rightsquigarrow$ jafter<B> est valide dans $\mathcal{M}_{WF}$:

On en déduit que $\mathcal{M}_{WF} \models$ at $i \rightsquigarrow$ (jat $P_1 \wedge B) \vee$(jat $P_2 \wedge \sim B$).

② at Out-if1 $\rightsquigarrow$ jafter Out-if1 est valide dans $\mathcal{M}_{WF}$:

$\mathcal{M}_{WF} \models$ after $P_1 \rightsquigarrow$ jafter $i^{(1)}$

③ at Out-if2 $\rightsquigarrow$ jafter Out-if2 est valide dans $\mathcal{M}_{WF}$:

$\mathcal{M}_{WF} \models$ after $P_2 \rightsquigarrow$ jafter $i^{(2)}$.

Cas des axiomes F7, F8

On s'attache au cas où l'instruction est un while : w:while B do P od ; ie l'action atomique concernée est le test <B>. On déduit de la définition [II.5].2 les équivalences suivantes :

at<B> $\equiv$ at w$\vee$after P

jafter<B> $\equiv$ (jat P$\wedge$B)$\vee$(jafter w$\wedge\sim$B).

De plus, on établit aisèment les implications suivantes :

$\mathcal{M}_{WF} \models$ at w $\Rightarrow$ at<B>, $\mathcal{M}_{WF} \models$ after P $\Rightarrow$ at<B>.

Puisque $\mathcal{M}_{WF} \models$ at<B> $\rightsquigarrow$ jafter<B>, on en déduit :

$\forall s \in S$[CIP], (at<B>)(s) $\Rightarrow$ As(s) où As est la partie droite dans la définition de la fatalité. Pour chaque s de S[CIP] on établit aisèment l'implication :

(at w)(s) $\Rightarrow$ As(s) et (after P)(s) $\Rightarrow$ As(s), par transitivité de l'implication. Ceci est fait pour chaque s :

$\forall s \in S$[CIP], (at w)(s) $\Rightarrow$ As(s)

$\forall s \in S$[CIP], (after P)(s) $\Rightarrow$ As(s).

$\mathcal{M}_{WF} \models$ at w $\rightsquigarrow$ (jat P∧B)∨(jafter w∧∼B)

$\mathcal{M}_{WF} \models$ after P $\rightsquigarrow$ (jat P∧B)∨(jafter w∧∼B).

Cas des axiomes F9, F10

Il s'agit ici de donner le comportement des actions atomiques Start et Finish, qui commence, resp. termine, le Cobegin. Au regard de la sémantique opérationnelle, à la suite de l'exécution de Start, les compteurs de program se positionnent simultanément au début de chaque processus composant le Cobegin et de même pour Finish, qui termine simultanément chaque processus. A priori, il n'y a aucune raison de blocage à ces deux actions atomiques. On déduit de la définition [II.5].2 que :

at Start ≡ at[Cobegin];

jafter Start ≡ jat $\mathbb{P}S_1 \wedge \ldots \wedge$ jat $\mathbb{P}S_n$;

où [Cobegin] ≡ [Cobegin $\mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n$ coend].

$\forall s \in S[C\mathbb{P}], ((\text{at Start}) \wedge P')(s) \leftrightarrow ((\text{at Start}) \wedge P\{\text{at Start}/\text{jat } \mathbb{P}S_1, \ldots, \text{jat } \mathbb{P}$

Or $\mathcal{M}_{WF} \models$ at Start $\rightsquigarrow$ jafter Start :

$\forall s \in S[C\mathbb{P}], ((\text{at Start}) \wedge P')(s) \Rightarrow (\text{at Start})(s)$

$(\text{at Start})(s) \Rightarrow \forall w \in T_s, \exists w' \in T_s, (w \leqslant w') \wedge (\exists i \in \text{dom}(w'), (\text{jafter Start})(w'(i$

si (at Start)(s) est vrai, alors $s = (\ell_1, m)$ or d'après ci-dessus il existe i tel que $w \leqslant w' \wedge$ (jafter Start)(w'(i)).

Nécessairement, i = 2 d'après la sémantique opérationnelle et $t[C\mathbb{P}](s$ et $s' = (\ell_1, \ldots, \ell_n, m)$ où (jat $\mathbb{P}S_1$)(s').

D'où, seul le contrôle change :

$\forall s \in S[C\mathbb{P}], (\text{at Start} \wedge P')(s) \Rightarrow [\forall w \in T_s, \exists w' \in T_s, (w \leqslant w')$

$\wedge (\exists i \in \text{dom}(w'), (\text{jafter Start} \wedge P)(w'(i)))]$.

On déduit : $\mathcal{M}_{WF} \models (\text{at Cobegin} \wedge P') \rightsquigarrow (\text{jat } \mathbb{P}S_1 \wedge \ldots \wedge \text{jat } \mathbb{P}S_n \wedge P)$.

Un raisonnement semblable conduit à la validité de F10.

III.2.3.- Correction des règles d'inférence

Règle 1 :

On suppose que $\mathcal{M}_{WF} \models P \Rightarrow Q$, où P, Q sont deux assertions de $\mathcal{A}[C\mathbb{P}]$. D'après notre préliminaire III.2.1, $\mathcal{M}_{WF}$ permet de valider les assertions de $\mathcal{A}[C\mathbb{P}]$ et dans le cas présent $P \Rightarrow Q$ est une assertion de $\mathcal{A}[C\mathbb{P}]$. Par définition de la notion de validité pour $\mathcal{M}_{WF}$:

$\{\mathcal{M}_{WF} \models P \Rightarrow Q \Leftrightarrow \forall s \in S[C\mathbb{P}], P(s) \Rightarrow Q(s)\}$

De plus, on note $\tilde{\rho}[C\mathbb{P}](\mathcal{M}_{WF}) = (S[C\mathbb{P}], t[C\mathbb{P}], T)$ où $T \subset T[C\mathbb{P}]$.

$\{\forall s \in S[C\mathbb{P}], [P(s) \Rightarrow Q(s)\} \leftrightarrow \{\forall s \in S[C\mathbb{P}], P(s) \Rightarrow (\forall w \in T_s, Q(w(i)))\}$

$\Leftrightarrow \{\forall s \in S[C\mathbb{P}], P(s) \Rightarrow (\forall w \in T_s, (w \leqslant w) \wedge Q(w(i)))\}$

$\Rightarrow \{\forall s \in S[C\mathbb{P}], P(s) \Rightarrow (\forall w \in T_s, \exists w' \in T_s, (w \leqslant w') \wedge (\exists i \in \text{dom}(w'), Q(w'(i))))\}$

$\Leftrightarrow \{\mathcal{M}_{WF} \models P \rightsquigarrow Q\}$.

On en déduit :

si $\mathcal{M}_{WF} \models P \Rightarrow Q$, alors $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

Règle 2 :

On suppose que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, P, Q deux assertions de $\mathcal{A}[C\mathbb{P}]$ et

$\mathcal{M}_{WF} \models \{\exists I \in \mathcal{A}[C\mathbb{P}], P \wedge R \Rightarrow I, I \Rightarrow QI, I \wedge Q \Rightarrow S\}$.

où R,S sont des assertions de $\mathcal{A}[C\mathbb{P}]$.

L'expression entre accolades signifie que chaque implication est valide pour $\mathcal{M}_{WF}$.

$\exists I \in \mathfrak{A}[CIP]$, $\mathcal{M}_{WF} \models P \wedge R \Rightarrow I$, (1)

$\mathcal{M}_{WF} \models I \Rightarrow OI$, (2)

$\mathcal{M}_{WF} \models I \wedge Q \Rightarrow S$ (3)

$\mathcal{M}_{WF} \models P \rightsquigarrow Q$ (4)

On traduit les points (1), (2), (3), (4) par :

$\forall(\underline{s},s,s',\bar{s}) \in S[CIP]^4$, $(P \wedge R)(\underline{s}) \Rightarrow I(\underline{s})$ (1')

$I(s) \wedge t[CIP](s,s') \Rightarrow I(s')$ (2')

$(I \wedge Q)(\bar{s}) \Rightarrow S(\bar{s})$ (3')

(4') $\forall s \in S[CIP], P(s) \Rightarrow (\forall w \in T_s, \exists w' \in T_s, (w \leqslant w') \wedge (\exists j \in dom(w'), Q(w'(j))))$.

Soit s un élément quelconque de S[CIP] tel que $(P \wedge R)(s)$ est vrai. D'après (1'), I(s) est vrai ; d'après (2'), I(s') est vraie pour chaque s' de S[CIP] tel que $t^*[CIP](s,s')$.

Soit w un élément quelconque de T_s.

Puisque $(P \wedge R)(s)$ est vrai, P(s) l'est aussi ; il existe w' de T_s tel que $w \leqslant w'$ et $\exists j \in dom(w')$, $Q(w'(j))$.

Or $t^*[CIP](s,w'(j))$ par définition des mots, d'où $I(w'(j))$ est vrai. On utilise alors (3') pour déduire que $S(w'(j))$ est vraie.

On a prouvé :

$\mathcal{M}_{WF} \models P \wedge R \rightsquigarrow Q \wedge S$ à partir des hypothèses.

Règle 3

On suppose que :

(1) $\mathcal{M}_{WF} \models P \rightsquigarrow \exists \alpha R(\alpha)$

(2) $\mathcal{M}_{WF} \models R(\alpha) \rightsquigarrow \exists \beta < \alpha, R(\beta)$, $\forall \alpha \in Ord$, $\alpha > 0$.

(3) $\mathcal{M}_{WF} \models R(0) \rightsquigarrow Q$ où P,Q,R(α) sont des assertions de $\mathfrak{A}[CIP]$ et $\alpha \in Ord$.

Soit s un élément quelconque de S[CIP] tel que P(s) est vraie.

Soit w un mot, commençant par s, quelconque.

A partir de (1) et puisque P(s) est vraie, on en déduit la propriété suivante :

Il existe une trace w' de T_s telle que :

$(w \leqslant w') \wedge (\exists j_1 \in dom(w'), (\exists \alpha R(\alpha))(w'(j_1)))$.

Soit $\alpha_1 \in Ord$ tel que $R(\alpha_1)(w'(j_1))$.

(1') $\exists w' \in T_s, (w \leqslant w') \wedge (\exists j_1 \in dom(w'), R(\alpha_1)(w'(j_1)))$
$\exists \alpha_1 \in Ord$.

Supposons que α_1 n'est pas nul. Puisque (2) est vérifiée, on en déduit, pour toute trace commençant par $w'(j_1)$, notée w_1.

Il existe une trace w_1' de $T_{w'(j_1)}$ telle que :

$(w_1 \leqslant w_1') \wedge (\exists j_2 \in dom(w_1'), (\exists \beta < \alpha_1, R(\alpha))(w_1'(j_2)))$.

On en déduit l'encadré suivant :

(2') $\exists w_1' \in T_{w'(j_1)}, \exists \alpha_2 \in Ord, (\alpha_1 > \alpha_2) \wedge (w_1 \leqslant w_1')$
$\wedge (\exists j_2 \in dom(w_1'), R(\alpha_2)(w_1'(j_2))$

On notera : $v_1 = w_1'[j_1], v_2 = v_1.w_1'[j_2], \ldots$ et $\alpha_1 > \alpha_2 > \ldots >$

On construit ainsi une suite en supposant que α_j est non-nul chaque fois. Puisque (Ord,<) est un ensemble muni d'une relation bien-fondée, la suite

(α_j) ne peut être décroissante infinie ; il existe k de $\mathbb{N}$ tel que $\alpha_k = 0$.

On en déduit par construction de (v_j) que :

$v_k(j_k)$ valide R(0) : on utilise alors (3) et on en déduit que, pour toute trace commençant par $v_k(j_k)$, v , il existe v' telle que

$(v \leqslant v') \wedge (\exists \ell \in dom(v'), Q(v'(\ell)))$.

Soit $w' = v_k[j_k].v'[\ell]$. w' est une trace commençante par s et $w \leqslant$

Posons $j = j_k + \ell$, alors : Q(w'(j)) est vraie :

$$\forall s \in S[CIP], P(s) \Rightarrow \{\forall w \in T_s, \exists w' \in T_s, (w \leqslant w') \wedge (\exists j \in dom(w'), Q(w'(j)))\}$$

soit $\mathcal{M}_{WF} \models P \rightsquigarrow Q$. Ce qui termine la correction de cette règle.

Règle 4

On suppose que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$ (1)

$\mathcal{M}_{WF} \models Q \Rightarrow R$ (2).

A partir de (1), on établit la propriété :

$$\forall s \in S[CIP], P(s) \Rightarrow [\forall w \in T_s, \exists w' \in T_s, (w \leqslant w') \wedge (\exists j \in dom(w'), Q(w'(j)))]$$

et à partir de (2) :

$\forall s \in S[CIP]\ Q(s) \Rightarrow R(s)$.

On en déduit que :

$(\exists j \in dom(w'), Q(w'(j))) \Rightarrow (\exists j \in dom(w'), R(w'(j)))$.

Soit

$$\forall s \in S[CIP], P(s) \Rightarrow [\forall w \in T_s, \exists w' \in T_s, (w \leqslant w') \wedge (\exists j \in dom(w'), R(w'(j)))]$$

Ce qui s'écrit de la façon suivante :

$\mathcal{M}_{WF} \models P \rightsquigarrow R$.

Nous terminons ce paragraphe par la remarque qui suit : dans la correction de chaque règle ci-dessus l'hypothèse d'équité faible n'a pas été utile et notre correction aurait pu être faite avec un modèle de fatalité de CIP quelconque.

III.2.4.- Justification des règles auxiliaires

Notre preuve ici consistera en fait en une justification s'appuyant sur certaines notions et certains choix que nous avons faits. En fait, nous avons voulu écarter de OL la déduction de propriétés comme celles d'implications ou celles d'invariance.

Règle 1A

Nous voulons construire un système de preuve qui soit correct et complet mais nous ne pourrons résoudre l'incomplétude relative à l'implication car Gödel a démontré que toute théorie axiomatisée de $\mathbb{N}$ (et toute extension) est incomplète (cf. SHOENFIELD [SHO]). De plus, notre règle est correcte à cause de l'extension de la notion de validité suivant un modèle de fatalité ; en effet, si $\{\forall s \in S[CIP], P(s) \Rightarrow Q(s)\}$, alors $\mathcal{M}_{WF} \models P \Rightarrow Q$. Cette règle nous semble utile pour distinguer la déduction dans OL de l'établissement de l'implication.

Règle 2A

La méthode proposée par OWICKI et LAMPORT [OWL] utilisait des propriétés d'invariance et la preuve de ces propriétés était déduite par l'intermédiaire de la méthode de LAMPORT [LAM2] et un"savant amalgame" de propriétés exprimées par l'opérateur modal temporel $\Box$. Nous pensons que de préconiser l'emploi d'un

système intuitivement correct et complet comme celui de LAMPORT [LAM2] cela conduit à un pseudoformalisme peu fiable. Afin d'assurer une complétude sémantique à OL, il faut que nous puissions prouver toutes les propriétés d'invariance de la règle 2 et ceci impose quelques conditions au langage d'assertions $\mathcal{A}$. Nous supposons que $\mathcal{A}[CIP]$ est suffisamment expressif au sens de COOK [COO] et, plus simplement nous supposons qu'il existe toujours une assertion invariante pour établir l'un des prémisses de la règle 2, d'ailleurs l'étude de COUSOT-COUSOT [COC1], COUSOT [COU] explique clairement la condition de COOK [COO]. La validité de notre règle est équivalente à la complétude sémantique de notre système de preuve ; cette règle permet d'abstraire la preuve d'invariance indépendamment d'un système de preuve. Notre choix se portera, en pratique, vers la méthode d'OWICKI-GRIES [OWG] qui a le mérite d'avoir été justifiée en détails.

Règle 3A

Comme les deux précédentes règles, celle-ci est, en quelque sorte, indépendante du système proprement dit et est utilisée en pratique pour tenir compte d'une hypothèse. On peut remplacer de façon équivalente par la notation $\{H\}_{OL} \vdash C$ où H est une hypothèse du type $P \rightsquigarrow Q$ et C une conclusion déduite à partir de H et de OL. Cette règle permet de prendre en compte une hypothèse comme, par exemple, pour le cas d'un algorithme d'exclusion mutuelle on suppose que tout processus en section critique en sortira un jour. Mais la validité de cette règle repose sur la notion d'hypothèse et celle-ci doit être cohérente et adéquate au cas considéré.

III.3.- COMPLETUDE SEMANTIQUE

Théorème [III.3].1 :

Soient CIP un programme parallèle quelconque,

$\mathcal{A}[CIP]$ un langage d'assertions pour CIP,

$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable,

P, Q deux assertions de $\mathcal{A}[CIP]$.

Si $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, alors $_{OL} \vdash P \rightsquigarrow Q$.

La preuve de ce théorème nécessite de donner des assertions intermédiaires telles que l'on puisse utiliser les règles et axiomes de OL. Cette preuve a été ébauchée dans les théorèmes [II.6].1, [II.6].2, [II.6].3.

Preuve :

Soient P, Q deux assertions de $\mathcal{A}[CIP]$ telles que :

$\mathcal{M}_{WF} \models P \rightsquigarrow Q$. D'après le théorème [II.6].3, il existe une suite d'assertions $(R(\alpha))_{\alpha \leq K}$ $\mathcal{U}[CIP]$ telles que :

(1) $\mathcal{M}_{WF} \models P \rightsquigarrow \exists \alpha R(\alpha)$.

(2) $\mathcal{M}_{WF} \models R(\alpha) \rightsquigarrow \exists \beta < \alpha, R(\beta)$, $\forall \alpha > 0$.

(3) $\mathcal{M}_{WF} \models R(0) \rightsquigarrow Q$.

et cette suite d'assertions dépend de l'opérateur Γ_Q dont l'étude a été faite dans le théorème [II.6].1 et le théorème [II.6].2.

Notre preuve consiste à prouver les trois lemmes suivants :

Lemme 1 :

$$_{OL}\vdash P \rightsquigarrow \exists\alpha R(\alpha)$$

Lemme 2 :

$$\forall\alpha > 0,\ \alpha \leqslant \kappa^{\mathcal{U}[CIP]}$$

$$_{OL}\vdash R(\alpha) \rightsquigarrow \exists\beta < \alpha, R(\beta).$$

Lemme 3 :

$$_{OL}\vdash R(0) \rightsquigarrow Q.$$

On fera dans ce qui suit la preuve de chaque lemme mais nous donnons dès à présent la fin de la preuve du théorème de complétude sémantique en utilisant ces lemmes :

En utilisant la règle 3 du système OL avec comme prémisses à cette règle les lemmes 1, 2 et 3 on en déduit $_{OL}\vdash P \rightsquigarrow Q$. □

III.3.1.- Preuve du lemme 1

Par construction de $(R(\alpha))_{\alpha\leqslant\kappa^{\mathcal{U}[CIP]}}$ et par hypothèse $(\mathcal{M}_{WF} \models P \rightsquigarrow$ on en déduit que :

$$\forall s \in S[CIP],\ P(s) \Rightarrow (\bigvee_{\alpha\leqslant\kappa^{\mathcal{U}[CIP]}} R(\alpha))(s)$$

ce que l'on écrit de façon équivalente :

$$\forall s \in S[CIP],\ P(s) \Rightarrow (\exists\alpha R(\alpha))(s).$$

On exprime ainsi que $P \Rightarrow \exists\alpha R(\alpha)$ est vraie pour CIP.

Par la règle 1A, on en déduit : $_{OL}\vdash P \Rightarrow \exists\alpha R(\alpha)$.

Par la règle 1, on en déduit ensuite :

$$_{OL}\vdash P \rightsquigarrow \exists\alpha\ R(\alpha).$$

III.3.2.- Preuve du lemme 2

Nous aurons besoin de résultats supplémentaires pour démontrer ce lemme 2 et ils seront prouvés ultérieurement mais leur signification est très intuitive.

(1) Transitivité de la relation de conduite

Règle 5 :

si $_{OL}\vdash U \rightsquigarrow V$, et, $_{OL}\vdash V \rightsquigarrow W$, alors $_{OL}\vdash U \rightsquigarrow W$.

(2) Propriété de localisation

La propriété suivante est vraie pour le programme CIP :

$$[\underline{at}\ Start]v[\underline{at}\ Finish]v[\underline{after}\ Finish]v \bigvee_{(a_1,\ldots,a_n)\in A_1[CIP]\times\ldots\times A_n[CIP]} [\underline{at}\ a_1\wedge\ldots\wedge\underline{at}\ a_n]$$

où $A_i[CIP]$ est l'ensemble des actions atomiques constituant le ième processus formant CIP ($A_i[CIP] = A[IPS_i]$).

La disjonction est exclusive.

(3) Equité faible des actions atomiques de CIP

Pour chaque action atomique a de CIP (ie : $a \in A[CIP]$),

$$_{OL}\vdash \underline{at}\ a \rightsquigarrow \underline{jafter}\ a$$

(4) Propriété d'exclusion de Start

$\forall a \in A[CIP]$, $(a \neq Start) \Rightarrow \sim (\underline{at}\ a \wedge \underline{at}\ Start)$ est vraie pour CIP.

(5) Propriété d'exclusion de Finish

$\forall a \in A[CIP]$, $(a \neq Finish) \Rightarrow \sim (\underline{at}\ a \wedge \underline{at}\ Finish)$ est vraie pour CIP.

(6) Règle de convergence (Règle 6)

Soit I un ensemble au plus dénombrable.

$(R_i)_{i\in I}$ une famille d'assertions de $\mathfrak{A}[C\mathbb{P}]$.

Règle 6 :

Si $_{OL}\vdash R \rightsquigarrow [\bigvee_{i\in I} R_i]$,

et $_{OL}\vdash R_i \rightsquigarrow T$, $\forall i \in I$,

alors $_{OL}\vdash R \rightsquigarrow T$.

Nous pouvons commencer la preuve : soit α un ordinal plus petit que $\kappa^{\mathfrak{U}[C\mathbb{P}]}$ et non nul.

$$R(\alpha)\Rightarrow\Big[(\underline{at}\ \mathrm{Start}\wedge R(\alpha))\vee(\underline{at}\ \mathrm{Finish}\wedge R(\alpha))\vee(\underline{after}\ \mathrm{Finish}\wedge R(\alpha))\vee \bigvee_{(a_1,\ldots,a_n)\in A[\mathbb{P}S_1]\times\ldots\times A[\mathbb{P}S_n]} (\underline{at}\ a_1\wedge\ldots\wedge\underline{at}\ a_n\wedge R(\alpha))\Big]$$

est vraie pour $C\mathbb{P}$. On en déduit à l'aide de 1A,

$$_{OL}\vdash R(\alpha)\Rightarrow\Big[(\underline{at}\ \mathrm{Start}\wedge R(\alpha))\vee(\underline{at}\ \mathrm{Finish}\wedge R(\alpha)\vee(\underline{after}\ \mathrm{Finish}\wedge R(\alpha))\vee \bigvee_{(a_1,\ldots,a_n)\in A[\mathbb{P}S_1]\times\ldots\times A[\mathbb{P}S_n]} (\underline{at}\ a_1\wedge\ldots\wedge\underline{at}\ a_n\wedge R(\alpha))\Big]$$

En appliquant la règle 1, on en déduit :

$$_{OL}\vdash R(\alpha)\rightsquigarrow\Big[(\underline{at}\ \mathrm{Start}\wedge R(\alpha))\vee(\underline{at}\ \mathrm{Finish}\wedge R(\alpha))\vee(\underline{after}\ \mathrm{Finish}\wedge R(\alpha))\vee \bigvee_{(a_1,\ldots,a_n)\in A[\mathbb{P}S_1]\times\ldots\times A[\mathbb{P}S_n]} (\underline{at}\ a_1\wedge\ldots\wedge\underline{at}\ a_n\wedge R(\alpha))\Big]$$

Nous allons procéder par cas comme nous le permet le théorème ci-dessus et la propriété préliminaire (6).

Cas 1 :

$$_{OL}\vdash [\underline{at}\ \mathrm{Start}\wedge R(\alpha)]\rightsquigarrow(\exists\beta<\alpha, R(\alpha)).$$

Nous utilisons dans ce cas la règle 3 qui nécessite l'emploi de la règle 3A et l'invention d'un invariant.

$I_{\mathrm{Start}} \equiv \rho[C\mathbb{P}](\tilde{I}_{\mathrm{Start}})$ où $\tilde{I}_{\mathrm{Start}}$ est tel que :

$$\tilde{I}_{\mathrm{Start}} = \{s\in S[C\mathbb{P}]/(\exists \underline{s}\in S[C\mathbb{P}],[t^*[C\mathbb{P}](\underline{s},s)]\wedge[(\underline{at}\ \mathrm{Start}\wedge R(\alpha))(\underline{s})] \wedge[(\underline{at}\ \mathrm{Start}\wedge R(\alpha))(s)\Rightarrow(\forall s'\in S[C\mathbb{P}],t[C\mathbb{P}](s,s')\Rightarrow(s'\in\Gamma_Q^\alpha))] \wedge[(\underline{jafter}\ \mathrm{Start})(s)\Rightarrow(s\in\Gamma_Q^\beta)\wedge(\beta<\alpha)]\}.$$

On vérifie que :

(1) $(\underline{at}\ \mathrm{Start}\wedge R(\alpha))\Rightarrow I_{\mathrm{Start}}$ est vrai pour $C\mathbb{P}$ par construction de Γ_Q et de $R(\alpha)$.

(2) Start est la seule action à exécuter et si on l'exécute, par construction de Γ_Q et de $R(\alpha)$, on décrémente strictement l'ordinal et l'état obtenu conduit Q, mais si l'on fait n'importe quelle transition ensuite jafter Start est faux et le reste :

$$\forall(s,s')\in S[C\mathbb{P}],[I_{\mathrm{Start}}(s)\wedge t[C\mathbb{P}](s,s')]\Rightarrow I_{\mathrm{Start}}(s').$$

2.1 : Soit s tel que

$$\exists\underline{s}\in S[C\mathbb{P}],[t^*[C\mathbb{P}](\underline{s},s)]\wedge[\underline{at}\ \mathrm{Start}\wedge R(\alpha))(\underline{s})] \wedge[(\underline{at}\ \mathrm{Start}\wedge R(\alpha))(s)\Rightarrow(\forall s'\in S[C\mathbb{P}],t[C\mathbb{P}](s,s')\Rightarrow(s'\in\Gamma_Q^\alpha))] \wedge[(\underline{jafter}\ \mathrm{Start})(s)\Rightarrow(s\in\Gamma_Q^{\beta,\beta<\alpha})]$$

et $(\underline{at}\ \mathrm{Start}\wedge R(\alpha))(s)$ est vrai.

Soit $s' \in S[C\mathbb{P}]$ tel que $t[C\mathbb{P}](s,s')$: on distingue deux cas

(at Start)(s') ou (jafter Start)(s').

Alors

$t^*[C\mathbb{P}](s,s') \wedge (at\ a \wedge R(\alpha))(s) \wedge (at\ Start)(s') \wedge R(\alpha)(s')$

ou $t^*[C\mathbb{P}](s,s') \wedge (at\ a \wedge R(\alpha))(s) \wedge (jafter\ Start)(s') \wedge (\beta<\alpha) \wedge (s' \in \Gamma_Q^\beta)$.

Ce qui revient à $I_{Start}(s')$.

2.2 : On suppose que $I_{Start}(s)$ est vrai et (jafter Start)(s). Dans ce cas, $(s \in \Gamma_Q^\beta) \wedge (\beta<\alpha)$,

Soit $s' \in S[C\mathbb{P}]$ tel que $t[C\mathbb{P}](s,s')$. Dans ce cas (jafter Start)(s

est faux et (at Start)(s') aussi.

De plus, $\exists s \in S[C\mathbb{P}], t^*[C\mathbb{P}](s,s') \wedge (at\ Start \wedge R(\alpha))(s)$.

On a établi $I_{Start}(s')$.

2.3 : On suppose que $I_{Start}(s)$ et $(\sim at\ Start)(s), (\sim jafter\ Start)(s)$.

Alors toute transition ne modifie pas I_{Start}.

On a établi que I_{Start} était invariant.

(3) $[I_{Start} \wedge jafter\ Start] \Rightarrow (\exists \beta < \alpha, R(\alpha))$

car $\bigvee_{\beta<\alpha} R(\beta) \equiv \bigvee_{\beta<\alpha} \rho[C\mathbb{P}](\Gamma_Q^\beta)$.

On utilise la règle 2A pour déduire :

$$OL \vdash \begin{array}{|l} at\ Start \wedge R(\alpha) \Rightarrow I_{Start} \\ I_{Start} \Rightarrow OI_{Start} \\ jafter\ Start \wedge I_{Start} \Rightarrow \exists \beta < \alpha, R(\beta) \end{array}$$

De plus, on dispose de propriétés préliminaires et nous utilisons la propriété (3) :

$OL \vdash at\ Start \rightsquigarrow jafter\ Start$

On déduit avec la règle 2 que :

$OL \vdash (at\ Start \wedge R(\alpha)) \rightsquigarrow jafter\ Start \wedge (\exists \beta < \alpha, R(\beta))$.

Or

$OL \vdash (jafter\ Start) \wedge (\exists \beta < \alpha, R(\beta)) \Rightarrow (\exists \beta < \alpha, R(\beta))$.

Par la règle 4, on déduit :

$OL \vdash at\ a \wedge R(\alpha) \rightsquigarrow \exists \beta < \alpha, R(\beta)$.

On aurait pu utiliser l'axiome F9 et prouver directement ce lemme.

Cas 2 :

$OL \vdash (at\ Finish \wedge R(\alpha)) \rightsquigarrow (\exists \beta < \alpha, R(\beta))$.

On utilise ici l'axiome F10 et la règle 6 :

$R(\alpha)$ est nécessairement $R(1)$, sinon :

- si $\alpha > 1$, cela signifie qu'il reste plus d'une action atomique à exécuter or c'est faux ; dans ce cas

  $\alpha > 1$, $at\ Finish \wedge R(\alpha) \Rightarrow false$

  et $false \Rightarrow \exists \beta < \alpha, R(\beta)$

  ce qui prouve le cas 2.

- si $\alpha = 1$, par l'axiome F10 et la règle 6 :

  $OL \vdash (at\ Finish \wedge R(\alpha)) \rightsquigarrow (jafter\ Finish \wedge P')$,

  où $P' \equiv R(\alpha)\left[\frac{jafter}{at\ Finish}\right]$

Par définition de $R(\alpha)(\alpha = 1)$, $P' \equiv R(0)$.

D'où :

$OL \vdash (\underline{at}\ \text{Finish} \wedge R(\alpha)) \rightsquigarrow \exists \beta < \alpha, R(0)$.

Cas 3 :

$OL \vdash (\underline{after}\ \text{Finish} \wedge R(\alpha)) \rightsquigarrow \exists \beta \wedge \alpha, R(\beta)$.

Dans ce cas, after Finish est vrai et $R(\alpha)$ aussi : il reste des actio atomiques à exécuter, ce qui est absurde :

$OL \vdash \underline{after}\ \text{Finish} \wedge R(\alpha) \Rightarrow \text{false}$

D'où :

$OL \vdash (\underline{after}\ \text{Finish} \wedge R(\alpha)) \rightsquigarrow \exists \beta < \alpha, R(\beta)$.

Cas 4 :

$OL \vdash (\underline{at}\ a_1 \wedge \ldots \wedge \underline{at}\ a_n \wedge R(\alpha)) \rightsquigarrow (\exists \beta < \alpha, R(\beta))$.

Par construction de Γ_Q et de $R(\alpha)$, nous déduisons qu'il existe une action atomique a dans $\{a_1, \ldots, a_n\}$ telle que, en s,

$$[\exists s' \in S[C\mathbb{P}], (\underline{at}\ a)(s) \wedge t[C\mathbb{P}](s,s') \wedge (\underline{jafter}\ a)(s') \wedge R(\alpha)(s) \wedge (s' \in \Gamma_Q^{\beta}) \wedge (\beta < \alpha)$$

$$\wedge [\forall s' \in S[C\mathbb{P}], \{(\underline{at}\ a)(s) \wedge t^*[C\mathbb{P}](s,s') \wedge (\underline{at}\ a)([s,s'[) \wedge R(\alpha)(s)\}$$

$$\Rightarrow \{(\beta < \alpha) \wedge (s' \in \Gamma^{\beta})\}]$$

L'action a conduit à une décrémentation de α, si elle est exécutée et dans un autre cas, stagner et ne point décrémenter, a est distincte de Start et Finish, sinon on utilise les préliminaires :

$(\underline{at}\ a_1 \wedge \ldots \wedge \underline{at}\ a_n \wedge R(\alpha)) \Rightarrow (\underline{at}\ a \wedge R(\alpha))$ (par propriété de $\Rightarrow$ et $\wedge$)

$OL \vdash (\underline{at}\ a_1 \wedge \ldots \wedge \underline{at}\ a_n \wedge R(\alpha)) \Rightarrow (\underline{at}\ a \wedge R(\alpha))$ (par la règle 1A)

$OL \vdash (\underline{at}\ a_1 \wedge \ldots \wedge \underline{at}\ a_n \wedge R(\alpha)) \rightsquigarrow (\underline{at}\ a \wedge R(\alpha))$ (par la règle 1).

Nous prouvons à l'aide des règles 2 et 2A que

$OL \vdash (\underline{at}\ a \wedge R(\alpha)) \rightsquigarrow \exists \beta < \alpha, R(\beta)$.

Pour cela, il nous faut inventer un invariant I_a^{α} tel que :

$$OL \vdash \boxed{\begin{array}{l} \underline{at}\ a \wedge R(\alpha) \Rightarrow I_a^{\alpha} \\ I_a^{\alpha} \quad \Rightarrow \quad OI_a^{\alpha} \\ (I_a^{\alpha} \wedge \underline{jafter}\ a) \Rightarrow (\exists \beta < \alpha, R(\beta)) \end{array}}$$

L'invariant que nous proposons exprime le fait que l'on provient d'un état qui satisfait $\underline{at}\ a \wedge R(\alpha)$ et que l'on va atteindre un état satisfaisant jafter a, ou qu'on atteint ou que l'on a atteint.

$$I_a^{\alpha}(s) \equiv \left[\begin{array}{l} \exists (\underline{s},s') \in S^2[C\mathbb{P}], t^*[C\mathbb{P}](\underline{s},s') \wedge t^*[C\mathbb{P}](\underline{s},s) \wedge [(\underline{at}\ a \wedge R(\beta))([\underline{s},s'[) \wedge (\beta \leq \alpha)] \\ \qquad \wedge [(\underline{jafter}\ a)(s') \wedge R(\beta)(s') \wedge (\beta < \alpha)] \wedge \\ \qquad \wedge [(t^*[C\mathbb{P}](\underline{s},s) \wedge t^*[C\mathbb{P}](s,s')) \Rightarrow (\underline{at}\ a \wedge R(\beta) \wedge (\beta \leq \alpha))(s)] \wedge \\ \qquad \wedge [t^*[C\mathbb{P}](s',s) \wedge (\underline{jafter}\ a)(s) \wedge (s \notin R(\beta) \wedge \beta \leq \alpha) \Rightarrow \underline{false}\ ] \\ \qquad \wedge [t^*[C\mathbb{P}](s',s) \wedge (\underline{jafter}\ a)(s) \wedge (s \in R(\beta)) \wedge (\beta < \alpha) \Rightarrow (s = s')] \end{array}\right]$$

Nous vérifions que I_a^{α} réalise les trois conditions du rectangle $\boxed{\ }$.

(1) Par construction de $R(\alpha)$, il existe s' tel que

$(\underline{jafter}\ a)(s') \wedge t^*[C\mathbb{P}](s,s') \wedge (\underline{at}\ a)([s,s'[) \wedge R(\alpha)(s)$

$\wedge (R(\beta)([s,s'[) \wedge \beta < \alpha) \wedge (R(\beta)(s') \wedge \beta < \alpha)$.

D'où : $\underline{at}\ a \wedge R(\alpha) \Rightarrow I_a^{\alpha}$ est vraie pour $C\mathbb{P}$.

Le fait d'avoir false signifie que l'exécution de $C\mathbb{P}$ peut se poursuivre

au delà de Q, sans garantir Q sous hypothèse d'équité faible par la suite.

(2) Il faut prouver que $I_a^\alpha \Rightarrow 0I_a^\alpha$ est vraie pour CIP.

Pour cela, on distingue les différents cas :

Cas 1 :

(at a∧R(β))(s)∧(β<α)∧I_a^α(s).

Soit s' quelconque de S[CIP] tel que t[CIP](s,s').

① (jafter a)(s') est vrai

Dans ce cas, on en déduit :

$\exists(s,s'')\in S^2[CIP], t^*[CIP](s,s)\wedge t^*[CIP](s,s'')\wedge$
$[(at\ a\wedge(R(\beta)))([s,s''[)\wedge\beta\leq\alpha]\wedge$
$(jafter\ a)(s'')\wedge t[CIP](s,s')\wedge(s'=s'')\wedge$
$(jafter\ a)(s')\wedge R(\beta)(s')\wedge(\beta<\alpha)$

Ce qui implique $I_a^\alpha(s')$.

② (at a)(s') est vrai

Puisque (at a∧R(β))(s)∧(β≤α)∧I_a^α(s), alors (at a∧R(β))(s')∧(β≤α), a n'a pas été exécuté, donc α stagne. On en déduit $I_a^\alpha(s')$.

③ Ces deux cas ① et ② sont les seuls, à cause de la stagnation de α.

Cas 2 :

(jafter a∧R(β))(s)∧(β<α)∧I_a^α(s).

Soit s'∈S[CIP] tel que t[CIP](s,s').

Dans ce cas, (jafter a)(s') est faux et il existe s_1, s_2 tel que

$t^*[CIP](s_1,s')\wedge(jafter\ a)(s_1)\wedge(\beta<\alpha)\wedge R(\beta)(s_1)$

$\wedge t^*[CIP](s_2,s_1)\wedge(at\ a\wedge R(\beta)\wedge\beta\leq\alpha)([s_2,s_1[)$.

Ce qui conduit à déduire $I_a^\alpha(s')$.

Cas 3 :

(jafter a)(s)∧(s∉R(β))∧(β≤α)∧I_a^α(s)

Dans ce cas, ceci est équivalent à false ; or false implique $I_a^\alpha(s')$ pour tout s' tel que t[CIP](s,s').

Cas 4 :

Il existe s_1 tel que $t[CIP](s_1,s)\wedge(jafter\ a)(s_1)\wedge(R(\beta)\wedge\beta<\alpha)(s_1)\wedge I_a^\alpha(s)$.

Soit s' tel que t[CIP](s,s').

Dans ce cas, s' vérifie encore $I_a^\alpha(s')$ par transitivité de $t^*[CIP]$.

On en déduit que :

$\forall(s,s')\in S^2[CIP], I_a^\alpha(s)\wedge t[CIP](s,s') \Rightarrow I_a^\alpha(s')$.

(3) $I_a^\alpha \wedge jafter\ a \Rightarrow \exists\beta < \alpha, R(\beta)$.

En effet, ou (jafter a)(s) ∧ (R(β)∧(β<α))(s) est vrai ou (jafter a)(s)∧(~ R(β)∧β≤α)(s) est vrai.

Dans les deux cas, cela implique $\exists\beta < \alpha, R(\beta)$.

A l'aide de la propriété (3) citée au début et de I_a^α on déduit avec la règle 2 :

$0_L \vdash (at\ a\wedge R(\alpha)) \rightsquigarrow jafter\ a\wedge\exists\beta < \alpha, R(\beta)$

Or :

$jafter\ a\wedge\exists\beta < \alpha, R(\beta) \Rightarrow \exists\beta \wedge \alpha, R(\beta)$.

Par la règle 1A et la règle 4 on déduit :

$_{OL}\vdash \underline{at}\ a\wedge R(\alpha) \rightsquigarrow \exists\beta < \alpha, R(\beta)$

A l'aide de la propriété (1), on en déduit :

$_{OL}\vdash \underline{at}\ a_1\wedge\ldots\wedge\underline{at}\ a_n\wedge R(\alpha)\rightsquigarrow \exists\beta < \alpha, R(\beta)$.

A l'aide de la règle 6, en utilisant les cas 1, 2, 3 on déduit :

$_{OL}\vdash R(\alpha)\rightsquigarrow \exists\beta < \alpha, R(\beta)$.

III.3.3.- Preuve du lemme 3

$R(0) \equiv \Gamma^0_Q \equiv Q$.

On en déduit :

$_{OL}\vdash R(0) \Rightarrow Q$ par la règle 1A

$_{OL}\vdash R(0)\rightsquigarrow Q$ par la règle 1.

III.3.4.- Preuves des propriétés intermédiaires

Nous résumons ces propriétés dans quelques lemmes.

Lemme 2.1 (Règle 6)

Si I est un ensemble dénombrable, P', Q', R'_i des assertions de $\mathfrak{A}$[CI

si $_{OL}\vdash P'\rightsquigarrow \bigvee_{i\in I} R'_i$, $_{OL}\vdash R'_i\rightsquigarrow Q'$, alors $_{OL}\vdash P'\rightsquigarrow Q'$.

Preuve du lemme 2.1

Posons $S(0) = Q'$, $S(i) = R_i$, pour $i \geqslant 1$,

$S(\omega) \equiv P'$ et $\forall\alpha \geqslant \omega$, $S(\alpha) \equiv P'$.

On vérifie les relations :

$P' \Rightarrow S(\omega)$

$P' \Rightarrow \bigvee_{\alpha\in Ord} S(\alpha)$ } en utilisant les propriétés de l'implication.

D'où $P' \Rightarrow \exists\alpha S(\alpha)$ et par la règle 1A et 1, on déduit

(1) $_{OL}\vdash P'\Rightarrow \exists\alpha\ S(\alpha)$.

Soit $\alpha > 0$ et $\alpha \in Ord$:

Par hypothèse :

si $\alpha = \omega$, ou $\alpha \geqslant \omega$, $_{OL}\vdash S(\alpha)\rightsquigarrow \bigvee_{\alpha'<\omega} S(\alpha')$

ce que l'on écrit encore $_{OL}\vdash S(\alpha)\rightsquigarrow \exists\beta < \alpha\ S(\beta)$.

si $\alpha > 0$ et $\alpha < \omega$, $_{OL}\vdash S(\alpha)\rightsquigarrow S(0)$

et $S(0) \Rightarrow \bigvee_{\alpha'<\alpha} S(\alpha')$

d'où par la règle 1A et 4, on déduit

$_{OL}\vdash S(\alpha)\rightsquigarrow \exists\beta < \alpha\ S(\beta)$.

(2) $\forall\alpha > 0$, $\alpha \in Ord$, $_{OL}\vdash S(\alpha)\rightsquigarrow \exists\beta < \alpha, S(\beta)$.

et trivialement :

(3) $_{OL}\vdash S(0)\rightsquigarrow Q'$.

On applique la règle 3 avec les prémisses (1), (2), (3) pour déduire :

$_{OL}\vdash P'\rightsquigarrow Q'$.

Fin de la preuve du lemme 2.1.

Lemme 2.2

Soit P, Q, R_1, R_2, R des assertions de $\mathfrak{A}$[CIP].

1 - Si $_{OL}\vdash P\rightsquigarrow R$, $_{OL}\vdash R\rightsquigarrow Q$, alors $_{OL}\vdash P\rightsquigarrow Q$.

2 - Si $_{OL}\vdash P \rightsquigarrow (R_1 \vee R_2)$, $_{OL}\vdash R_1 \rightsquigarrow Q$, $_{OL}\vdash R_2 \rightsquigarrow Q$,
alors $_{OL}\vdash P \rightsquigarrow Q$.

Preuve du lemme 2.2

1 - On applique le lemme 2.1, pour $I = \{1\}$.

2 - On applique le lemme 2.1, pour $I = \{1,2\}$.

Fin de la preuve du lemme 2.2.

Lemme 2.3

1 - $\forall a \in A[C\mathbb{P}]$, $(a \neq Start) \Rightarrow \sim(\underline{at}\ a \wedge \underline{at}\ Start)$

2 - $\forall a \in A[C\mathbb{P}]$, $(a \neq Finish) \Rightarrow \sim(\underline{at}\ a \wedge \underline{at}\ Finish)$

3 - $\bigvee_{a\in A[C\mathbb{P}]} \underline{at}\ a$ est vraie pour $C\mathbb{P}$

4 - $\forall (a,b) \in A[C\mathbb{P}]^2$, $[(a \neq Start)\wedge(b \neq Start)\wedge(a \neq Finish)\wedge(b \neq Finish)]$
$\Rightarrow$
$[\sim(\underline{at}\ a \wedge \underline{at}\ b) \Leftrightarrow (\exists i \in [n], (a \in A[\mathbb{P}S_i])\wedge(b \in A[\mathbb{P}S_i])]$

5 - $_{OL}\vdash \underline{at}\ a \rightsquigarrow \underline{jafter}\ a$, $\forall a \in A[C\mathbb{P}]$.

6 - $\underline{at}\ Finish \vee \underline{at}\ Start \vee \underline{after}\ Finish \vee [\bigvee_{(a_1,\ldots,a_n)\in \overline{A[\mathbb{P}S_1]\times\ldots\times A[\mathbb{P}S_n]}} [\underline{at}\ a_1 \wedge \ldots \wedge \underline{at}\ a_n]]$
est vrai pour $C\mathbb{P}$.

Preuve du lemme 2.3

1 - Les assertions de contrôle $\underline{at}\ a$ pour chaque action atomique a de

ont été définies par :

$\underline{at}\ a \equiv \rho[C\mathbb{P}](AV(a))$.

Soit $a \in A[C\mathbb{P}]$ telle que $a \neq Start$ et $s \in S[C\mathbb{P}]$.

$(\underline{at}\ a \wedge \underline{at}\ Start)(s)$ ssi $s \in AV(a) \cap AV(Start)$.

$s \in AV(a)\cap AV(Start)$ ssi $(s=(\underline{\ell}_1,m))\wedge[((s=((\ell_1,\ldots,\ell_n),m)))\vee(s=(\bar{\ell}_2,m))]$

Or $\underline{\ell}_1 \neq \bar{\ell}_2$ et $\underline{\ell}_1 \neq (\ell_1,\ldots,\ell_n)$.

On en déduit que $AV(a)\cap AV(Start) = \emptyset$.

D'où $\sim(\underline{at}\ a \wedge \underline{at}\ Start)$ est vraie pour tout état s de $S[C\mathbb{P}]$.

2 - On procède de même en remplaçant Start par Finish et $\bar{\ell}_2$ par $\underline{\ell}_1$, $\underline{\ell}_1$ par $\bar{\ell}_2$.

3 - Cette propriété est vraie, si on suppose que être après $C\mathbb{P}$ c'est être avant skip. Dans cette condition-là, on en déduit que $\rho[C\mathbb{P}](S[C\mathbb{P}])$ est vraie pour tout état de $C\mathbb{P}$. On peut décomposer $\rho[C\mathbb{P}](S[C\mathbb{P}])$ suivant les étiquettes : $\rho[C\mathbb{P}](S[C\mathbb{P}]) \equiv \bigvee_{a\in A[C\mathbb{P}]} \underline{at}\ a$.

4 - Soient a, b deux actions atomiques de $C\mathbb{P}$ distinctes de Start et Finish.

$(\underline{at}\ a \wedge \underline{at}\ b)(s)$ ssi $s \in AV(a)\cap AV(b)$.

Supposons que a et b appartiennent au même programme séquentiel composant $C\mathbb{P}$.

$s \in AV(a)\cap AV(b)$ ssi $s = ((\ell_1,\ldots,\ell_n),m)\wedge$
$s = ((\ell'_1,\ldots,\ell'_n),m)\wedge$
$(\forall j \in [n] \setminus \{i\}, \ell'_j = \ell_j)\wedge$
$(\ell_i \neq \ell'_i)$.

On en déduit : $AV(a)\cap AV(b) = \emptyset$.

D'où $\sim(\underline{at}\ a \wedge \underline{at}\ b)$ est vrai pour CP.

5 - On procède par cas.

5.1 $a \in \{v:=e, v:=?, \underline{skip}\}$

Par définition de $\underline{at}\ a$ et $\underline{jafter}\ a$:

$\underline{et}\ a \equiv \underline{at}\ S$ et $\underline{jafter}\ a \equiv \underline{jafter}\ S$ où $S \in \{v:=e, v:=?, \underline{skip}\}$.

$OL \vdash \underline{at}\ S \rightsquigarrow \underline{jafter}\ S$ est un axiome.

5.2 $a = \langle B \rangle$ et $S \equiv \underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}$

$\underline{at}\ a \equiv \underline{at}\ S$ et $\underline{jafter}\ a \equiv \underline{jat}\ S_1 \wedge B \vee \underline{jat}\ S_2 \wedge \sim B$.

On en déduit par l'axiome F4,

$OL \vdash \underline{at}\ a \rightsquigarrow \underline{jafter}\ a$.

5.3 $a = \langle B \rangle$ et $S \equiv \underline{while}\ B\ \underline{do}\ S'\ \underline{od}$

$\underline{at}\ a \equiv \underline{at}\ S \vee \underline{after}\ S'$ et $\underline{jafter}\ a \equiv \underline{jat}\ S' \wedge B \vee \underline{jafter}\ S \wedge \sim B$.

D'après les axiomes F7 et F8 :

$OL \vdash \underline{at}\ S \rightsquigarrow \underline{jafter}\ a$

$OL \vdash \underline{after}\ S' \rightsquigarrow \underline{jafter}\ a$.

Or $\underline{at}\ a \Rightarrow \underline{at}\ S \vee \underline{after}\ S'$, soit

$OL \vdash \underline{at}\ a \rightsquigarrow \underline{at}\ S \vee \underline{after}\ S'$.

D'après le lemme 2.2, on en déduit :

$OL \vdash \underline{at}\ a \rightsquigarrow \underline{jafter}\ a$.

5.4 $((a = \text{Out-if1})\ \underline{ou}\ (a = \text{Out-if2}))\ \underline{et}\ S \equiv \underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}$

$\underline{at}\ a \equiv \underline{at}$ Out-if1 (resp. $\underline{at}$ Out-if2) et $\underline{jafter}\ S^{(1)} \equiv \underline{jafter}\ a$.

Par les axiomes F5, F6, on en déduit :

$OL \vdash \underline{at}\ a \rightsquigarrow \underline{jafter}\ a$.

5.6 $(a = \text{Start})\ \underline{et}\ CP \equiv \underline{Cobegin}\ P S_1 \| \ldots \| P S_n\ \underline{coend}$

$\underline{at}\ a \equiv \underline{at}\ CP$ et $\underline{jafter}\ a \equiv \underline{jat}\ P S_1 \wedge \ldots \wedge \underline{jat}\ P S_n$.

$OL \vdash \underline{at}\ a \rightsquigarrow \underline{jafter}\ a$ par l'axiome F9.

5.7 $(a = \text{Finish})\ \underline{et}\ CP \equiv \underline{Cobegin}\ P S_1 \| \ldots \| P S_n\ \underline{coend}$

$\underline{at}\ a \equiv \bigvee_{i=1}^{n} [\underline{after}\ P S_1 \wedge \ldots \wedge \underline{jafter}\ P S_i \wedge \ldots \wedge \underline{after}\ P S_n]$

$\underline{jafter}\ a \equiv \underline{jafter}\ CP$

On applique le lemme 2.2 et l'axiome F10 :

$OL \vdash \underline{at}\ a \rightsquigarrow \underline{jafter}\ a$.

6 - On utilise le point 3 de ce théorème, en décomposant n fois puis on simplifie à l'aide des points 1, 2, 4.

Fin du lemme 2.3

III.4.- QUELQUES REMARQUES AU SUJET DE OL

Les ordinaux dénombrables se sont révélés nécessaires dans la preuve de complétude sémantique ; ces ordinaux, en fait, constituent des variables auxiliaires de démonstration. Notre preuve de complétude sémantique considère tous les ordinaux dénombrables a priori ; ce résultat peut paraître plus faible que celui de APT et PLOTKIN [APP] mais nous objecterons que la solution de APT et PLOTKIN [APP] ne répond pas au problème que nous nous sommes posés. En effet APT et PLOTKIN [APP] étudient le non déterminisme récursif et à cette fin ne considèrent que des objets récursifs : prédicats, fonctions et domaines.

Notre problème est celui plus général du non déterminisme dénombrable. Certe une amélioration de ce résultat pourrait consister à utiliser des objects récursifs et une étude ultérieure sera menée dans ce sens.

Le problème de la minimalité déductive des règles de OL est intéressant explorer. Les axiomes F1,...,F10 constituent l'image axiomatique de l'équité faible et sont en nombre minimal par le choix même de la syntaxe. Nous envisageons maintenant quelques sous-systèmes de OL et regardons s'ils sont complets malgré tout.

Notons OL_1, OL_2, OL_3, OL_4 les sous-systèmes de OL sans la règle 1, la règle 2, la règle 3, la règle 4.

① Cas de OL_1

Sous hypothèse d'équité faible on peut envisager la propriété

at x:=t $\rightsquigarrow$ after x:=t.

La preuve de cette propriété exige l'utilisation de la règle 4 :

$OL \vdash$ at x:=t $\rightsquigarrow$ jafter x:=t et $OL \vdash$ jafter x:=t $\Rightarrow$ after x:=t

ou l'utilisation de la règle 2 ou 3.

Dans chacun des cas on a besoin de prouver une propriété liée à l'implication :

Règle 2 :

i ≡ jafter a $\Rightarrow$ after a où a = x:=t est invariante pour CIP.

Donc $_{OL} \vdash$ at a $\rightsquigarrow$ jafter a$\wedge$after a

et jafter a$\wedge$after a $\Rightarrow$ after a, et on est bloqué.

Règle 4 :

$OL \vdash$ at x:=t $\rightsquigarrow$ jafter x:=t, $_{OL} \vdash$ jafter x:=t $\Rightarrow$ after x:=t

On est encore bloqué.

Règle 3 :

$OL \vdash$ at a $\rightsquigarrow \exists\alpha R(\alpha)$

$OL \vdash R(\alpha) \rightsquigarrow \exists\beta < \alpha, R(\beta)$

$OL \vdash R(0) \rightsquigarrow$ after a

On se bloque aussi parce qu'on a besoin de l'implication pour établir chaque prémisse.

Résultat 1 : OL_1 est sémantiquement incomplet.

② Cas OL_2

La règle 2 est essentielle pour déduire et enrichir les assertions de fatalité. Soit i une assertion invariante pour CIP. Supposons que $P \rightsquigarrow Q$ soit vraie pour CIP.

Alors sémantiquement $(P\wedge i) \rightsquigarrow (Q\wedge i)$ est vrai aussi.

On ne peut le déduire à partir de $P \rightsquigarrow Q$ et de i.

Si (par exemple), i s'écrit (jafter a $\Rightarrow$ R) et P ≡ at a, Q ≡ jafter a.

On ne peut déduire (at a$\wedge$i) $\rightsquigarrow$ jafter a$\wedge$R à partir de OL_2.

La seule règle applicable est la règle 3 et dans ce cas on ne peut observer la transition a qu'au travers de l'axiome de fatalité et, nécessairement, on utilisera une règle du type 2 pour prouver cela.

Résultat 2 : OL_2 est sémantiquement incomplet.

③ Cas OL_3

Le système d'OWICKI et LAMPORT ne contenait pas de règle d'induction. Or pour prouver la terminaison d'un programme, une induction est nécessaire.

④ Cas OL_4

La relation $\rightsquigarrow$ est transitive et afin de prouver la transitivité de $\rightsquigarrow$ il faut utiliser la règle 4.

Afin de prouver la transitivité de $\rightsquigarrow$, on utilise une suite $(S(\alpha))_{\alpha\in Ord}$ telle que :

$OL \vdash P \rightsquigarrow \exists\alpha S(\alpha)$,

$\forall\alpha > 0, \alpha \in Ord, \ OL \vdash S(\alpha) \rightsquigarrow \exists\beta < \alpha, S(\beta)$

$OL \vdash S(0) \rightsquigarrow Q$.

La règle 4 est nécessaire pour déduire :

$OL \vdash S(\alpha) \rightsquigarrow \exists\beta < \alpha S(\beta)$, pour $\alpha \geqslant 2$.

Car on a besoin de $OL \vdash S(1) \Rightarrow \exists\beta < \alpha, S(\beta)$.

L'autre façon est d'utiliser la règle 2 et, dans ce cas-là, aucun invariant n'est manifeste :

On suppose $OL \vdash S(2) \rightsquigarrow S(1)$ et $OL \vdash S(1) \Rightarrow \exists i < 2, S(i)$.

On en déduit que :

$S(2)(s) \wedge t^*[C\mathbb{P}](s,s') \wedge S(1)(s') \Rightarrow \exists i < 2, S(i)(s')$ est vrai pour tout couple (s,s') de $S[C\mathbb{P}]^2$.

D'où par la règle 2 :

$OL \vdash S(2) \rightsquigarrow S(1) \wedge \exists i < 2, S(i)$,

et on ne peut plus poursuivre car il manque la règle 4.

Résultat 4 : OL_4 est sémantiquement incomplet.

D'un point de vue sémantique, OL est donc minimal.

III.5.- REMARQUES A PROPOS D'AUTRES TRAVAUX

L'étude de la correction et de la complétude sémantique du système OL conduit à la mise en évidence de certains ordinaux. Cette étude des propriétés de fatalité distingue deux philosophies temporelles : une philosophie linéaire et une philosophie arborescente. Notre système de preuve mélange ces deux aspects du temps d'exécution : chaque axiome d'équité faible linéarise l'arborescence due au choix du processus à activer à tout instant. Ces deux philosophies du temps conduisent à deux logiques temporelles : la logique temporelle linéaire et la logique temporelle arborescente. BEN-ARI, MANNA et PNUELI [BMP] étudient cette logique temporelle arborescente et l'approche d'OWICKI et LAMPORT [OWL] se veut être une logique linéaire temporelle mais leur argumentation reste philosophique et informelle, VAN BENTHEM [VAN] offre un panorama important de ces logiques du temps notamment pour ce qui concerne la densité du temps, la discrétion ... mais ceci est une étude de l'expressivité et non de la déductibilité des phénomènes temporels. De plus, les règles de déduction ne sont pas adaptées aux programmes, MANNA et PNUELI [MAP1], [MAP2], [MAP3] tentent d'axiomatiser les notions de fatalité, d'invariance, d'équité, de justice, mais le cadre temporel est trop informellement utilisé. Nous pensons que le but des auteurs ci-dessus, est de définir un cadre unique pour déduire, exprimer et prouver des propriétés de programmes. Cette démarche nous paraît encore trop aventureuse car a priori on n'a pas précisé toutes les propriétés des programmes.

Cette volonté d'étendre un formalisme conduit à des systèmes sortant du cadre primitif de ce formalisme : par exemple, le système d'OWICKI et GRIES

[OWG] étend la logique de HOARE [HOA] aux programmes parallèles mais nous semble être un pseudo-système formel à cause de la règle du Cobegin qui exige l'absence d'interférence, notion nullement axiomatisée. LAMPORT [LAM 3] étudie le lien entre le type de temps considéré et le type du programme étudié et en déduit que la logique temporelle linéaire est adéquate pour l'étude des programmes parallèles, alors que la logique temporelle arborescente est adéquate pour l'étude des programmes non déderministes ; cette argumentation est revue et corrigée par EMERSON et HALPERN [EMA 2].

IV - MÉTHODE DES TREILLIS DE PREUVE

IV - MÉTHODE DES TREILLIS DE PREUVE

Le système de preuve OL étend celui d'OWICKI et LAMPORT [OWL] et comme celui d'OWICKI et LAMPORT on peut lui associer une méthode pratique basée sur OL et permettant de représenter les preuves sous forme de diagrammes appelés treillis de preuve. Ce point nous paraît intéressant à développer formellement car dans les exemples pratiques les preuves de fatalité sont complexes.

IV.1.- METHODE DES TREILLIS DE PREUVE

Nous exposons la méthode et donnons un résultat de complétude et de correction relatif à cette méthode. Mais avant tout il convient de préciser la notion de treillis de preuve.

Définition [IV.1].1 :

Soit $C\mathbb{P}$ un programme parallèle,

$\mathcal{A}[C\mathbb{P}]$ un langage d'assertions pour $C\mathbb{P}$,

$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable de $C\mathbb{P}$,

P,Q deux assertions de $\mathcal{A}[C\mathbb{P}]$.

On appelle treillis de preuve pour $C\mathbb{P}$ de noeud entrant P et de noeud sortant Q, et on note $TP(P,Q)$, tout diagramme fini acyclique dont les noeuds sont étiquetés par des assertions de $\mathcal{A}[C\mathbb{P}]$ et dont les flèches sont telles que :

1.- Pour tout noeud d'étiquette R de $TP(P,Q)$, distinct de Q, la flèche $Q \rightarrow R$ n'existe pas.

2.- Pour tout noeud d'étiquette R de $TP(P,Q)$, distinct de P, la flèche $R \rightarrow P$ n'existe pas.

3.- Si les flèches issues de R dans TP(P,Q) sont $R \to R_k$ pour $k \in \{1,\ldots,\ell\}$, alors $\mathcal{M}_{WF} \models R \rightsquigarrow [\bigvee_{k\in[\ell]} R_k]$.

Une preuve dans OL peut être schématisée par un diagramme tel que chaque flèche puisse être prouvée à l'aide de OL. Cette remarque constitue la description informelle de la méthode des treillis de preuve.

Méthode des treillis de preuve

Soit CP un programme parallèle,

$\mathcal{A}[CP]$ un langage d'assertions pour CP,

P,Q deux assertions de $\mathcal{A}[CP]$,

$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable de CP.

Pour prouver $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, on construit un treillis de preuve TP(P,Q) tel que :

si $R \to R_k$, $k \in \{1,\ldots,\ell\}$ sont les flèches de R à R_k,

alors $_{OL}\vdash R \rightsquigarrow (R_1 \vee \ldots \vee R_\ell)$.

Cette méthode nécessite le système OL mais elle constitue un support graphique intéressant pour utiliser OL. Nous donnerons plus tard une illustration de cette méthode, qui sera en même temps une illustration de preuve avec OL. En effet nous prouverons que l'algorithme corrigé de BEN-ARI [BEN2] du ramasse-miettesest tel que toute miette est ramassée fatalement au bout d'un temps fini. Cet exemple expose la difficulté des preuves et surtout montre les problèmes liés à une automatisation de telles techniques.

Cependant avant d'étudier la correction et la complétude de cette méthode

nous l'illustrons par un exemple simple que nous avons exploité au chapitre précédent. Nous donnons le programme CP_1 et la propriété qui l'accompagne sous forme du treillis de preuve qui suit.

Rappelons CP_1 :

$CP_1 \equiv$ Cobegin [p:=false] [] while p do [Skip] od coend

Propriété : $_{OL}\vdash (at\ CP_1 \wedge p) \rightsquigarrow (after\ CP_1 \wedge \sim p)$

On se reportera à la figure représentant le treillis de preuve associé à la preuve. Nous justifions chaque lien existant dans le treillis de preuve, en se repérant par des numéros cerclés et la justification est dans la preuve formelle faite précédemment :

(1) D'après (1)

(2) D'après (7) et (9), par transitivité de

(3) D'après (16)

(4) D'après (19), en utilisant la règle 1A,

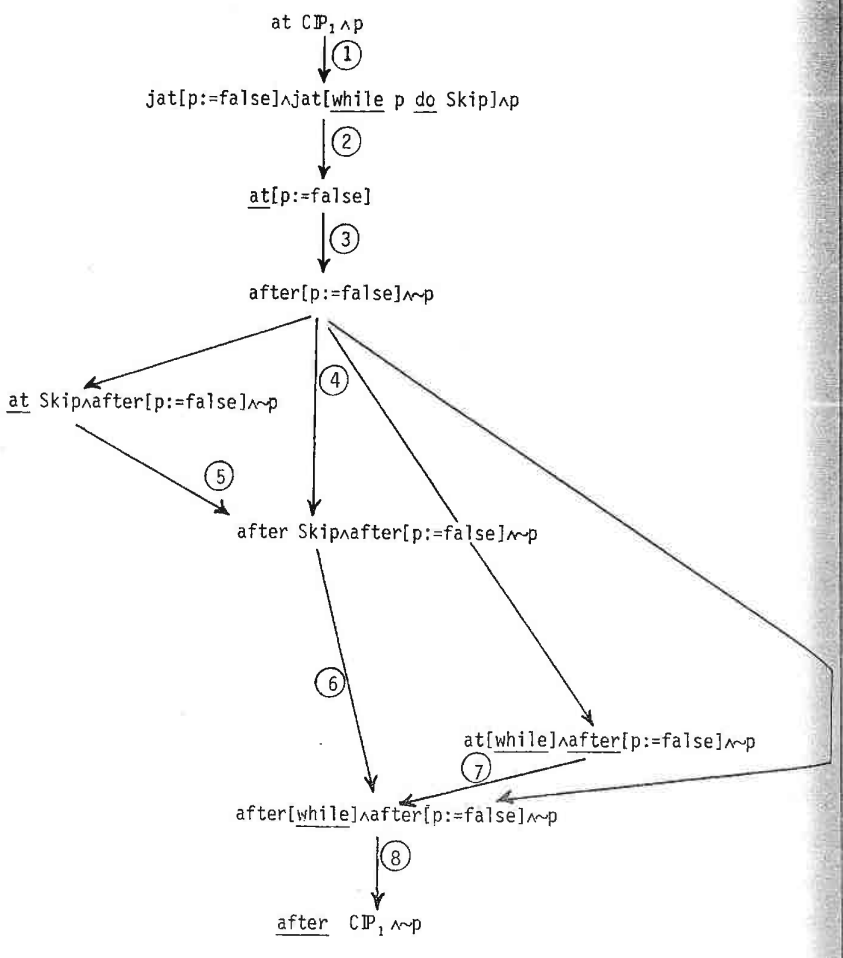
(5) D'après (22)

(6) D'après (23)

(7) D'après (20)

(8) D'après (27).

Une remarque est à faire au sujet des deux preuves : le treillis de preuve dépend de la preuve formelle mais il permet d'ébaucher une preuve formelle. De plus, le treillis de preuve permet de renseigner l'informaticien sur l'état d'avancement de sa preuve et lui permet aussi d'avoir une vue globale du problème.



Treillis de preuve pour CIP_1

TP(at CIP_1 ∧p, after CIP_1 ∧~p)

La méthode des treillis de preuve se justifie et ceci est l'objet du théorème suivant. Nous pourrons par la suite con sidérer le cas plus conséquent du ramasse-miette de BEN-ARI [BEN2].

Théorème [IV.1].1 :

La méthode des treillis de preuve est correcte.

Preuve

Soit CIP un programme parallèle quelconque,

$\mathcal{A}$[CIP] un langage d'assertions pour CIP

P,Q deux assertions de $\mathcal{A}$[CIP],

$\mathcal{M}_{WF}$ le modèle de fatalité faiblement équitable de CIP.

On suppose qu'il existe un treillis de preuve TP(P,Q) construit à l'aide de la méthode ci-dessus. TP(P,Q) désignera indistinctement le treillis de preuve et l'ensemble des assertions. On définit sur TP(P,Q) une notion de longueur d'un noeud quelconque de TP(P,Q),R, à Q par $\ell(R,Q)$: $\ell(R,Q)$ est définie pour tout noeud de TP(P,Q) par :

- $\ell(Q,Q) = 0$
- $\forall R \in TP(P,Q)-\{Q\}$, $\ell(R,Q) = Max(\ell(R',Q)/(R' \in TP(P,Q)) \wedge (R \rightarrow R')) + 1$.

En procédant à une induction sur la longueur du noeud R au noeud Q, on montre que, pour tout noeud R de TP(P,Q), $\mathcal{M}_{WF} \models R \rightsquigarrow Q$. On appliquera cette propriété à P. La preuve s'inspire de celle d'OWICKI et LAMPORT

① $\ell(R,Q) = 0$ et $R \in TP(P,Q)$

Seul Q vérifie cela et évidemment, $\mathcal{M}_{WF} \models Q \rightsquigarrow Q$.

② Supposons que $\ell(R,Q) = n$, $R \in TP(P,Q)$ et, pour tout entier k strictement plus petit n, pour toute assertion R' de $TP(P,Q)$ telle que $\ell(R',Q) = k$, $\mathcal{M}_{WF} \models R' \rightsquigarrow Q$.

Notons Im-Suc(R), le sous-ensemble de $TP(P,Q)$ défini comme suit :

Im-Suc(R) = $\{R' \in TP(P,Q) / R \rightarrow R'\}$. Puisque $TP(P,Q)$ est fini, on peut énumérer Im-Suc(R) :

Im-Suc(R) = $\{R_1,\ldots,R_\ell\}$ où $\ell = |\text{Im-Suc}(R)|$.

Soit $R_i \in \text{Im-Suc}(R)$ $(i \in \{1,\ldots,\ell\})$.

$\ell(R,Q) = \text{Max}(\ell(R_j,Q)/j \in \{1,\ldots,\ell\})+1 = n$.

On en déduit que : $\ell(R_i,Q) < n$, $\forall i \in \{1,\ldots,\ell\}$.

D'après l'hypothèse d'induction, pour tout i de $\{1,\ldots,\ell\}$, $\mathcal{M}_{WF} \models R_i \rightsquigarrow Q$. Par complétude de OL, on déduit que

(1) $\quad \forall i \in \{1,\ldots,\ell\}, {}_{OL}\vdash R_i \rightsquigarrow Q$.

D'après le point 3 de la définition des treillis de preuve, $\mathcal{M}_{WF} \models R \rightsquigarrow \bigvee_{i\in\{1,\ldots,\ell\}} R_i$. Par complétude de OL, on en déduit alors :

(2) $\quad {}_{OL}\vdash R \rightsquigarrow \bigvee_{i\in\{1,\ldots,\ell\}} R_i$.

On applique alors la règle 5 prouvée dans la preuve de complétude pour $I = \{1,\ldots,\ell\}$. On en déduit que ${}_{OL}\vdash R \rightsquigarrow Q$ et par correction $\mathcal{M}_{WF} \models R \rightsquigarrow Q$.

$\forall R \in TP(P,Q)$, $\mathcal{M}_{WF} \models R \rightsquigarrow Q$.

En particulier, si $R \equiv P$, $\mathcal{M}_{WF} \models P \rightsquigarrow Q$. □

Nous examinons maintenant la complétude de cette méthode qui, du reste, est triviale.

Théorème [IV.1].2 :

La méthode des treillis de preuve est complète.

Preuve :

Soit $C\mathbb{P}$ un programme parallèle,

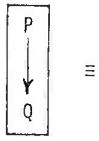
$\mathcal{A}[C\mathbb{P}]$ un langage d'assertions pour $C\mathbb{P}$,

$\mathcal{M}_{WF}$ un modèle de fatalité faiblement équitable,

P,Q deux assertions de $\mathcal{A}[C\mathbb{P}]$ telles que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$.

Par complétude de OL, on déduit ${}_{OL}\vdash P \rightsquigarrow Q$.

Soit alors le treillis de preuve suivant :



$\equiv TP_0(P,Q)$

$TP_0(P,Q)$ est un treillis de preuve de noeud entrant P et de noeud sortant Q construit d'après la méthode. □

La complétude de la méthode des treillis de preuve est triviale car nous utilisons grossièrement mais correctement le système axiomatique OL correct et sémantiquement complet. Cette méthode va être maintenant illustrée par le ramasse-miettes erroné de BEN-ARI [BEN2].

IV.2.- APPLICATION DE LA METHODE DES TREILLIS DE PREUVE

La méthode des treillis de preuve a été précédemment illustrée à l'aide de l'algorithme CIP_1 dont nous avons montré la correction totale. CIP_1 est un programme parallèle trivial et nous donnons maintenant une preuve de fatalité concernant le deuxième algorithme du ramasse-miettes proposé par BEN-ARI [BEN2]. La preuve nous a permis d'y déceler une erreur : nous voulions prouver que toute miette est fatalement ramassée un jour. L'algorithme a été modifié, afin de le rendre correct pour cette propriété de fatalité. Dans un premier temps, nous donnons une description du problème du ramassage de miettes et l'algorithme du ramasse-miettes en parallèle. Puis quelques définitions préliminaires sont données pour être utilisées par la suite dans la preuve, que nous décrivons en terme de treillis de preuve commentés.

IV.2.1.- L'algorithme du ramasse-miettes de BEN-ARI

Le problème du ramassage des miettes se pose, lorsqu'on considère des systèmes comme un interpréteur LISP, dans lequel des manipulations de pointeurs entraînent que certains noeuds du système deviennent inaccessibles à partir d'une racine donnée. Cette situation conduit à une classe de noeuds du système appelés miettes ; de tels noeuds doivent être identifiés et ramassés, afin qu'ils puissent être utilisés ultérieurement par le système à sa discrétion. Dans un tel système on distingue deux types de noeuds à un instant donné quelconque :

- les noeuds accessibles à partir d'une racine
- les noeuds inaccessibles à partir d'une racine ou miettes.

Les solutions envisagées pour ce type de problème sont de deux types :

- une solution séquentielle, c'est-à-dire que l'activité du système est momentanément stoppée et le ramasse-miettes est activé. Cette opération est faite chaque fois que le système parvient à un état critique au niveau des noeuds disponibles.
- une solution parallèle, c'est-à-dire que l'activité du système se fait en parallèle avec celle du ramasse-miettes. Cette solution améliore notoirement la précédente mais pose le problème de l'identification des miettes.

Notre étude se classe dans le choix de la solution parallèle et cette solution est source d'erreurs fréquentes. DIJKSTRA [DLMS] en citent notamment une non triviale à détecter. Nous décrivons tout d'abord la structure de données, sur laquelle s'exécutera l'algorithme proposé :

M est un tableau de noeuds dont la dimension est Number-of-Nodes ; parmi ces noeuds, certains sont appelés racines et leur indice associé est élément de {1,...,Number-of-Roots}, on distingue une racine particulière que l'on dénote FREE (FREE $\in$ {1,...,Number-of-Roots}). Le nombre de racines est Number-of-Roots. Un noeud est un objet structuré dont ces champs sont :

- un tableau qui désigne ses fils ; ce tableau est de dimension Number-of-Sons.
- une variable contenant une des trois couleurs possibles : blanc, gris ou noir, qui sera utilisée pour identifier les miettes.

Formellement, on décrit la structure de données par le diagramme suivant

```
DATA STRUCTURE
TYPE HUE is (WHITE, GRAY, BLACK);
TYPE INDEX is new INTEGER range 1..Number-of-Nodes;
SUBTYPE ROOTS is INDEX range 1..Number-of-Roots;
TYPE SONS is new INTEGER range 1..Number-of-Sons;
TYPE NODE is record
        SON:ARRAY(SONS) of INDEX;
        COLOR:HUE:=BLACK;
        end-record;
M:ARRAY (INDEX) of NODE;
```

Nous poursuivons par la description informelle de l'algorithme proprement dit. L'algorithme du ramasse-miettes en parallèle sera désigné par le symbole IM‖C ; IM‖C est composé de deux programmes séquentiels IM et C partageant la structure de données ci-dessus.

IM est le mutateur, qui simule l'activité du système : il détruit un lien d'un noeud accessible vers un autre noeud accessible ou établit un lien d'un noeud accessible vers un autre noeud accessible : il crée les miettes.

C est le collecteur, qui doit identifier les miettes (c'est-à-dire, les noeuds qui ne sont plus accessibles à partir d'une racine) et les ramasser, pour les mettre dans le réservoir des noeuds disponibles, que nous appelons liste libre.

La relation parallèle doit être telle que l'activité d'identification des

miettes par le collecteur n'est pas gênée par l'activité simultanée du mutateur. Une solution utilisée déjà par DIJKSTRA [DLMS] et reprise par BEN-ARI [BEN1][BEN2] consiste à utiliser des couleurs pour identifier les miettes : on ramasse les noeuds blancs. Mais il faut introduire 3 couleurs : gris, blanc et noir. A l'initialisation tous les noeuds sont noirs et un noeud qui devient gris est suspecté d'être miette ; une opération de propagation du blanc permet de savoir, si c'est véritablement une miette. Nous décrivons plus formellement IM et C.

(1) IM est constitué de deux actions atomiques exécutées en séquence et à la discrétion de IM. L'activité de IM est cyclique.

```
A0 : M(M(R).SON(S)).COLOR:=GRAY;
A1 : M(R).SON(S):=T;
```

IM s'assure que R et T sont accessibles à partir d'une racine.

(2) C est constitué de 3 phases exécutées en séquence et à la discrétion de C. L'activité de C est cyclique.

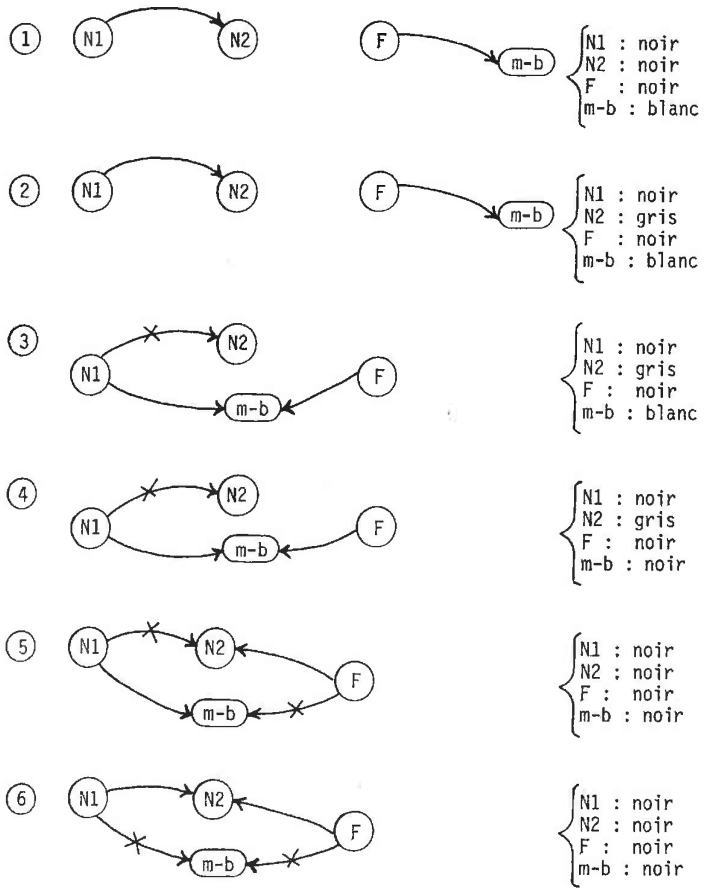
```
S1 ;
S2 ;
S3 ;
```

→ S1 blanchit les noeuds gris, qui sont suspectés être des miettes (inaccessibles à partir d'une racine) et propage la couleur blanc à tous les des-

cendants de ces noeuds gris blanchis.

→ S2 rétablit la situation des noeuds blanchis abusivement et qui sont en fait encore accessibles à partir d'une racine. S2 propage le noir et à la fin de S2, tous les noeuds blancs sont des miettes.

→ S3 est la phase de ramassage des noeuds miettes blancs et on les adjoint à la liste libre, après les avoir noircis BEN-ARI [BEN2] faisait d'abord l'adjonction du noeud blanc miette et noircissait seulement ensuite ce noeud. Cette inversion d'opérations conduit à une erreur : en effet, si on fait d'abord l'adjonction de la miette blanche à cette liste libre, notée m-b, et si ℂ se bloque alors, m-b est accessible à partir de la racine FREE et peut être utilisée par 𝕄 ; 𝕄 crée un lien avec m-b, puis détruit ce lien en grisant m-b d'abord ; ℂ se réveille et noircit m-b, qui est noir alors ; ℂ ramasse la miette précédemment faite à la suite de la création du lien avec m-b et 𝕄 crée un lien avec cette miette recyclée : m-b devient alors noire et inaccessible. Aucune partie de ℂ ne peut la blanchir, elle devient inaccessible pour toute exécution de 𝕄ℂ, à partir de cet instant.



```
S1 ≡ C0 : For I in INDEX loop
     C1 :     if M(I).COLOR=GRAY  and I≠M(R).SON(S)
     C2 :       then M(I).COLOR:=WHITE;
     C3 :     end-if
     C4 : end-loop;
     C5 : Propagate-White;

S2 ≡ {Black-count:Integer:=1;Old-Black-count:Integer:=0}

C6 ≡ B0 : Begin
     B1 : For I in ROOTS loop
     B2 :     M(I).COLOR:=BLACK;
     B3 : end-loop;
     B4 : While Black-count >Old-Black-count loop
     B5 :     Old-Black-Count:=Black-count;
     B6 :     For I in INDEX loop
     B7 :         if M(I).COLOR>=GRAY
     B8 :           then For J in SONS loop
     B9 :                    if I≠R or J≠S
                             then M(M(I).SON(J)).COLOR:=BLACK;
                             end-if
     B10 :                   end-loop
     B11 :        end-if
     B12 :    end-for;
     B13 :    Black-count:=0;
     B14 :    For I in INDEX loop
     B15 :        if M(I).COLOR>=GRAY
     B16 :           then Black-count:=Black-count+1;
     B17 :        end-if
     B18 :    end-loop
     B19 : end-loop

S3 ≡ C7 : For I in INDEX loop
     C8 :     if M(I).COLOR=WHITE
     C9 :        then M(I).COLOR:=BLACK;
     C10 :            APPEND-TO-FREE(I);
     C11 :    end-if
     C12 : end-loop.
```


Propagate-White et APPEND-TO-FREE(I) sont des parties laissées non précisées et ceci est volontaire, afin de montrer l'utilisation de la règle 3A de OL.

Nous terminons cette partie et nous nous intéressons à la preuve de la propriété de fatalité suivante : toute miette est ramassée fatalement un jour.

IV.2.2.- Préliminaires à la preuve

Nous aurons besoin au cours de la preuve de certaines assertions portant sur l'état de $\mathbb{MC}$. On notera s un état quelconque de $\mathbb{MC}$.

(1) Pour chaque élément k de {1,...,Number-of-Nodes},

(G(k))(s) ≡ (M(k).COLOR=GRAY)(s)

(B(k))(s) ≡ (M(k).COLOR=BLACK)(s)

(W(k))(s) ≡ (M(k).COLOR=WHITE)(s)

Ces trois assertions permettent d'exprimer l'état du champ COLOR de chaque noeud : gris, noir ou blanc.

(2) $\mathbb{MC}$ est étiqueté par les lettres indexées suivantes :

A_i, B_j, C_ℓ, D_m où $i \in \{0,1\}$, $j \in \{0,\ldots,20\}$, $\ell \in \{0,\ldots,12\}$, $m \in \{0,\ldots,8\}$.

On écrit (at e)(s) pour signifier que, dans l'état s, le contrôle est à l'instruction étiquetée e.

(3) A chaque partie S1, S2, S3, nous associons des assertions de contrôle dont le sens est évident :

(at S1)(s) ≡ (at C0)(s)

(at S2)(s) ≡ (at C6)(s)

(at S3)(s) ≡ (at C7)(s)

$(\underline{in}\ S_1^1)(s) \equiv \bigvee_{i\in\{0,\ldots,4\}} (\underline{at}\ Ci)(s)$

$(\underline{in}\ S_1^2)(s) \equiv \underline{at}\ C5$

$(\underline{in}\ S2)(s) \equiv \bigvee_{j\in\{0,\ldots,20\}} (\underline{at}\ Bj)(s)$

$(\underline{in}\ C5)(s) \equiv \bigvee_{i\in\{0,\ldots,8\}} (\underline{at}\ Di)(s)$

$(\underline{in}\ S3)(s) \equiv \bigvee_{i\in\{7,\ldots,12\}} (\underline{at}\ Ci)(s)$

$(\underline{after}\ S1)(s) \equiv (\underline{at}\ S2)(s)$

$(\underline{after}\ S2)(s) \equiv (\underline{at}\ S3)(s)$

(4) Soient k et k' deux noeuds ie $k,k' \in \{1,\ldots,Number\text{-}of\text{-}Nodes\}$.
On dit que k est accessible à partir de k' dans l'état s, si l'assertion ACC(k,k') est vraie en s :

$$ACC(k,k')(s) \equiv [\exists p\geq 1,(i_1,\ldots,i_p)\subset\{1,\ldots,Number\text{-}of\text{-}Nodes\},$$
$$[(i_1=k)\wedge(i_p=k')\wedge(\forall j\in\{2,\ldots,p\},\exists \ell\in Sons(i_{j-1}),$$
$$[i_j=M(i_{j-1}).Son(\ell)])]](s).$$

On note $Sons(i_{j-1})$ les fils de i_{j-1}.
Intuitivement, ACC(k,k') signifie que k' est accessible à partir de k ie il y a un chemin de k à k'.

(5) Nous définissons ici l'assertion P(k) qui signifie que k est une miette ou non : toute racine ne peut être une miette et tout noeud inaccessible à partir d'une racine ou accessible à partir d'une miette est une miette, à condition de n'être accessible à partir d'aucune racine.
Soit $k \in \{1,\ldots,Number\text{-}of\text{-}Nodes\}$.

$$P(k)(s) = [(k\notin ROOTS)(s)\wedge[(\forall R\in ROOTS,\sim ACC(R,k))(s)]]$$

(6) Nous donnons une définition de l'assertion qui précise qu'un noeud est dans la liste des noeuds disponibles.

$$FREE(k)(s) \equiv [\forall R\in ROOTS-\{FREE\},\sim ACC(R,k)(s)]\wedge$$
$$ACC(FREE,k)(s)\wedge[B(k)(s)]\wedge$$
$$[\forall \ell\in\{2,\ldots,Number\text{-}of\text{-}Sons\},(M(k).Son(\ell)=NIL)(s)]]$$

(7) Au cours de la preuve de fatalité, nous aurons besoin de quelques assertions supplémentaires, notamment pour les inductions.

$$(G\text{-}APRES)(s) \equiv ([(\underline{in}\ S_1^1)\wedge(I>k)]\vee[\underline{in}\ S_1^2]$$
$$\vee[(I=k)\wedge\underline{in}\ S_1^1\wedge\sim\underline{at}\ C1\wedge\sim\underline{at}\ C2])(s)$$

Intuitivement, cela signifie que, si k est gris et miette, il ne sera identifié qu'au prochain cycle après.

$$(G\text{-}MAINT)(s) \equiv ([(\underline{in}\ S_1^1)\wedge(I<k)]$$
$$\vee[(I=k)\wedge(\underline{at}\ C1\vee\underline{at}\ C2)])(s)$$

Intuitivement, cela signifie que, si k est gris et miette, il sera identifié dans ce cycle maintenant.

$$(I\text{-}BLANC(m))(s) \equiv \bigvee_{k-I\leq m} [P(k)\wedge G(k)\wedge([\underline{in}\ S_1^1\wedge(I\neq k)]\vee[(I=k)\wedge\underline{at}\ C1]\vee[(I=k)\wedge\underline{at}\ C2])$$

IV.2.3.- Preuve de fatalité à propos de IMC

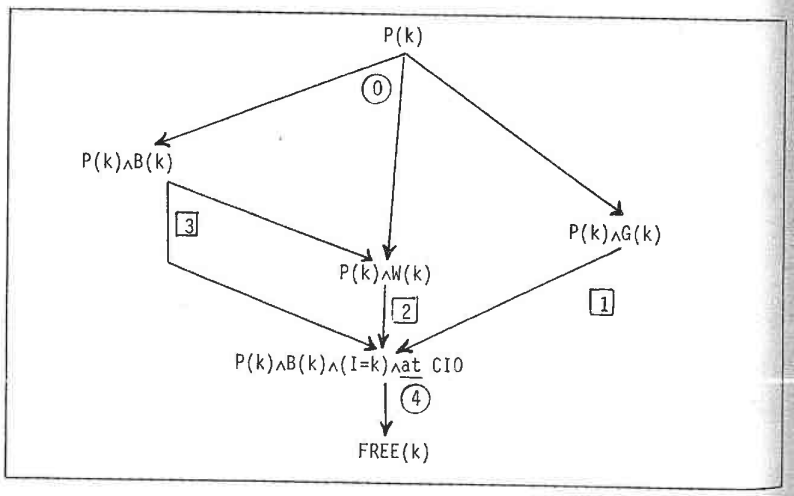
Théorème [IV.2.3].1 :

Soit k un élément de {1,...,Number-of-Nodes},

$0_L \vdash P(k) \rightsquigarrow FREE(k)$

La preuve consiste en la construction progressive et descendante du treillis de preuve avec P(k) comme noeud entrant et FREE(k) comme noeud sortant. Nous accompagnerons nos diagrammes de commentaires permettant de justifier chaque diagramme. Chaque partie cerclée est justifiée directement et chaque partie carrée est justifiée ultérieurement par un treillis de preuve.

Etape 0



Treillis de preuve n° 0

Commentaires sur le treillis de preuve n° 0

(0)

$P(k) \Rightarrow [(P(k)\wedge G(k))\vee(P(k)\wedge W(k))\vee(P(k)\wedge B(k))]$ est vrai pour cet algorithme car les seules couleurs sont gris, blanc ou noir : $G(k)\vee B(k)\vee W(k)$ est invariant pour cet algorithme.

$0_L \vdash P(k) \Rightarrow [(P(k)\wedge G(k))\vee(P(k)\wedge W(k))\vee(P(k)\wedge B(k))]$ (par la règle 1A)

$0_L \vdash P(k) \rightsquigarrow [(P(k)\wedge G(k))\vee(P(k)\wedge W(k))\vee(P(k)\wedge B(k))]$ (par la règle 1)

[1] , [2] , [3]

On se reportera aux treillis de preuve n° 1, 2, 3.

(4)

On suppose que k est une miette noire et que le contrôle est en C10 et que I contient k.

Nous utilisons la règle 2 et, à cette fin, nous exhibons une assertion $I^{(1)}$ permettant de prouver que $P(k)\wedge B(k)\wedge(I=k)$ est invariant, tant que le contrôle est en C10 :

$$I^{(1)} \equiv \{[(P(k)\wedge B(k)\wedge \underline{at}\ C10) \Rightarrow 0((P(k)\wedge B(k)\wedge \underline{at}\ C10)\vee(FREE(k)\wedge \underline{jafter}\ C10))]$$
$$\wedge[(\underline{jafter}\ C10) \Rightarrow ([(\otimes(P(k)\wedge B(k)\wedge \underline{at}\ C10) \Rightarrow FREE(k)]\wedge$$
$$[\sim\otimes(P(k)\wedge B(k)\wedge \underline{at}\ C10) \Rightarrow false])]$$

où 0 et $\otimes$ sont des opérateurs temporels définis par :

Soit A une assertion quelconque de $\mathcal{A}[C\mathbb{P}]$.

$(0A)(s)$ ssi $[\forall s' \in S[C\mathbb{P}], t[C\mathbb{P}](s,s') \Rightarrow A(s')]$.

$(\otimes A)(s)$ ssi $[\forall s' \in S[C\mathbb{P}], t[C\mathbb{P}](s',s) \Rightarrow A(s')]$.

On vérifie les assertions suivantes :

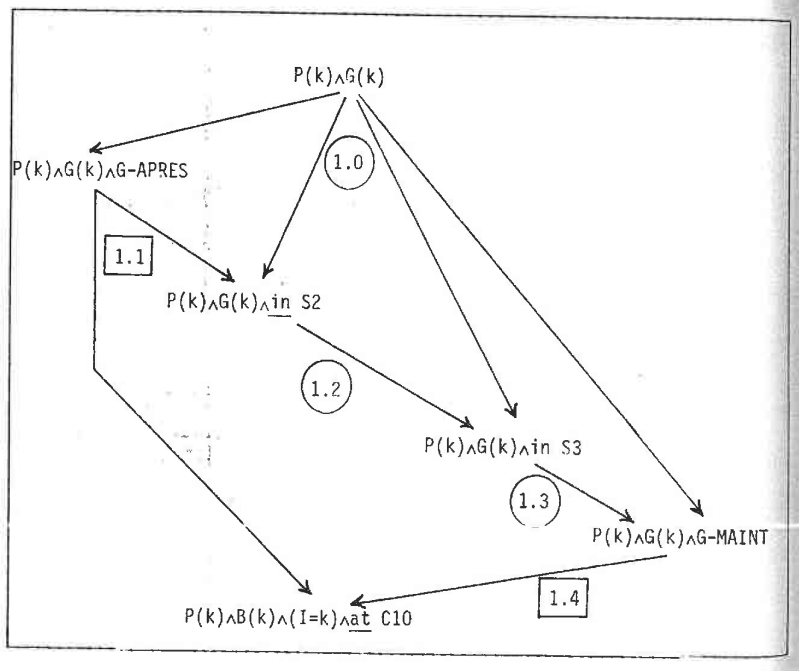
$$(1)\quad \begin{cases} I^{(1)} \Rightarrow OI^{(1)}, \\ \underline{at}\ C10 \wedge P(k) \wedge B(k) \wedge (I=k) \Rightarrow I^{(1)} \\ I^{(1)} \wedge \underline{jafter}\ C10 \Rightarrow FREE(k). \end{cases}$$

La règle 3A est utilisée pour déduire que la phase d'adjonction à la liste libre termine toujours :

$$OL \vdash \underline{at}\ C10 \rightsquigarrow \underline{jafter}\ C10.$$

Par application de la règle 2A à (1), puis de la règle 2 on en déduit la flèche du schéma.

Etape 1



Treillis de preuve n° 1

Commentaires sur le treillis de preuve n° 1

(1.0)

$$P(k)\wedge G(k) \Rightarrow [(P(k)\wedge G(k)\wedge \text{G-APRES}) \vee (P(k)\wedge G(k)\wedge \underline{in}\ S2) \vee (P(k)\wedge G(k)\wedge \underline{in}\ S3) \vee (P(k)\wedge G(k)\wedge \text{G-MAINT})]$$

est vraie, puisque G-APRES$\vee$in S2$\vee$in S3$\vee$G-MAINT est invariante pour cet algorithme : Le contrôle se trouve toujours en un point de contrôle du collecteur.

$$OL \vdash (P(k)\wedge G(k)) \rightsquigarrow [(P(k)\wedge G(k)\wedge \text{G-APRES}) \vee (P(k)\wedge G(k)\wedge \underline{in}\ S2) \vee (P(k)\wedge G(k)\wedge \underline{in}\ S3) \vee (P(k)\wedge G(k)\wedge \text{G-MAINT})]$$

(par la règle 1A, puis la règle 1).

[1.1]

On se reportera au treillis n° 1.1.

(1.2)

Remarquons que $P(k)$ est faux, si k est une racine ; $P(k)\wedge G(k)$ est invariant pour la phase S2 dite de marquage. Nous utilisons ici encore les opérateurs temporels O et $\otimes$ pour exprimer une assertion $I^{(2)}$.

$$I^{(2)} \equiv \Big((P(k)\wedge G(k)\wedge \underline{in}\ S2) \Rightarrow O[(P(k)\wedge G(k)\wedge \underline{in}\ S2) \vee (P(k)\wedge G(k)\wedge \underline{jafter}\ S2)]\Big)$$

$$\wedge\Big((\underline{jafter}\ S2) \Rightarrow \big[(\otimes (P(k)\wedge G(k)\wedge \underline{in}\ S2) \Rightarrow (P(k)\wedge G(k))) \wedge (\sim \otimes (P(k)\wedge G(k)\wedge \underline{in}\ S2) \Rightarrow \underline{false})\big]\Big)$$

On vérifie que $I^{(2)}$ est tel que :

$$(2)\quad \begin{cases} I^{(2)} \Rightarrow 0I^{(2)} \\ (\underline{in}\ S2 \wedge P(k) \wedge G(k)) \Rightarrow I^{(2)} \\ (I^{(2)} \wedge \underline{jafter}\ S2) \Rightarrow ((P(k) \wedge G(k)) \vee false) \end{cases}$$

On utilise la règle 2A pour (2), puis la règle 2, en utilisant le fait que S2 termine :

$_{0L} \vdash \underline{in}\ S2 \rightsquigarrow \underline{jafter}\ S2$, que l'on prouve par la règle 3.

(1.3)

Cette preuve est la même que celle ci-dessus : dans le cycle S3, $P(k) \wedge G($ est invariant ; pour le prouver, il suffit de considérer l'assertion $I^{(3)}$ obtenue à partir de $I^{(2)}$ en substituant S3 à S2. De même, on déduit la terminaison de S3 par la règle 3 : $_{0L} \vdash \underline{in}\ S3 \rightsquigarrow \underline{jafter}\ S3$.

Ceci nous permet de déduire :

(i) $_{0L} \vdash (\underline{in}\ S3 \wedge P(k) \wedge G(k)) \rightsquigarrow (\underline{jafter}\ S3 \wedge P(k) \wedge G(k))$

On déduit par définition de jafter que

(ii) $_{0L} \vdash (\underline{jafter}\ S3 \wedge P(k) \wedge G(k)) \Rightarrow (\underline{after}\ S3 \wedge P(k) \wedge G(k))$.

Par la règle 4 avec (i) et (ii), on déduit :

(iii) $_{0L} \vdash (\underline{in}\ S3 \wedge P(k) \wedge G(k)) \rightsquigarrow (\underline{after}\ S3 \wedge P(k) \wedge G(k))$

On peut déduire à partir des définitions des assertions de contrôle que :

$_{0L} \vdash \underline{after}\ S3 \rightsquigarrow \underline{jafter}\ S1$ ($\mathbb{C}$ est un programme cyclique).

De nouveau, $P(k) \wedge G(k)$ est invariant pour ce changement de point de contr
ce qui nous permet de déduire :

(iv) $_{0L} \vdash (\underline{after}\ S3 \wedge P(k) \wedge G(k) \rightsquigarrow (\underline{jat}\ S1 \wedge P(k) \wedge G(k))$.

Par le choix de jat S1, on se trouve au début de S1, on en déduit que l'assertion suivante est vraie :

(v) $(\underline{jat}\ S1 \wedge P(k) \wedge G(k)) \Rightarrow (P(k) \wedge G(k) \wedge G\text{-MAINT})$

: on se trouve au début de la boucle et $I < k$.

On notera l'utilité de l'assertion jat dans les preuves :

(vi) $_{0L} \vdash (\underline{after}\ S3 \wedge P(k) \wedge G(k)) \rightsquigarrow (P(k) \wedge G(k) \wedge G\text{-MAINT})$

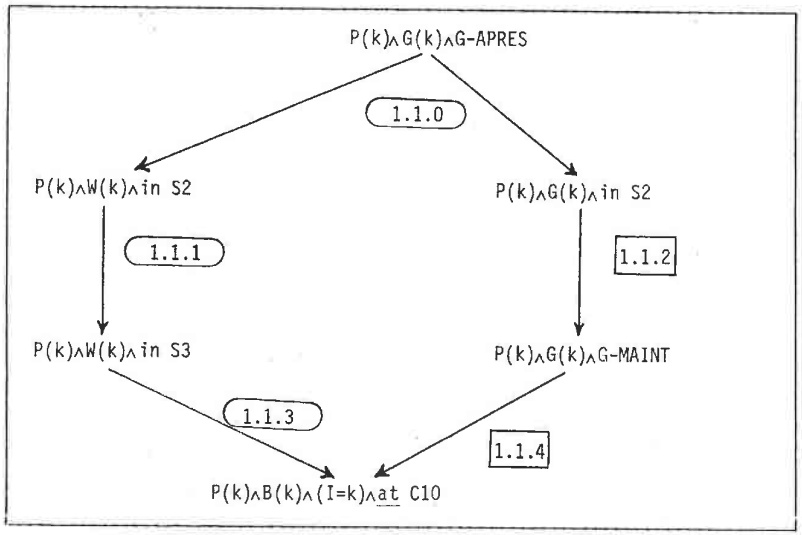
(par la règle 1A avec (v),

puis la règle 4 avec (iv) et (v) déduit).

$_{0L} \vdash (\underline{in}\ S3 \wedge P(k) \wedge G(k)) \rightsquigarrow (P(k) \wedge G(k) \wedge G\text{-MAINT})$

(par la règle 5 avec (iii) et (vi)).

Etape 1.1



Treillis de preuve n° 1.1

Commentaires sur le treillis de preuve n° 1.1

(1.1.0)

Pour cette partie nous utilisons la règle d'induction, en décomposant tout d'abord in S1 :

in S1 ≡ [at C0∨at C1∨at C2∨at C3∨at C4]∨[at C5]

≡ [in S_1^1]∨[in S_1^2].

Nous prouvons alors :

(1) $OL \vdash (P(k)\wedge G(k)\wedge \text{in } S_1^1 \wedge \text{G-APRES}) \rightsquigarrow (P(k)\wedge G(k)\wedge \text{in } S_1^2 \wedge \text{G-APRES})$

et

(2) $OL \vdash (P(k)\wedge G(k)\wedge \text{in } S_1^2 \wedge \text{G-APRES}) \rightsquigarrow \begin{bmatrix} (P(k)\wedge G(k)\wedge \text{in S2}) \\ \vee (P(k)\wedge W(k)\wedge \text{in S2}) \end{bmatrix}$

(a) Preuve de (1)

Pour cette preuve nous utilisons une induction et nous proposons la propriété suivante : $\forall n \in \{0,\ldots,N-k\}$,

$IG(n) \equiv P(k)\wedge G(k)\wedge \text{in } S_1^1 \wedge (N-I = n)\wedge[(k<I)\vee((k=I)\wedge\sim\text{at C1}\wedge\sim\text{at C2})]\wedge(k\leq I)$

où N désigne le nombre maximal de noeuds (ie : N = Number-of-Nodes).

$(P(k)\wedge G(k)\wedge \text{G-APRES in } S_1^1) \Rightarrow \bigvee_{n\leq n_0} IG(n)$ où $n_0 = N-k$.

On en déduit que :

$OL \vdash (P(k)\wedge G(k)\wedge \text{G-APRES}\wedge \text{in } S_1^1) \rightsquigarrow (\exists n \leq n_0,\ IG(n))$

(par les règles 1A,1).

Soit $n \in \{1,\ldots,n_0\}$.

in S_1^1 ≡ at C0∨at C1∨at C2∨at C3∨at C4 est une propriété invariante pour cet algorithme.

$IG(n) \Rightarrow [(\text{at C0}\wedge IG(n))\vee(\text{at C1}\wedge IG(n))\vee(\text{at C2}\wedge IG(n))\vee(\text{at C3}\wedge IG(n))\vee(\text{at C4}\wedge IG(n))$

est vraie pour ce programme.

$OL \vdash IG(n) \rightsquigarrow [(\text{at C0}\wedge IG(n))\vee(\text{at C1}\wedge IG(n))\vee(\text{at C2}\wedge IG(n))\vee$

$(\text{at C3}\wedge IG(n))\vee(\text{at C4}\wedge IG(n))]$.

at C0∧IG(n) ⇒ false car n est supposé compris entre 1 et n_0, ce qui signifie que $I \geq k$ et $I \leq N$.

$OL \vdash \text{false} \rightsquigarrow \exists n' < n$, IG(n') est déduit par le fait que false implique toute propriété et on poursuit par les règles 1A et 1.

at C1∧IG(n) ⇒ (at C1∧P(k)∧G(k)∧(k < I)∧(I = N-n)) par définition de IG(n) et choix de n.

$OL \vdash \text{at C1} \rightsquigarrow (\text{j-at C2}\wedge(M(I).COLOR=GRAY))\vee$

$(\text{j-after C3}\wedge(M(I).COLOR\neq GRAY))$

$\forall(s,s') \in S[\mathbb{M}C]$, (at C1∧P(k)∧G(k)∧(k < I))(s)∧

$t^*[AL](s,s')\wedge \text{j-after}\langle M(J).COLOR:=GRAY\rangle(s')$

⇒ (P(k)∧G(k)∧(k < I))(s').

P(k)∧G(k)∧(k < I) est invariant entre C1 et C2 ou C3 : cette assertion n'est pas modifiée tant que le test n'est pas évalué et exécuté.

On applique alors la règle 2 :

$OL \vdash (\text{at C1}\wedge P(k)\wedge G(k)\wedge(k < I)) \rightsquigarrow [(\text{j-at C2}\wedge(M(I).COLOR=GRAY)\wedge$

$P(k)\wedge G(k)\wedge(k < I))\vee$

$(\text{j-after C3}\wedge(M(I).COLOR\neq GRAY))\wedge$

$P(k)\wedge G(k)\wedge(k < I))]$.

De plus :

$[\text{j-at C2}\wedge(M(I).COLOR=GRAY)\wedge P(k)\wedge G(k)\wedge(k < I) \Rightarrow [\text{at C2}\wedge P(k)\wedge G(k)\wedge(k < I)]$

est vrai pour cet algorithme.

$OL \vdash [j\text{-}\underline{at}\ C2 \wedge P(k) \wedge G(k) \wedge (k<I) \wedge (M(I).COLOR=GRAY)] \rightsquigarrow$

$\rightsquigarrow [\underline{at}\ C2 \wedge P(k) \wedge G(k) \wedge (k < I)]$(Par les règles 1A, 1).

On déduit de même :

$OL \vdash [j\text{-}\underline{after}\ C3 \wedge P(k) \wedge G(k) \wedge (k < I) \wedge (M(I).COLOR \neq GRAY)] \rightsquigarrow$

$\rightsquigarrow [\underline{after}\ C3 \wedge P(k) \wedge G(k) \wedge (k < I)]$.

On utilise la règle 6 :

$OL \vdash [\underline{at}\ C1 \wedge P(k) \wedge G(k) \wedge (k < I)] \rightsquigarrow [(\underline{at}\ C2 \wedge P(k) \wedge G(k) \wedge (k < I)) \vee$

$(\underline{after}\ C3 \wedge P(k) \wedge G(k) \wedge (k < I))]$.

On procède comme ci-dessus en utilisant l'axiome :

$OL \vdash \underline{at}[M(I).COLOR:=WHITE] \rightsquigarrow j\text{-}\underline{after}[M(I).COLOR:=WHITE]$

Ce qui est équivalent à :

$OL \vdash \underline{at}\ C2 \rightsquigarrow j\text{-}\underline{after}\ C2$ et

$j\text{-}\underline{after}\ C2 \equiv j\text{-}\underline{at}\ C3$

Par le fait que $P(k) \wedge G(k) \wedge (k < I)$ est invariant entre C2 et C3, on en déduit :

$OL \vdash [\underline{at}\ C2 \wedge P(k) \wedge G(k) \wedge (k < I)] \rightsquigarrow [j\text{-}\underline{at}\ C3 \wedge P(k) \wedge G(k) \wedge (k < I)]$

Or

$j\text{-}\underline{at}\ C3 \Rightarrow \underline{at}\ C3$; on en déduit :

$OL \vdash [\underline{at}\ C2 \wedge P(k) \wedge G(k) \wedge (k < I)] \rightsquigarrow [\underline{at}\ C3 \wedge P(k) \wedge G(k) \wedge (k < I)]$.

On utilise maintenant l'axiome d'équité relatif à la sortie de la conditionnelle :

$OL \vdash \underline{at}\ C3 \rightsquigarrow j\text{-}\underline{after}\ C3$.

On recommence le même raisonnement :

$OL \vdash [\underline{at}\ C3 \wedge P(k) \wedge G(k) \wedge (I < k)] \rightsquigarrow [j\text{-}\underline{after}\ C3 \wedge P(k) \wedge G(k) \wedge (I > k)]$.

On rassemble les différents résultats pour déduire :

$OL \vdash [\underline{at}\ C1 \wedge P(k) \wedge G(k) \wedge (I < k)] \rightsquigarrow [\underline{after}\ C3 \wedge P(k) \wedge G(k) \wedge (I > k)$

$OL \vdash [\underline{at}\ C1 \wedge IG(n)] \rightsquigarrow [\underline{at}\ C4 \wedge P(k) \wedge G(k) \wedge (I > k)]$.

$N-I = n$.

On distinguera le FOR comme un While ; ainsi on a l'axiome suivant :

$OL \vdash \underline{at}\ C4 \rightsquigarrow [((I' \leqslant N) \wedge j\text{-}\underline{at}\ C1)$ où $I' = I+1$

$\vee (I' > N) \wedge j\text{-after FOR}]$

$OL \vdash (\underline{at}\ C4 \wedge P(k) \wedge G(k) \wedge (I > k)) \rightsquigarrow (\underline{at}\ C4 \wedge IG(n))$.

On a prouvé :

$OL \vdash \underline{at}\ C1 \wedge IG(n) \rightsquigarrow \underline{at}\ C4 \wedge IG(n)$

$n = N-I$.

Tant que le contrôle est en C4, IG(n) est vrai et le reste jursqu'à ce que la transition se fasse.

$OL \vdash \underline{at}\ C4 \wedge IG(n) \rightsquigarrow IG(n-1)$

On a prouvé que :

$OL \vdash \underline{at}\ C4 \wedge IG(n) \rightsquigarrow \exists n' < n,\ IG(n')$

$OL \vdash \underline{at}\ C1 \wedge IG(n) \rightsquigarrow \exists n' < n,\ IG(n')$.

$OL \vdash \underline{at}\ C2 \wedge IG(n) \rightsquigarrow \exists n' < n,\ IG(n')$.

$OL \vdash \underline{at}\ C3 \wedge IG(n) \rightsquigarrow \exists n' < n,\ IG(n')$.

D'où :

$OL \vdash IG(n) \rightsquigarrow \exists n' < n,\ IG(n')$.

$IG(0) \Rightarrow [P(k) \wedge G(k) \wedge in\ S_1^1 \wedge (I=N) \wedge (k \leqslant I) \wedge$

$I(k < N) \vee ((k=N) \wedge \sim \underline{at}\ C1 \wedge \sim \underline{at}\ C2)]]$.

On prouve de même, par cas, que :

$OL \vdash IG(0) \rightsquigarrow (\underline{after}\ S_1^1 \wedge P(k) \wedge G(k) \wedge G\text{-}APRES)$

On en déduit que :

$OL \vdash (P(k) \wedge G(k) \wedge G\text{-}APRES \wedge \underline{in}\ S_1^1) \rightsquigarrow (P(k) \wedge G(k) \wedge \underline{after}\ S_1^1 \wedge G\text{-}APRES)$

$OL \vdash (after\ S_1^1 \wedge P(k) \wedge G(k) \wedge G\text{-}APRES) \Rightarrow (in\ S_1^2 \wedge P(k) \wedge G(k) \wedge G\text{-}APRES)$

$OL \vdash (in\ S_1^1 \wedge P(k) \wedge G(k) \wedge G\text{-}APRES) \rightsquigarrow (in\ S_1^2 \wedge P(k) \wedge G(k) \wedge G\text{-}APRES)$

(par la règle 3, puis la règle 4).

(b) Preuve de (2)

$OL \vdash P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES) \rightsquigarrow [(P(k) \wedge G(k) \wedge \underline{in}\ S_2) \vee (P(k) \wedge W(k) \wedge \underline{in}\ S_2)$

En fait, un noeud miette gris peut être accessible à partir d'un noeud blanc miette et il faut donc distinguer ces cas par l'implication suivante :

$(P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES) \Rightarrow [(P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES \wedge (\exists k', (k \neq k') \wedge P(k') \wedge W(k') \wedge ACC(k',k))) \vee (P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES \wedge (\forall k', (P(k') \wedge W(k') \wedge (k \neq k') \Rightarrow \sim ACC(k',k)))]$

est vraie pour ce programme.

$OL \vdash (P(k) \wedge G(k) \wedge G\text{-}APRES \wedge in\ S_1^2) \rightsquigarrow [(P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES) \wedge (\exists k', (k \neq k') \wedge P(k') \wedge W(k') \wedge ACC(k',k)) \vee (P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES \wedge (\forall k', P(k') \wedge W(k') \wedge (k \neq k') \Rightarrow \sim ACC(k',k)))]$

(par les règles 1A, 1).

$in\ S_1^2 \equiv \underline{at}\ D0 \underline{vat}\ D1 \underline{vat}\ D2 \underline{vat}\ D3 \underline{vat}\ D4 \underline{vat}\ D5 \underline{vat}\ D6$

$OL \vdash in\ S_1^2 \rightsquigarrow j\text{-}\underline{after}\ S_1^2$ est déduit par la règle 3 et est fait en utilisant les diverses hypothèses d'équité comme pour la preuve (1) ci-dessus.

$P(k) \wedge G(k)$ est invariant pour S_1^2.

On en déduit :

$OL \vdash (P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES \wedge (\forall k', P(k') \wedge (k \neq k') \wedge W(k') \Rightarrow \sim ACC(k',k))) (P(k) \wedge G(k) \wedge j\text{-}\underline{after}\ S_1^2)$.

$OL \vdash (P(k) \wedge G(k) \wedge j\text{-}\underline{after}\ S_1^2) \rightsquigarrow (P(k) \wedge G(k) \wedge in\ S_2)$.

On prouve maintenant que :

$OL \vdash (P(k) \wedge G(k) \wedge in\ S_1^2 \wedge G\text{-}APRES \wedge (\exists k' (k \neq k') \wedge P(k') \wedge W(k') \wedge ACC(k',k))) \rightsquigarrow (P(k) \wedge W(k) \wedge \underline{in}\ S_2)$

Au cours de la phase de propagation du blanc (S_1^2), k devient fatalement blanc car k' est blanc au début de cette phase.

(1.1.1)

On prouve tout d'abord que cette phase S2 termine et ceci est fait à l'aide de l'induction de la règle 3.

Puis on utilise le fait que $P(k) \wedge W(k)$ est invariant pour la phase S2 : par la règle 2 :

$OL \vdash P(k) \wedge W(k) \wedge in\ S2 \rightsquigarrow P(k) \wedge W(k) \wedge \underline{jafter}\ S2$.

Ceci permet d'obtenir (1.1.1), en ajoutant que

$OL \vdash (P(k) \wedge W(k) \wedge j\text{-}\underline{after}\ S2) \Rightarrow (P(k) \wedge W(k) \wedge \underline{after}\ S2)$

D'où :

$OL \vdash (P(k) \wedge W(k) \wedge in\ S2) \rightsquigarrow (P(k) \wedge W(k) \wedge \underline{after}\ S2)$

Or :

$\vdash_{OL} P(k) \wedge W(k) \wedge \underline{after}\ S2 \Rightarrow P(k) \wedge W(k) \wedge \underline{in}\ S3$:

on utilise la règle 4, pour déduire (1.1.1).

[1.1.2] : On considère le treillis de preuve n° 1 et on extrait ce treillis [1.1.2], à l'aide de (1.2) et (1.3) .

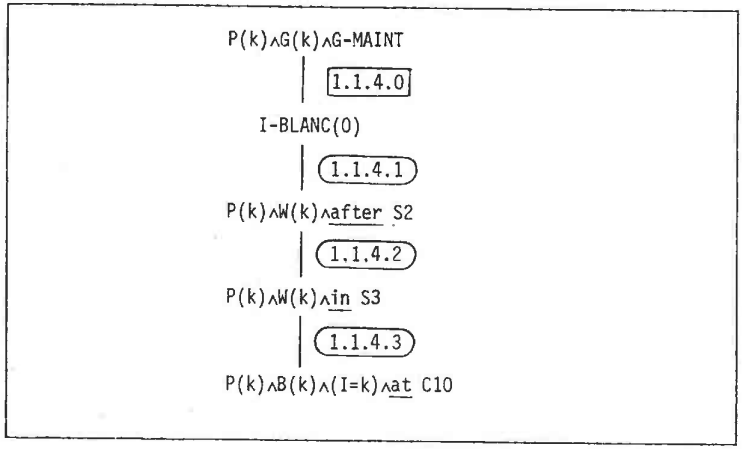
(1.1.3)

Si k est une miette blanche lorsque le contrôle est en S3, k était blanche au début de S3 : On utilise à nouveau le fait que $P(k) \wedge W(k)$ est invariant pour l'exécution du programme de 0 à k et on utilise une induction qui permet de montrer que k devient noir et miette en C10.

[1.1.4]

On verra le treillis de preuve n° 1.1.4.

Etape 1.1.4



Treillis de preuve n° 1.1.4

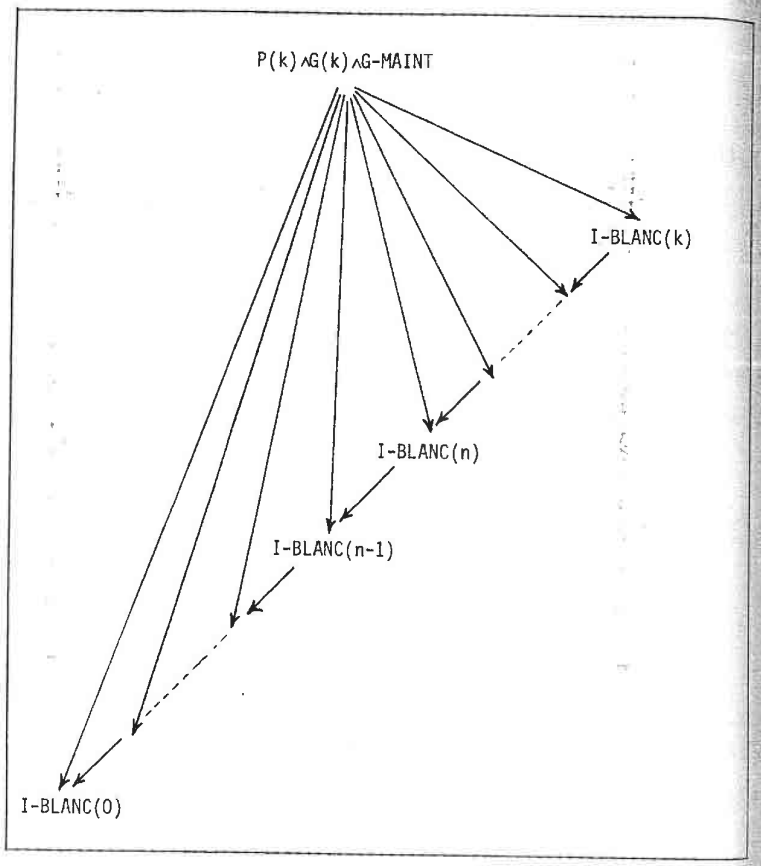
[1.1.4.0] : on verra le treillis de numéro 1.1.4.0.

(1.1.4.1) : Par la définition de I-BLANC(0), on en déduit que la boucle et S2 se terminent, en ayant mis k à blanc.

(1.1.4.2) : after S2 ⇒ in S3 ; on en déduit avec la règle 1A ce point.

(1.1.4.3) : On verra le treillis 1.1 et le lien (1.1.3).

Etape 1.1.4.0



Commentaires sur le treillis de preuve n° 1.1.4.0

Ce treillis représente l'induction sur n, à l'aide de l'assertion I-BLANC(n). Il suffit de prouver :

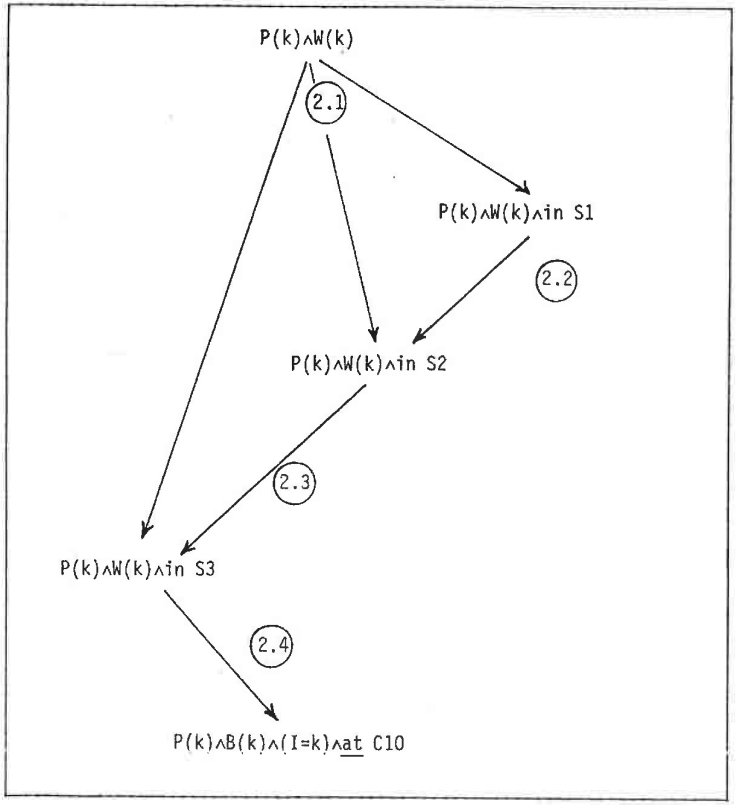
$OL \vdash (P(k) \wedge G(k) \wedge G\text{-MAINT}) \rightsquigarrow \exists k_1 \leq k,\ I\text{-BLANC}(k_1)$

$OL \vdash I\text{-BLANC}(k_1) \rightsquigarrow \exists k_2 < k_1,\ I\text{-BLANC}(k_2).\ k_1 > 0$

Etape 1.4

Ce treillis de preuve (1.4) est justifié par le treillis de preuve (1.1.4) déjà justifié et qui lui est équivalent.

Etape 2



Treillis de preuve n° 2

Commentaires sur le treillis de preuve n° 2

(2.1)

L'assertion suivante $\underline{in}\ S1 \vee \underline{in}\ S2 \vee \underline{in}\ S3$ est invariante pour cet algorithme ; on en déduit que :

$$P(k)\wedge W(k) \Rightarrow [(P(k)\wedge W(k)\wedge \underline{in}\ S1)\vee (P(k)\wedge W(k)\wedge \underline{in}\ S2)\vee (P(k)\wedge W(k)\wedge \underline{in}\ S3)].$$

$$_{OL}\vdash P(k)\wedge W(k) \rightsquigarrow [(P(k)\wedge W(k)\wedge \underline{in}\ S1)\vee (P(k)\wedge W(k)\wedge \underline{in}\ S3)].$$

(par la règle 1A, puis la règle 1).

(2.2)

Il suffit ici d'utiliser le fait que $P(k)\wedge W(k)$ est invariant pendant la phase S1. En effet, au cours de cette phase, on blanchit ce qui est alors gris et on propage le blanc.

Trivialement, $_{OL}\vdash \underline{in}\ S1 \rightsquigarrow \underline{j\text{-}after}\ S1$ est prouvé à l'aide de deux inductions successives. On utilise enfin, un invariant i" obtenu par substitution de S1 à S2.

D'où $\quad _{OL}\vdash in\ S1\wedge P(k)\wedge W(k) \rightsquigarrow \underline{j\text{-}after}\ S1\wedge P(k)\wedge W(k).$

Or $\quad _{OL}\vdash \underline{j\text{-}after}\ S1\wedge P(k)\wedge W(k) \Rightarrow \underline{after}\ S1\wedge P(k)\wedge W(k).$

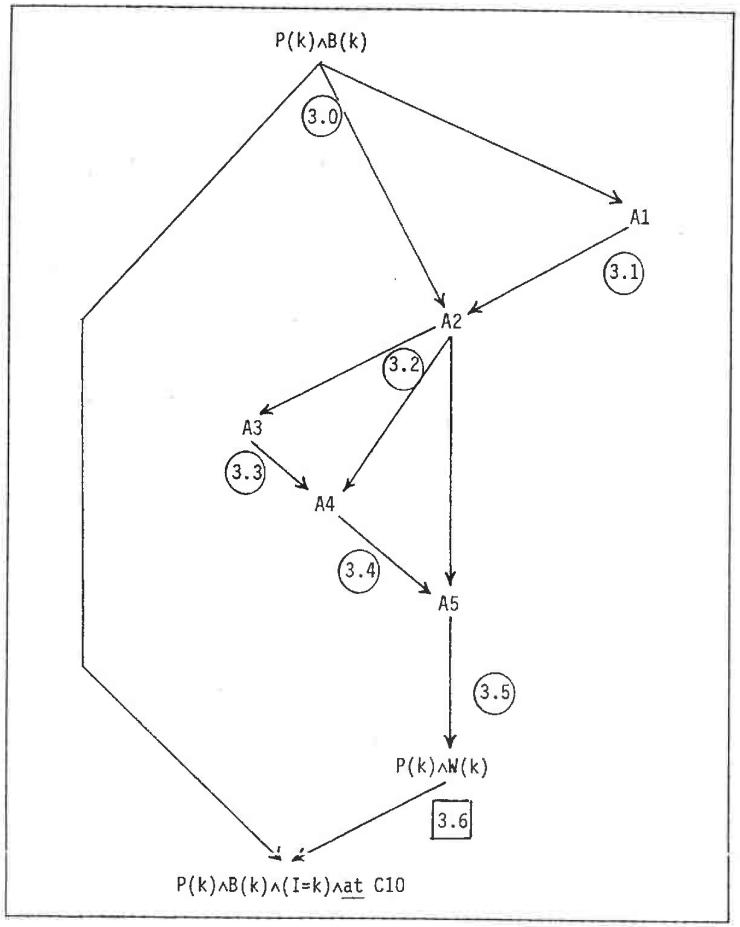
D'où, en utilisant la règle 4, on en déduit la partie 2.2.

(2.3)

Même commentaire que (1.1.1) dans [1.1] .

(2.4) Même commentaire que (1.1.3) dans [1.1] .

Etape 3



Treillis de preuve n° 3

Commentaires sur le treillis de preuve n° 3

$A1 \equiv P(k)\wedge B(k)\wedge(\exists k',(k\neq k')\wedge P(k')\wedge G(k')\wedge ACC(k',k))\wedge(\sim at\ C10\vee(I\neq k))$

$A2 \equiv P(k)\wedge B(k)\wedge(\exists k',(k\neq k')\wedge P(k')\wedge W(k')\wedge ACC(k',k))\wedge(\sim \underline{at}\ C10\vee(I\neq k))$

$A3 \equiv P(k)\wedge B(k)\wedge(\exists k',(k\ k')\wedge P(k')\wedge W(k')\wedge ACC(k',k))\wedge(\sim \underline{at}\ C10\vee(I\neq k))\wedge in\ S3$

$A4 \equiv A2\wedge in\ S2$

$A5 \equiv A2\wedge in\ S1$

(3.0)

Si on considère le fait d'être une miette noire, on constate que ce cas se présente si on se trouve juste avant l'ajout de cette miette à la liste libre, ou si on considère une miette accessible à partir d'une miette grise ou blanche.

Par la règle 1A :

$OL \vdash P(k)\wedge B(k) \Rightarrow (P(k)\wedge B(k)\wedge \underline{at}\ C10\wedge(I=k))$

$\vee(P(k)\wedge B(k)\wedge(\exists k',(k'\neq k)\wedge(P(k')\wedge W(k'))\wedge ACC(k',k)$

$\vee(P(k)\wedge B(k)\wedge(\exists k',(k'\neq k)\wedge(P(k')\wedge G(k'))\wedge ACC(k',k))$

On utilise la règle 1 pour déduire (3.0) .

(3.1)

On a prouvé que $OL \vdash P(k")\wedge G(k") \rightsquigarrow P(k")\wedge W(k")$;

$P(k)\wedge B(k)\wedge(\sim \underline{at}\ C10\vee(I\neq k))$ est invariant pendant la phase qui modifie k' :

$OL \vdash P(k')\wedge G(k') \rightsquigarrow P(k')\wedge W(k')$.

On en déduit par la règle 2 que :

$OL \vdash (P(k)\wedge B(k)\wedge \exists k',(k\neq k')\wedge P(k')\wedge G(k'))\wedge(\sim \underline{at}\ C10\vee(I\neq k))$

$\rightsquigarrow P(k)\wedge B(k)\wedge(\exists k',(k\neq k')\wedge P(k')\wedge W(k'))\wedge(\sim \underline{at}\ C10\vee(I\neq k))$

(3.2)

Il s'agit ici encore d'une analyse par cas et elle s'opère autour de la définition du contrôle :

$[P(k)\wedge B(k)\wedge(\exists k',(k\neq k')\wedge P(k')\wedge W(k'))\wedge(\sim \underline{at}\ C10\vee(I\neq k)] \Rightarrow$

$[(P(k)\wedge B(k)\wedge(\exists k',(k\neq k')\wedge P(k')\wedge P(k')\wedge W(k')\wedge(\sim \underline{at}\ C10\vee(I\neq k))\wedge(in\ S1 \vee in\ S2 \vee in\ S3)$

On en déduit (3.2) par les règles 1A, 1.

(3.3) , (3.4)

Ces deux cas impliquent l'assertion false : l'algorithme $\mathbb{MC}$ propage la blanc complètement et une miette accessible d'une miette blanche ne peut être noire, lorsque le contrôle est en S2 ou S3.

(3.5)

$P(k)\wedge B(k)\wedge in\ S1\wedge(\exists k',(k\neq k')\wedge P(k')\wedge W(k')\wedge ACC(k',k))\wedge$

$(\sim \underline{at}\ C10\vee(I\neq k)) \equiv A(k).$

Nous prouvons que k devient blanc au cours de la phase de propagation. Ceci est dû au fait que ou bien k' est blanchi au cours de la pré-phase S_1^1 ou k' est blanchi pendant la phase de propagation et tout élément accessible est alors blanchi.

$OL \vdash \underline{at}\ C5 \rightsquigarrow \underline{j\text{-}after}\ C5$ et

k devient blanc.

(3.6)

Il s'agit du treillis 2

IV.3.- IMPLANTATION DE L'HYPOTHESE D'EQUITE FAIBLE PAR DES ENSEMBLES BIEN ORDONNES

Nous voulons donner ici un cadre général pour définir une implantation au niveau de la sémantique opérationnelle afin que l'ensemble des traces d'exécution soit exactement l'ensemble des traces équitables. En se référant à l'article de PLOTKIN [PLO], celui-ci parle d'une sémantique positive ou génératrice ; contrairement à APT et OLDEROG [APO], nous transformons la sémantique et non la syntaxe ; notre étude a pour but de justifier certains types d'ordinaux dans les preuves comme les ordinaux polynomiaux simples ($<\omega^{\omega}$). Nous donnons d'abord quelques rappels préliminaires à notre étude.

IV.3.1.- Préliminaires

Définition [IV.3.1].1 :

Soit W un ensemble, $<$ une relation binaire bien fondée sur W. On dit que W est un ensemble bien ordonné par $<$, si $<$ est telle que :

1.- $<$ est antiréflexive (ie $\forall x \in W$, non $(x < x)$).

2.- $<$ est transitive, antisymétrique.

3.- $<$ est connexe : $\forall (x,y) \in W^2$, $(x < y) \vee (x > y) \vee (x = y)$.

Définition [IV.3.1].2 (et propriété) :

Soit $(W,<)$ un ensemble bien ordonné. Il existe un unique élément w_0 dans W, appelé minimum de W :

$\forall w \in W$, $w \neq w_0 \Rightarrow w_0 < w$.

Justification de la définition ci-dessus

Unicité :

Soient w_0 et w_0' tels que $\forall w \in W$, $w \neq w_0 \Rightarrow w_0 < w$

$w \neq w_0' \Rightarrow w_0' < w$.

Donc $w_0 < w_0'$ et $w_0' < w_0$. Par antisymétrie on en déduit que $w_0 = w_0'$.

Existence :

Supposons qu'il n'existe pas de tel w dans W.

Alors $\forall w_0 \in W$, $\exists w_0' \in W$, $(w_0 \neq w_0') \wedge \text{non}(w_0 \neq w_0')$

Puisque $w_0 < w_0' \vee w_0' < w_0 \vee w_0' = w_0$ et que $w_0 \neq w_0'$ alors $w_0' < w_0$.

On construit à partir de w_0 une suite décroissante infinie $w_0 > \ldots > w_i > \ldots$ ce qui est absurde car $<$ est bien-fondée.

Un objet comme $(W,<)$ est appelé un arbre par Chang et Keisler et w_0 est la racine de l'arbre.

Définition [IV.3.1].3 :

Soit $(W,<)$ un ensemble bien ordonné. On associe à tout élément w de W un ordinal noté $o(w)$ et $|W|$ est l'ordinal de $(W,<)$ défini ainsi :

(1) $o(w_0) = 0$

(2) $o(w) = \text{Sup}(o(w')+1/w' < w)$

(3) $o(W,<) = \text{Sup}(o(w)/w \in W)$.

Citons quelques exemples d'ensembles bien ordonnés.

(1) $(N,<)$ l'ensemble des naturels munis de l'ordre habituel est bien fondé et $o(N,<) = \omega$.

(2) $(N^2,<_{lex})$ l'ensemble produit de N et N muni de l'ordre lexicographique est bien ordonné et $o(N^2,<) = \omega^2$.

En fait, on peut associer réciproquement à tout ordinal un ensemble bien ordonné. Nous nous restreindrons aux ordinaux dénombrables. ω_1 est le premier ordinal non-dénombrable.

On définit sur $(W,<)$ deux opérations de façon inductive, en remarquant le fait suivant :

Soit Ord l'ensemble des ordinaux dénombrables.

Ord est construit par les 3 opérations suivantes :

1.- $o \in Ord$

2.- si $\alpha \in Ord$, $\alpha+1 \in Ord$

3.- si $(\alpha_i)_{i\in I}$ est une suite croissante dénombrable de Ord, $Sup(\alpha_i/i\in I) \in Ord$.

Suc et Pred sont deux opérations définies sur Ord et à valeurs dans Ord :

(1) $Suc(o) = 1$ et $Pred(o) = 0$.

(2) $Suc(\alpha+1) = Suc(\alpha)+1$.

(3) $Suc(Sup(\alpha_i/i\in I)) = Sup(Suc(\alpha_i/i\in I))$.

(4) $Pred(\alpha+1) = \alpha$.

(5) $Pred(Sup(\alpha_i/i\in I)) = \alpha_i$, $i \in I$.

(6) $Pred(\alpha) = \alpha$, $\forall \alpha \geqslant o(W,<)$.

IV.3.2.- Sémantique générative

Définition [IV.3.2].1 :

Soit CP un programme parallèle,

$\mathcal{A}[CP]$ un langage d'assertions pour CP,

$\mathcal{M}$ un modèle de fatalité de CP,

A[CP] l'ensemble des actions atomiques de CP.

(1) ind est une bijection de A[CP] dans $\{1,\ldots,|A[CP]|\}$ ie elle associe un numéro à chaque action atomique.

(2) enabled(a) est un prédicat défini pour chaque action atomique a de A[CP] par :

$s \in S[CP]$

enabled(a)(s) ssi (at a)(s)

(3) Soit i un élément de $\{1,\ldots,|A[CP]|\}$.

enabled'(i) ssi enabled($ind^{-t}(i)$)(s).

(4) Pour chaque état s de S[CP], on définit une fonction non déterministe, notée $?^S$ qui associe de façon non déterministe un élément de W à a, $a \in A[CP]$.

$?^S(a) \in W$.

Nous transformons la sémantique opérationnelle (S[CP], t[CP], T[CP]) en une autre sémantique ($S^f[CP]$, $t^f[CP]$, $T^f[CP]$).

f signifie équitable où $\mathcal{M}$ = (S[CP], t[CP], T[CP]).

Définition [IV.3.2].2 :

f est une application de S[CP] dans $S^f[CP]$ qui permet de définir $t^f[CP]$ sur $S^f[CP]$ et $T^f[CP]$:

(5) $f : S[CP] \to S^f[CP]$.

$s \in S[CP]$, $s = (\ell_1,\ldots,\ell_n,m)$

$f(s) = (w_1,\ldots,w_\ell,\ell_1,\ldots,\ell_n,m)$ où $\ell = |A[CP]|$ et

$\forall i \in \{1,\ldots,\ell\}$, $\exists! a \in A[CP]$, ind(a) = i.

(6) Soient s,s' deux états quelconques de S[CP].

$$t^f[CIP](f(s),f(s')) \equiv$$

$$\Big[\exists i\in[\ell],(enabled(i)(s))\wedge(w_i=\min(w_j/(j\neq i)\wedge enabled(j)(s)))$$

$$\wedge(ind^{-1}(i)\in A[PS_{i_0}])\wedge(\forall j\neq i_0,\ \ell_j=\ell'_j)\wedge$$

$$t[CIP](s,s')\wedge$$

$$(\forall j\in\{1,\ldots,\ell\},(enabled(j)(s)) \Rightarrow (w'_j\in Pred(w_j)))\wedge$$

$$(\forall j\in\{1,\ldots,\ell\}\sim enabled(j)(s) \Rightarrow (w'_j=w_j))\wedge$$

$$[(w'_i=Suc^k(w_i))\wedge(k>n)]\wedge\sim enabled(i)(s')\wedge$$

$$(\forall j\in\{1,\ldots,\ell\}\setminus(i),\ enabled(j)(s) \Rightarrow enabled(j)(s'))\Big].$$

Théorème [IV.3.2].

f est une application réalisant l'équité faible : ie

$T^f[CIP] = T^f_{WP}[CIP]$.

Preuve :

Soit a une action atomique quelconque de

s un état de CIP tel que at a est vraie en s.

(at a)(s) ssi (at a)(f(s)).

$f(s) = (w_1,\ldots,w_\ell,\ell_1,\ldots,\ell_n,m)$. On suppose que $ind(a) = 1$ et que ℓ_1 est l'étiquette juste devant a dans CIP sinon on permute.

Soit w un mot de $T^f[CIP]$ commençant par f(s).

Notons $w(i) = (w^i_1,\ldots,w^i_\ell,\ell^i_1,\ldots,\ell^i_n,m_i)$.

On associe à chaque transition $t^f[CIP](s_1,s_2)$ une action atomique $b(s_1,s_2)$ qui est celle qui est effectivement exécutée.

Pour tout $i < |w|$, $b(w(i),w(i+1)) = b(i)$.

Supposons qu'il existe un mot w' tel que $w \leq w'$ et w' infini avec $\forall i \in \mathbb{N}$, $b(i) \neq a$. On a prolongé la définition de b à w'.

Cela signifie que :

$\forall i \in \mathbb{N}$, $\exists b \in A[CIP]$, $(b(i) = b)\wedge(w^i_{b(i)} \leq w^i_a)$.

On déduit de la sémantique $t^f[CIP]$, que $w^i_a > w^{i+1}_a > \ldots$ est une chaine décroissante infinie.

Puisque $<$ est bien fondée, on déduit que $\exists i \in \mathbb{N}$, $w^i_a = w_0$.

D'où $\forall j \geq i$, $w^j_a = w^j_{b(j)} = w_0$. Puisque le nombre d'actions atomiques est fini, il existe $k \geq i$ tel que $\forall j \geq k$, $(w^j_{b(j)} > w_0)$. Ce qui est une contradiction.

D'où $\exists i \in \mathbb{N}$, $b(i) = a$. □

On en déduit que l'on peut modéliser les suites équitables d'un programme CIP par la fonction f.

Théorème [IV.3.2].2 :

Soit CIP un programme parallèle,

(W,<) un ensemble bien ordonné,

f une application d'équité faible.

Alors $(S^f[CIP], t^f[CIP], T^f[CIP])$ est la structure sémantique de fatalité faiblement équitable.

Preuve :

D'après le théorème ci-dessus. □

IV.3.3.- Implantation équitable

Définition [IV.4.3].1 :

On appelle implantation équitable pour CIP une application f sur un ensemble bien-ordonné (W,<). On note o(f) l'ordinal de (W,<), o(W,<).

Théorème [IV.3.3].1 :

Soit $C\mathbb{P}$ un programme parallèle,

$(S[C\mathbb{P}], t[C\mathbb{P}], T_{WP}[C\mathbb{P}])$ la structure sémantique de fatalité faiblement équitable.

Alors il existe une implémentation imp telle que :

(1) si $w \in T_{WF}[C\mathbb{P}]$, il existe $w' \in T^{imp}[C\mathbb{P}]$ tel que $w' = imp(w)$.

(2) si $w' \in T^{imp}[C\mathbb{P}]$, il existe $w \in T_{WF}[C\mathbb{P}]$ tel que $w' = imp(w)$.

(3) Pour tout P,Q de $\mathcal{L}$, $P \rightsquigarrow Q$ sous hypothèse d'équité faible ssi $(S^{imp},t^{imp}[C\mathbb{P}], T^{imp}[C\mathbb{P}]) \models P' \rightsquigarrow Q'$ si $P' \approxeq P$ et $Q' \approxeq Q$.

(4) $o(imp) = |(S[C\mathbb{P}], t[C\mathbb{P}], T_{WF}[C\mathbb{P}])|$.

Preuve :

Déduite des définitions précédentes et résultats précédents. □

L'implantation ci-dessus décrit précisément la situation mais en pratique tout est implanté sur des structures telles que $(N,<)$ et l'ordinal n'est que polynômial simple $(<\omega^{\omega})$. Cette remarque se traduit par le résultat suivant :

Théorème [IV.3.3].2 :

Soit $C\mathbb{P}$ un programme parallèle,

P,Q deux assertions de $\mathcal{L}$ pour $C\mathbb{P}$,

imp une implantation sur $(W,<)$.

Alors

si $(S^{imp}[C\mathbb{P}],t^{imp}[C\mathbb{P}], T^{imp}[C\mathbb{P}]) \models P \rightsquigarrow Q$,

$o(P,Q) \leqslant o(imp)$.

Ainsi, si on considère une implantation imp sur N on obtient $o(imp) = \omega^j$ pour un certain j et on déduit le résultat suivant :

Théorème [IV.3.3].3 :

Soit $C\mathbb{P}$ un programme parallèle,

imp une implantation sur W telle que $o(imp) < \omega^{\omega}$.

P,Q deux assertions sur $\mathcal{L}$.

Alors

$(S^{imp}[C\mathbb{P}], t^{imp}[C\mathbb{P}], T^{imp}[C\mathbb{P}]) \models P \rightsquigarrow Q$ ssi

$\exists n \in \mathbb{N}$, $\exists\{m_1,m_2,\ldots,m_n\} \subset \mathbb{N}$,

$$P \rightsquigarrow \exists m_1 P_1(m_1)$$

$$P_1(m) \rightsquigarrow P_1(0)$$

$$\vdots$$

$$P_n(0) \rightsquigarrow Q.$$

Preuve :

On la déduit du théorème de complètude et du théorème ci-dessus en posant :

$R_i(n) = \bigvee_{\alpha=\omega^i.n+B} R(\alpha)$ et $\alpha \leqslant o(P,Q)$.

On terminera par quelques remarques sur ces résultats.

En fait nous avons voulu montrer que les programmes pratiques n'étaient à envisager que dans des structures polynômiales et ceci a pour effet de simplifier et de donner des éléments constructifs de preuve dans OL. Aucun ordinal n'est apparu dans la preuve du ramasse-miettes ceci étant dû à un raisonnement polynômial.

IV.4.- ABSTRACTION DES TREILLIS DE PREUVE

Nous donnons maintenant une caractérisation formelle abstraite de la méthode utilisée ; cette caractérisation conduit à lier directement les ordinaux, les preuves dans OL et les treillis de preuve. Nous introduisons des notions abstraites dont nous donnons des interprétations.

Définition [IV.4].1 :

Soit $\mathcal{A}$ un langage d'assertions, $\mathcal{O}$ la classe des ordinaux.

On appelle diagramme ordinal tout objet de la forme $P \lhd\!\!\lhd_{\alpha} Q$ où P,Q sont des assertions de $\mathcal{A}$, α un membre de la classe $\mathcal{O}$.

On notera $\mathcal{O}[\mathcal{A}]$, la classe des diagrammes ordinaux sur $\mathcal{A}$. A cette notion abstraite, nous associons une interprétation au modèle ordinal.

Définition [IV.4].2 :

Soit $\mathcal{A}$ un langage d'assertions, $\mathcal{O}[\mathcal{A}]$ la classe des diagrammes ordinaux sur $\mathcal{A}$, E un ensemble non-vide, r une relation binaire sur E, T une partie de $T(E,r)$.

Une interprétation sur $\mathcal{O}[\mathcal{A}]$, notée J, est un objet tel qu'il existe un couple de fonctions ρ et $\tilde{\rho}$ telles que :

$$\mathcal{A} \underset{\rho}{\overset{\tilde{\rho}}{\rightleftarrows}} \mathcal{P}(E)$$

(1) $\forall P \in \mathcal{A}$, $\forall e \in E$, $P(e) \Leftrightarrow e \in \tilde{\rho}(P)$.

(2) $\forall P,Q \in \mathcal{A}$, $(J \models P \lhd\!\!\lhd_{\alpha} Q) \Leftrightarrow (*)$

$\alpha \in \mathcal{O}$

$$(*)\ \begin{cases} \forall e \in E, (P(e) \Rightarrow (\forall t \in T_e,\ \exists t' \in T_e,\ (t \leqslant t') \wedge (\exists i \in \mathrm{dom}(t'), Q(t'(i))))) \\ \text{et } \alpha \geqslant \|\Gamma\| \text{ où } \Gamma \text{ est l'opérateur associé à cette propriété.} \end{cases}$$

On remarquera de plus que $\mathrm{Card}(\alpha) \leqslant \mathrm{Card}(S)$. Cette définition généralise la notion de modèle de fatalité introduit auparavant.

Définition [IV.4].3 :

Soit $\mathcal{A}$ un langage d'assertions,

$\mathcal{O}[\mathcal{A}]$ la classe des diagrammes ordinaux sur $\mathcal{A}$.

Une interprétation J sur $\mathcal{O}[\mathcal{A}]$ est dénombrable, si la relation binaire r relative à J est dénombrable.

Etant données deux interprétations J et J' sur $\mathcal{O}[\mathcal{A}]$, on dit que J est plus faible que J', et on note $J \prec\!\!\!\cdot\ J'$, si $S \subseteq S'$, $T \subseteq T'$, $r \subseteq r'$ où S,S',T,T',r et r' sont relatifs à J et J'.

On peut citer un exemple utilisé souvent auparavant : le modèle de fatalité est une interprétation sur $\mathcal{O}[\mathcal{A}[C\mathbb{P}]]$. Comme précédemment nous déduisons quelques propriétés au sujet des ordinaux associés à des diagrammes ordinaux pour une certaine interprétation.

Théorème [IV.4].1 :

Soit $\mathcal{A}$ un langage d'assertions, J une interprétation sur $\mathcal{O}[\mathcal{A}]$.

1.- Si J est dénombrable, alors, pour tout ordinal α, toutes assertions P,Q de $\mathcal{A}$, si $\{J \models P \lhd\!\!\lhd_{\alpha} Q$, alors $\alpha < \omega_1\}$.

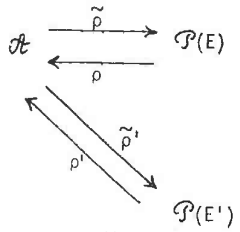
2.- Si, pour tout ordinal α, pour toutes assertions P,Q de $\mathcal{A}$, {si $J \models P \lhd\!\!\lhd_{\alpha} Q$, alors $\alpha < \omega_1$, alors il existe une interprétation J' sur $\mathcal{A}$ telle que :

- $J' \prec\!\!\prec J$.
- La classe des diagrammes ordinaux validés par J est la même que celle validée par J'.
- J' est dénombrable.

Preuve :

1.- Puisque J est dénombrable, on applique les résultats du chapitre 1 à (E,r,T) et on en déduit que est dénombrable.

2.- Supposons que J valide une classe de diagrammes ordinaux attachés à des ordinaux dénombrables. On définit une interprétation, notée J' en posant :



(E,r,T), (E',r',T').

(1) $E = E'$.

(2) Soit e,e' deux éléments quelconques de S,
soit Γ, l'ensemble de tous les opérateurs associés aux diagrammes ordinaux validés par J : d'après l'hypothèse, tout element Γ de Γ est tel que $\|\Gamma\| < \omega_1$.
On pose alors :
$r'(e,e')$ ssi $r(e,e') \wedge (\exists \Gamma \in \Gamma, \exists \alpha \leqslant \|\Gamma\|, (e \in \Gamma^{\alpha}) \wedge (e' \in \Gamma^{\beta}) \wedge (\beta < \alpha)$

(3) $T' = T \cap T(E,r)$.

On vérifie que J' réalise les trois conditions :

(1) $J' \prec\!\!\prec J$

(2) Soient $v(J), v(J')$ les classes respectives de validation de J et de J'. Puisque $T' \subset T$, $r' \subset r$, $S' \subset S$, $v(J') \subset v(J)$.
Soit $P \lhd\!\!\alpha\!\!\rhd Q$ tel que $J \models P \lhd\!\!\alpha\!\!\rhd Q$.
Soit Γ l'opérateur associé à $P \lhd\!\!\alpha\!\!\rhd Q$,
$\forall (e,e') \in (I_{\Gamma})^2$, $r(e,e')$ ssi $r'(e,e')$.
D'où $J' \models P \lhd\!\!\alpha\!\!\rhd Q$.

(3) Soit $suc(e) = \{e' \in S' / r'(e,e')\}$.
Soit $e \in \Gamma^{\alpha}$, pour $\Gamma \in \Gamma$ et $\alpha \leqslant \|\Gamma\|$.
$Card(suc(e)) \leqslant Card(\beta \in ord / \exists \Gamma \in \Gamma, (e \in \Gamma^{\alpha}) \wedge (e' \in \Gamma^{\beta}) \wedge (\beta < \alpha))$
$\leqslant \omega$.
D'où r' est dénombrable.

Définition [IV.4].3 :

Soit $\mathcal{A}$ un langage d'assertions,
J une interprétation sur $\mathcal{O}[\mathcal{A}]$,
On appelle ordinal de J, et on note $|J|$, tel que
$|J| = Sup\{\alpha / (P,Q \in \mathcal{A}) \wedge (J \models P \lhd\!\!\alpha\!\!\rhd Q)\}$.

Théorème [IV.4].2 :

Soit $\mathcal{A}$ un langage d'assertions.

1.- Si J est une interprétation sur $\mathcal{O}[\mathcal{A}]$ telle que

$\mathcal{A} \underset{\rho}{\overset{\tilde{\rho}}{\rightleftarrows}} E$, alors $|J| = Sup(\|\Gamma\| / \Gamma \in \Gamma_J)$ où Γ_J est l'ensemble des opérateurs associés à $\mathcal{O}[\mathcal{A}]$ interprété par J.

2.- Si J est dénombrable, alors $|J| \leqslant \omega_1$.

3.- Si J est une interprétation telle que $|J| < \omega_1$, alors il existe une interprétation dénombrable J' telle que

$J' \prec\!\!\!- J$, $|J'| = |J|$ et pour toutes assertions P,Q de $\mathcal{A}$,

$$J \models P \langle\alpha\rangle Q \text{ ssi } J' \models P \langle\alpha\rangle Q.$$

Preuve :

1.- Il s'agit tout simplement d'une autre formulation de la définition car à tout diagramme ordinal $P \langle\alpha\rangle Q$ valide pour J on associe un opérateur Γ tel que $P \Rightarrow \rho(\Gamma^{\alpha})$.

2.- Si J est une interprétation dénombrable, alors tout diagramme ordinal valide pour J est tel que son ordinal est dénombrable, donc $|J| \leqslant \omega_1$.

Supposons que $|J| = \omega_1$.

Dans ce cas, pour tout $\alpha < \omega_1$, il existe Q de $\mathcal{A}$ tel que $\|\Gamma_Q\| = \alpha$ où $\|\Gamma_Q\|$ est l'ordinal calculé à partir de Q, en arrière.

Soit Q_1, Q_2 telles que $Q_1, Q_2 \in \mathcal{A}$.

On remarque que $\Gamma_{Q_1} \cup \Gamma_{Q_2} \subset \Gamma_{Q_1 \vee Q_2}$

et $$\|\Gamma_{Q_1 \vee Q_2}\| \geqslant \|\Gamma_{Q_1}\|.$$
$$\geqslant \|\Gamma_{Q_2}\|.$$

Pour tout $\alpha < \omega_1$, il existe Q_α tel que $Q_\alpha \in \mathcal{A}$ et $\|\Gamma_{Q_\alpha}\| = \alpha$, soit $Q \equiv \bigvee_{\alpha \in Ord} Q_\alpha$. On doit examiner les différentes hypothèses au sujet de $\mathcal{A}$

(1) Q est expressible dans $\mathcal{A}$ ie $\mathcal{A}$ est de type $\mathcal{L}_{\alpha\beta}$ où $\alpha > \omega_1$.

(2) Q n'est pas expressible dans $\mathcal{A}$ ie $\mathcal{A}$ est de type.

Si Q n'est pas expressible dans $\mathcal{A}$, alors l'opérateur Γ_Q a pour

ordinal de clôture, par définition ω_1. Nous avons supposé J dénombrable, donc $|J| < \omega_1$.

3.- Soit J une interprétation telle que $|J| < \omega_1$.

J valide les diagrammes ordinaux attachés aux ordinaux dénombrables et, par le théorème IV.4.1, on en déduit qu'il existe J' dénombrable tel que $J' \prec\!\!\!- J$. Puisque J' valide les mêmes diagrammes ordinaux, alors $|J'| = |J|$.

On note $\mathcal{O}[\mathcal{A}]_J$, l'ensemble des diagrammes ordinaux valides pour J. On supposera J appartenant à un sur-ensemble noté I. On définit alors deux ensembles :

$$O_{\omega_1}[\mathcal{A}] = \{\mathcal{O}[\mathcal{A}]_J / J \in I(\omega_1)\}$$
$$I(\omega_1) = \{J \in I / |J| < \omega_1\}.$$

On en déduit la propriété suivante à propos de $I(\omega_1)$ et de $O_{\omega_1}[\mathcal{A}]$.

Théorème [IV.4].3 :

Soit $\mathcal{A}$ un langage d'assertions.

Il existe une surjection de $I(\omega_1)$ dans $O_{\omega_1}[\mathcal{A}]$.

Preuve :

Il suffit, pour cela, de définir la relation d'équivalence suivante notée $\approx$:

$J \approx J'$ ssi J et J' valident les mêmes diagrammes ordinaux.

Soit $I(\omega_1)/_{\approx}$, l'ensemble des classes d'équivalences de $I(\omega_1)$ modulo $\approx$.

Soit $s : I(\omega_1) \to O_{\omega_1}[\mathcal{A}]$

$J \to s(J)$ où $s(J) = \mathcal{O}[\mathcal{A}]_J$. □

Cette brève étude des diagrammes ordinaux et de leurs interprétations dénombrables, nous permet de donner une correspondance existant avec les treillis de preuve précédemment utilisés. L'interprétation J d'un diagramme ordinal sur un langage $\mathcal{A}$ est à comparer avec les modèles de fatalité. Les modèles de fatalité s'intéressent à des programmes et à des langages d'assertions $\mathcal{A}[\ ]$ associés à ces programmes.

Théorème [IV.4].4 :

Soit CP un programme parallèle,

$\mathcal{A}[CP]$ un langage d'assertions pour CP,

$\mathcal{M}_{WF}$ le modèle sémantique de fatalité faiblement équitable.

1.- $\mathcal{M}_{WF}$ est une interprétation sur $\mathcal{A}[CP]$.

2.- $\mathcal{O}[\mathcal{A}[CP]]_{\mathcal{M}_{WF}}$ est l'ensemble des assertions de fatalité valides pour $\mathcal{M}_{WF}$.

3.- Il existe trois applications s_1, s_2, i telles que

- s_1, s_2 soient des surjections.
- i est une bijection.
- $\mathcal{C}[CP][\mathcal{A}[CP]] \xrightarrow{s_1} \mathcal{O}[\mathcal{A}[CP]]_{\mathcal{M}_{WF}}$, $s_2 \downarrow \sim$, i : $\mathcal{C}[CP][\mathcal{A}[CP]] \to \mathcal{O}[\mathcal{A}[CP]]_{\mathcal{M}_{WF}}$

où $\mathcal{C}[CP][\mathcal{A}[CP]]$ est l'ensemble des treillis de preuve relatifs à CP construits avec la méthode et $\sim$ est une relation d'équivalence telle que :

$TP(P,Q) \sim TP(P',Q')$ ssi $P \equiv P'$ et $Q \equiv Q'$.

Preuve :

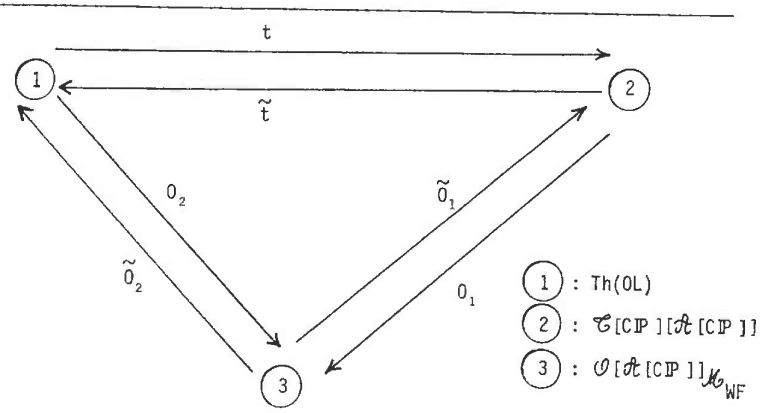
Elle se déduit immédiatement des résultats précédents.

On peut identifier les classes de treillis de preuves aux diagrammes ordinaux dont l'ordinal est plus petit que $|\mathcal{M}_{WF}|$. On écrira donc :

$P,Q \in \mathcal{A}[CP]$,

$TP(P,Q) \underset{\mathcal{M}_{WF}}{\equiv} P \lhd \alpha \rhd Q$ où $\alpha \leqslant |\mathcal{M}_{WF}|$.

Nous terminons ce paragraphe par une remarque globale entre la méthode des treillis de preuve, le système formel OL et les assertions de fatalité valides pour $\mathcal{M}_{WF}$.



Th(OL) désigne l'ensemble des théorèmes du système OL.

t permet de translater une preuve dans OL en une preuve par la méthode des treillis de preuve : c'est la méthode.

$\tilde{t}$ est obtenue par correction et complétude de OL à l'aide des autres fonctions.

$\tilde{O}_2$ est une abstraction au niveau de OL et qui existe par complétude de OL.

O_2 est une opération inverse de O qui existe par correction de OL.

$\tilde{O}_1$ est une abstraction au niveau de la méthode.

O_1 est une opération inverse de O_1. On écrira les propriétés suivantes : $P,Q \in \mathcal{L}[CIP]$.

$$t({}_{OL} \vdash P \rightsquigarrow Q) = \boxed{\begin{matrix} P \\ \downarrow \\ Q \end{matrix}} = TP(P,Q)$$

$$\tilde{t}(TP(P,Q)) = ({}_{OL} \vdash P \rightsquigarrow Q).$$

$$O_1(TP(P,Q)) = (\mathcal{M}_{WF} \models P \rightsquigarrow).$$

$$\tilde{O}_1(\mathcal{M}_{WF} \models P \rightsquigarrow Q) = TP(P,Q)$$

$$O_2({}_{OL} \vdash P \rightsquigarrow Q) = (\mathcal{M}_{WF} \models P \rightsquigarrow Q).$$

$$\tilde{O}_2(\mathcal{M}_{WF} \models P \rightsquigarrow Q) = ({}_{OL} \vdash P \rightsquigarrow Q).$$

On utilise dans la preuve de correction de la méthode des treillis de preuve souvent ce diagramme qui représente le raisonnement. On justifie ainsi la méthode et son énoncé. On peut récapituler les diverses règles de OL en un tableau qu'il faut comprendre de la façon suivante : tout ce qui est à gauche est à prendre pour hypothèse et à droite comme conclusion de la partie gauche.

$\Rightarrow$: $P \Rightarrow Q$	$P \lhd_0 Q$
$\square$: $P \wedge R \bullet\!\!-\!\!\boxed{S}\!\!-\!\!\bullet Q$; $P \lhd_\alpha Q$, $\beta \leq \alpha$	$P \wedge R \lhd_\beta Q \wedge S$
$\rightsquigarrow$: $P \lhd Q$	$P \lhd_\alpha Q$ $\alpha \in Ord$
$P \lhd_\alpha Q$, $Q \Rightarrow R$	$P \lhd_\alpha R$
$P \lhd_\alpha Q$, $R \lhd_\beta P$, $\alpha,\beta \in Ord$, $\alpha+\beta = \gamma$	$R \lhd_\gamma Q$
$\forall i \in I \subseteq \mathbb{N}$, $P_i \lhd_{\alpha_i} Q$ et $\alpha = \lim_{i \in I} \alpha_i$	$\bigvee_{i \in I} P_i \lhd_\alpha Q$
$P \lhd_\alpha Q$	${}_{OL} \vdash P \rightsquigarrow Q$
${}_{OL} \vdash P \rightsquigarrow Q$	$P \lhd_\alpha Q$

V - GÉNÉRALISATION AU CAS DES PROGRAMMES DISTRIBUÉS

V - GÉNÉRAL'SATION AU CAS DES PROGRAMMES DISTRIBUÉS

V.1.- PRESENTATION

L'étude des programmes a été, jusqu'ici, menée pour le cas de programmes parallèles formés de programmes séquentiels non-déterministes partageant des variables communes. Le mécanisme de "partage de variables communes" est un des mécanismes de communication rencontrés dans la littérature ; parmi les autres mécanismes de communication, on peut citer la notion de rendez-vous rencontrée dans CSP et qui a été étudiée d'un point de vue axiomatique par APT et CO [AFD], APT [APT2] et la notion d'envoi de messages dans un tampon de façon synchrone ou asynchrone. Ce dernier cas nous intéressera particulièrement dans ce chapitre et notre point de départ sera l'article de SCHLICHTING et SCHNEIDER [SCS] qui donnent notamment une méthode de preuve d'invariance de style HOARE [HOA] pour ce genre de programmes. Notre problème est de considérer le mécanisme d'envoi asynchrone de messages dans un tampon pour le cas des propriétés de fatalité et un exemple intéressant sera l'étude d'une de ces propriétés dans le cas de la version améliorée de l'algorithme d'exclusion mutuelle dans un réseau de N sites de RICART et AGRAWALA [RIA], due à CARVAHLO et ROUCAIROL [CAR]. Nous présentons une variante de la méthode de SCHLICHTING et SCHNEIDER [SCS], en la justifiant à l'aide d'une sémantique opérationnelle et en l'illustrant par la preuve d'exclusion mutuelle de l'algorithme précédemment cité. En supposant quelques hypothèses, nous introduisons un système de preuve de fatalité adapté à ce type de programmes et la méthode des treillis de preuve appliquée à ce cas est utilisée pour prouver que toute requête de section critique dans le réseau fait par un site est satisfaite

fatalement. Nous spécifions d'abord la syntaxe des programmes distribués.

V.2.- SYNTAXE ET SEMANTIQUE DES PROGRAMMES DISTRIBUES

Précédemment, au chapitre II, $\mathcal{P}$ désignait la classe des programmes séquentiels non-déterministes. Nous enrichissons cette classe, en lui adjoignant deux structures de base correspondant à l'envoi et à la réception d'un message :

[Send mes to dest] et [Receive mes when cond]

On note $\mathcal{P}[\leftrightarrow]$, la classe obtenue.

Définition [V.2].1 :

$\mathcal{P}[\leftrightarrow]$ est la plus petite classe de programmes telle que :

(1) [v:=e], [v:=?], [Skip], [Send mes to dest], [Receive mes when cond] sont éléments de $\mathcal{P}[\leftrightarrow]$.

où $v \in \mathcal{V}$, $e \in \mathcal{E}$, mes $\in [\leftrightarrow]$, dest $\in \mathcal{N}$, cond $\in \mathcal{B}$.

(2) $\mathcal{P}[\leftrightarrow]$ est stable par application finie des règles suivantes :

(2.1) Si $S_1, S_2 \in \mathcal{P}[\leftrightarrow]$, alors $[S_1 ; S_2] \in \mathcal{P}[\leftrightarrow]$.

(2.2) Si $S_1, S_2 \in \mathcal{P}[\leftrightarrow]$, $B \in \mathcal{B}$, alors [if B then S_1 else S_2] $\in \mathcal{P}[\leftrightarrow]$, [while B do S od] $\in \mathcal{P}[\leftrightarrow]$.

$\mathcal{V}\{[\text{resp. } \mathcal{E}], [\text{resp. } [\leftrightarrow]], [\text{resp. } \mathcal{N}], [\text{resp. } \mathcal{B}]\}$ est l'ensemble des variables {[resp. des expressions], [resp. des messages], [resp. des noms de sites ou de noeuds], [resp. des expressions booléennes]}.

On définit trivialement $\mathcal{C}[\leftrightarrow]$ comme $\mathcal{C}$, à partir de $\mathcal{P}[\leftrightarrow]$.

Définition [V.2].2 :

Soit $\mathcal{P}[\leftrightarrow]$ la classe définie c-dessus.

$\mathcal{C}[\leftrightarrow]$ est la plus petite classe contenant $\mathcal{P}[\leftrightarrow]$ et stable par application de la règle suivante :

si $\mathbb{P}_1 \in \mathcal{P}[\leftrightarrow], \ldots, \mathbb{P}_n \in \mathcal{P}[\leftrightarrow]$, alors [Cobegin $\mathbb{P}_1 \,[\!]\, \ldots \,[\!]\, \mathbb{P}_n$ coend] $\in \mathcal{C}[\leftrightarrow]$

A l'aide de $\mathcal{C}[\leftrightarrow]$, on définit $\mathcal{D}$ la classe des programmes distribués.

Définition [V.2].3 :

$\mathcal{D}$ est la plus petite classe contenant $\mathcal{C}[\leftrightarrow]$ et stable par application de la règle suivante :

si $C\mathbb{P}_1, \ldots, C\mathbb{P}_m \in \mathcal{C}[\leftrightarrow]$, alors [Disbegin $C\mathbb{P}_1 \,[\!]\, \ldots \,[\!]\, C\mathbb{P}_m$ disend] $\in \mathcal{D}$.

Tout élément de $\mathcal{C}[\leftrightarrow]$ sera appelé un processus et tout élément de $\mathcal{P}[\leftrightarrow]$ apparaissant dans un processus sera un sous-processus de celui-ci.

Nous avons ainsi précisé la nature des objets concernés par cette étude, contrairement à SCHLICHTING et SCHNEIDER [SCS] qui disposent de "processus" sans aucune définition formelle de ces objets.

On étend la fonction d'étiquetage à $\mathcal{D}$, notée lab par :

1.- $D\mathbb{P} \equiv$ [Cobegin $\mathbb{P}S_1 \,[\!]\, \ldots \,[\!]\, \mathbb{P}S_n$ coend]

$\underline{lab}(D\mathbb{P}) \equiv (\underline{\ell}_1 : \text{Cobegin } \underline{lab}(\mathbb{P}S_1) \,[\!]\, \ldots \,[\!]\, \underline{lab}(\mathbb{P}S_n) \text{coend} \ ; \ \bar{\ell}_2)$ où

$\underline{\ell}_1 \notin \bigcup_{i=1}^{n} \mathcal{L}[\mathbb{P}S_i]$, $\bar{\ell}_2 \notin \bigcup_{i=1}^{n} \mathcal{L}[\mathbb{P}S_i]$, $\underline{\ell}_1 \in \underline{\mathcal{L}}$, $\bar{\ell}_2 \in \bar{\mathcal{L}}$

$\forall (i,j) \in [n]$, $i \neq j \Rightarrow \mathcal{L}[\mathbb{P}S_i] \cap \mathcal{L}[\mathbb{P}S_j] = \emptyset$.

$\mathcal{L}[D\mathbb{P}] = \{\underline{\ell}_1, \bar{\ell}_2\} \cup \bigcup_{i=1}^{n} \mathcal{L}[\mathbb{P}S_i]$,

2.- $D\mathbb{P} \equiv$ [Disbegin $C\mathbb{P}_1 \,[\!]\, \ldots \,[\!]\, C\mathbb{P}_m$ disend]

$\underline{lab}(D\mathbb{P}) \equiv (\ell : \text{Disbegin } \underline{lab}(C\mathbb{P}_1) \,[\!]\, \ldots \,[\!]\, \underline{lab}(C\mathbb{P}_m) \text{disend} \ ; \ \ell')$

où $\ell \in \underline{\mathcal{L}}$, $\ell' \in \bar{\mathcal{L}}$ et $\ell \notin \bigcup_{i=1}^{n} \mathcal{L}[C\mathbb{P}_i]$, $\ell' \notin \bigcup_{i=1}^{m} \mathcal{L}[C\mathbb{P}_i]$, $\ell \neq \ell'$,

$$\mathcal{L}[D\mathbb{P}] = \{\ell,\ell'\} \cup \bigcup_{i=1}^{n} \mathcal{L}[C\mathbb{P}_i],$$

$$\forall (i,j) \in [m],\ (i \neq j) \Rightarrow \mathcal{L}[C\mathbb{P}_i] \cap \mathcal{L}[C\mathbb{P}_j] = \emptyset.$$

A chaque processus $C\mathbb{P}_i$ de $D\mathbb{P}$, on associe un ensemble de variables noté $\mathcal{V}[C\mathbb{P}_i]$, on supposera que :

$$\forall (i,j) \in [m],\ (i \neq j) \Rightarrow \mathcal{V}[C\mathbb{P}_i] \cap \mathcal{V}[C\mathbb{P}_j] = \emptyset.$$

Avant de donner une sémantique à chaque programme distribué, nous définissons, au préalable, une notion auxiliaire, celle de multiensemble.

Définition [V.2].4 :

(1) Un multiensemble M est un couple formé d'un ensemble $|M|$ et d'une application m de $|M|$ dans $\mathbb{N}$ dont le domaine est $|M|$. On notera $M(x)$, pour $x \in |M|$, où $M(x) = m(x)$.

(2) Soient M_1 et M_2 deux multiensembles.

(i) $M_1 \oplus M_2$ symbolise le multiensemble de domaine $|M_1| \cup |M_2|$ et d'application $m = m_1 + m_2$ sur $|M_1| \cup |M_2|$.

(ii) On dit que $M_1 \subset M_2$ ssi $|M_1| \subset |M_2|$ et $m_1 \leqslant m_2$

(iii) si $M_1 \subset M_2$, on définit $M_2 \ominus M_1$ le multiensemble M tel que $|M| = |M_2| \cup |M_1|$ et $m(x) = m_2(x) - m_1(x)$ pour tout x de $|M|$.

Un ensemble E est un multiensemble, si on lui associe l'application e telle que :

$\forall x \in E,\ e(x) = 1$ et $dom(e) = E$.

Soit $\mathbb{P}$ un processus de $D\mathbb{P}$; on lui associe la notion d'état par la définition suivante.

Définition [V.2].5 :

Soit $\mathbb{P}$ un processus de $D\mathbb{P}$ ($D\mathbb{P} \in \mathcal{D}$) ; $\mathbb{P}S_1,\ldots,\mathbb{P}S_n$ les sous-processus de $\mathbb{P}$; $\mathcal{L}[\mathbb{P}S_i]$ ($i \in \{1,\ldots,n\}$), l'ensemble des étiquettes associées à $\mathbb{P}S_i$; $\mathcal{V}[\mathbb{P}]$ l'ensemble des variables de $\mathbb{P}$.

On appelle état de $\mathbb{P}$, et on note s, le 3-uplet formé d'une multi-étiquette $\bar{\ell}$ de $\mathcal{L}[\mathbb{P}]$, d'une application m de $\mathcal{V}[\mathbb{P}]$ dans $\mathcal{M}[\mathbb{P}]$ et d'un multiensemble appelé tampon de $\mathbb{P}$, noté τ.

$\mathcal{L}[\mathbb{P}]$ est un ensemble dont les éléments sont des multiétiquettes ie : $\bar{\ell} \in \mathcal{L}[\mathbb{P}]$ ssi $\{\bar{\ell} = \underline{\ell}_1$ ou $\bar{\ell} = \bar{\ell}_2$ ou $\bar{\ell} \in \prod_{i=1}^{n} \mathcal{L}[\mathbb{P}S_i]\}$.

$\mathcal{M}[\mathbb{P}]$ désigne l'ensemble des valeurs des variables de $\mathbb{P}$.

On note $S[\mathbb{P}]$ l'ensemble des états de $\mathbb{P}$. On définit une relation sur $S[\mathbb{P}]$, appelée relation de transition qui décrit l'exécution pas-à-pas de chaque élément de $\mathbb{P}$. Les sous-processus $\mathbb{P}S_1,\ldots,\mathbb{P}S_n$ partagent des variables communes. En fait, nous définissons un état en précisant l'état du tampon associé au processus $\mathbb{P}$, sans se référer à des processus qui pourraient éventuellement communiquer ; nous introduisons la notion d'environnement de $\mathbb{P}$ pour un programme $D\mathbb{P}$ contenant $\mathbb{P}$.

Un environnement de $\mathbb{P}$, noté E, est tel que :

si $D\mathbb{P}$ est un programme distribué de $\mathcal{D}$ contenant $\mathbb{P}$,

$E(\mathbb{P}, D\mathbb{P}) = \{1,\ldots,n\}$ où

($D\mathbb{P} \equiv$ Disbegin $\mathbb{P}_1 \| \ldots \| \mathbb{P}_n$ disend) et $\exists i \in \{1,\ldots,n\}$, $\mathbb{P} \equiv \mathbb{P}_i$.

Nous définissons une relation de transition $t[\mathbb{P}]/_E$ pour un environnement donné quelconque de $\mathbb{P}$.

Soient s,s' deux éléments quelconques de $S[\mathbb{P}]$,

E un environnement quelconque de $\mathbb{P}$.

$t[\mathbb{P}]/_E(s,s')$ si et, seulement si,

$\Big[(\underline{lab}(\mathbb{P})\equiv \ell_1:\underline{Cobegin}\ \ell_1:\alpha_1 \| \ldots \| \ell_n:\alpha_n\ \underline{coend};\bar{\ell}_2:)\wedge(s=(\ell_1,m,\tau))\wedge$

$(s'=((\ell_1,\ldots,\ell_n),m,\tau)\Big]\vee$

$\Big[(\underline{lab}(\mathbb{P})\equiv \ell_1:\underline{Cobegin}\ \alpha_1;\ell_1':\| \ldots \alpha_n;\ell_n':\underline{coend};\bar{\ell}_2:)\wedge$

$(s=((\ell_1',..,\ell_n'),m,\tau))\wedge(s'=(\bar{\ell}_2,m,\tau))\Big]\vee$

$\Big[(\mathbb{P}\equiv\underline{Cobegin}\ \mathbb{P}S_1\| \ldots \|\mathbb{P}S_n\underline{coend})\wedge(s=((\ell_1,\ldots,\ell_n),m,\tau)\wedge(s'=((\ell_1',..,\ell_n'),m',\tau')$

$\wedge\Big\{\exists!i\in[n],(\forall j\in[n]\setminus\{i\},(\ell_j=\ell_j'))\wedge(\underline{lab}(\mathbb{P}S_i)\equiv\alpha;\ell_i:S;\ell_i':\beta)\wedge$

$(S\neq[\underline{Receive}\ \text{mes}\ \underline{when}\ \text{cond}])\wedge(S\neq[\underline{Send}\ \text{mes}\ \underline{to}\ \text{dest}])\wedge$

$((S\in\{[v:=e],[v:=?],[\underline{Skip}]\})\vee(S\in\{<B>\,|[\underline{if}\ B\ \underline{then}\ ..\underline{fi}]\})$

$\vee(S\in\{<B>\,|[\underline{while}\ B\ \underline{do}\ S''\ \underline{od}]\}))\wedge$

$t[\mathbb{P}S_i]((\ell_i,m),(\ell_i',m'))\wedge(\tau=\tau')\Big\}\Big]\vee$

$\Big[(s=((\ell_1,\ldots,\ell_n),m,\tau)\wedge(s'=((\ell_1',\ldots,\ell_n'),m',\tau'))\wedge(m'=m[^{vm}/_{mes}])\wedge$

$\exists i\in[n],(\forall j\in[n]\setminus\{i\},\ell_j=\ell_j')\wedge(\underline{lab}(\mathbb{P}S_i)\equiv\alpha;\ell:\underline{Receive}\ \text{mes}\ \underline{when}\ \text{cond};\ell':\beta)$

$\wedge(\ell_i=\ell)\wedge(\ell_i'=\ell')\}\wedge\text{cond}[vm]\wedge(\tau=\tau'\oplus\{vm\})\Big]$

$\vee\Big[(s=((\ell_1,\ldots,\ell_n),m,\tau)\wedge(s'=((\ell_1',\ldots,\ell_n'),m',\tau'))\wedge(m=m')\wedge$

$(\exists i\in[n],(\forall j\in[n]\setminus\{i\},\ell_j=\ell_j')\wedge(\underline{lab}(\mathbb{P}S_i)\equiv\alpha;\ell:\underline{Send}\ \text{mes}\ \underline{to}\ \text{dest};\ell':\beta)\wedge$

$(\ell_i=\ell)\wedge(\ell_i'=\ell')\vee(\exists!d\in E,(\text{dest}=d)\wedge(\tau_d'=\tau_d\oplus\{\text{mes}[m]\}))\Big]$

On note mes[m], l'évaluation de mes dans l'état mémoire m de $\mathbb{P}$ et τ_d,τ_d' désignent les tampons de d respectivement avant livraison de mes[m] et après livraison de mes[m] à d.

On appelle sémantique opérationnelle de $\mathbb{P}$ dans l'environnement E le couple $(S[\mathbb{P}], t[\mathbb{P}]/_E)$. On définit une telle sémantique pour tout DIP de $\mathcal{D}$.

Définition [V.2].6 :

Soit DIP un programme distribué, $\mathbb{P}_1,\ldots,\mathbb{P}_m$ les processus composant DIP.

On appelle état de DIP, et on note s, tout m-uplet $s = (s_1,\ldots,s_m)$ où s_i est un état de $\mathbb{P}_i$.

S[DIP] est l'ensemble des états de DIP et est égal à $\prod_{i=1} S[\mathbb{P}_i] \cup \{s_{in},s_{fin}\}$ où s_{in} et s_{fin} sont les états tels que le contrôle se trouve avant Disbegin et après disend respectivement.

On définit alors une relation de transition t[DIP] sur S[DIP] par :

soient s,s' deux états de DIP :

$E_1 = E_2 = \ldots = E_m = \{1,\ldots,m\} = E$.

$t[DIP](s,s')$ si, et seulement si,

$\Big[(\underline{lab}(DIP)\equiv\ell:\underline{Disbegin}\ \ell_1:\alpha_1\| \ldots \|\ell_m:\alpha_m\ \underline{coend};\ell':)\wedge$

$(\forall i\in E,(s_i'=(\ell_i,m_i,\tau_i)))\wedge(\forall i\in E,(s_i=(\ell,m_i,\tau_i)))\wedge$

$(s=(s_1,\ldots,s_m))\wedge(s'=(s_1',\ldots,s_m'))\Big]\vee$

$\Big[(\underline{lab}(DIP\equiv\ell:\underline{Disbegin}\ \alpha_1;\ell_1':\| \ldots \|\alpha_m;\ell_m':\underline{coend};\ell':)\wedge$

$(\forall i\in E,(s_i'=(\ell',m_i,\tau_i)))\wedge(\forall i\in E,s_i=(\ell_i',m_i,\tau_i))\wedge$

$(s=(s_1,\ldots,s_m))\wedge(s'=(s_1',\ldots,s_m'))\Big]\vee$

$\Big[\binom{\exists i\in E}{\exists j\in E},\ t[\mathbb{P}_i]_E(s_i,s_i')\wedge(\forall k\in E\setminus\{i,j\},s_k=s_k')$

$\wedge((i\neq j)\Rightarrow(\ell_j=\ell_j')\wedge(m_j=m_j')\wedge(\tau_j'=\tau_j\oplus\{\text{mes}[m_j]\})$

$\wedge((i=j)\Rightarrow(\tau_i'=\tau_i\oplus\{\text{mes}[m_i]\}))\Big]$.

On appelle sémantique opérationnelle de DIP le couple (S[DIP], t[DIP]). Ainsi peut-on associer à tout programme distribué une sémantique. Nous avons supposé que tout message envoyé est immédiatement délivré à son destinataire ; ceci revient à supposer que le moyen de communication est complètement fiable

et de plus cela revient à supposer cette opération comme atomique. LAMPORT [LAM3] définit une méthode d'approche de la correction de programme sans considérer la notion d'atomicité et il serait intéressant de regarder si sa méthode peut se généraliser ici.

Nous supposons ici encore qu'il existe deux fonctions que l'on peut utiliser chaque fois que l'on a besoin d'une assertion et que l'on dispose d'une définition ensembliste de celle-ci. La notion d'action atomique est liée à la sémantique opérationnelle et comprendra deux nouveaux cas possibles : l'envoi et la réception. Notre intérêt se portera sur deux types de propriétés de DIP : l'invariance et la fatalité.

Comme pour le cas des programmes parallèles, on étend les notions de langage d'assertions, de traces, de modèle de fatalité pour le cas des programmes distribués.

En fait, les notions restent identiques modulo la substitution de t[DIP] à t[CIP], de S[DIP] à S[DIP].

Avant d'entreprendre l'étude de l'algorithme cité auparavant, nous précisons la méthode de preuves d'invariance que nous utiliserons. Elle sera appliquée à la preuve de l'exclusion mutuelle dans l'algorithme amélioré par CARVAHLO et ROUCAIROL [CAR] et dû à RICART et AGRAWALA [RIA]. Nous illustrerons le système OL étendu à ce type de programme, en prouvant que tout site demandant sa section critique l'obtiendra fatalement.

V.3.- PREUVES DE PROPRIETES D'INVARIANCE

V.3.1.- Généralités

SCHLICHTING et SCHNEIDER [SCS] donnent une méthode à la HOARE, pour prouver des propriétés d'invariance de programmes distribués dans le cas où la communication est asynchrone via un tampon. Nous exposons de façon condensée les trois points de cette méthode :

- annoter séquentiellement chaque processus de DIP.
- prouver la satisfaction des assertions figurant devant les primitives de communication : ie montrer que les communications se font correctement par rapport aux assertions.
- prouver l'absence d'interférence.

Cependant nous ne considérons pas la même décomposition de l'invariant global et le point 2 sera un point particulier du point 3. L'axiome sur la primitive de réception (receive) nous paraît trop grossier et nous l'avons quelque peu modifié en tenant compte du tampon comme variable partagée par chaque processus de DIP et de la sémantique opérationnelle de DIP. Notre langage d'assertions $\mathcal{A}$[DIP] est supposé suffisamment expressif au sens de COOK [COO]. Notre méthode est une extension de celle d'OWICKI et GRIES [OWG] en utilisant les compteurs (at, in, after) à la place des variables auxiliaires ; nous pensons que l'axiome "miraculeux" du receive complique la preuve et la rend plus longue.

Nous proposons alors la méthode suivante issue d'un compromis entre celle d'OWICKI et GRIES [OWG], COUSOT R. [COU] et SCHICHTING et SCHNEIDER [SCS] :

(1) Annoter séquentiellement le programme DIP en tenant compte de l'assertion I à prouver invariante.

(2) Prouver les absences d'interférences.

(3) Pour le point (1) , utiliser les axiomes et règles à la HOARE-OWICKI-GRIES.

$P, Q, P_1, Q_1, P_2, P_3 \in \mathcal{A}[DIP]$,

A1 : $\{P\}$ Skip $\{P\}$

A2 : $\{P[e/_v]\}v:=e\{P\}$

A3 : $\{P[\tau_{dest} \oplus \{mes\}/_{\tau_{dest}}]\}$Send mes to dest$\{P\}$

A4 : $\{P[\tau \ominus \{mes\}/_{\tau}] \wedge (mes \in \tau) \wedge cond[mes]\}$

Receive mes when cond

$\{P \wedge cond\}$

où τ, τ_{dest} sont des tampons (considérés comme multiensembles)

R1 : $\dfrac{\{P \wedge B\}S1\{Q\},\ \{P \wedge \sim B\}S2\{Q\}}{\{P\}\ \underline{if}\ B\ \underline{then}\ S1\ \underline{else}\ S2\ \underline{fi}\{Q\}}$

R2 : $\dfrac{\{P \wedge b\}S\{P\}}{\{P\}\underline{while}\ B\ \underline{do}\ S\ \underline{od}\{P \wedge \sim B\}}$

R3 : $\dfrac{\{P_1\}S1\{P_2\},\ \{P_2\}S2\{P_3\}}{\{P_1\}S1;S2\{P_3\}}$

R4 : $\dfrac{P \Rightarrow P_1, \{P_1\}S\{Q_1\}, Q_1 \Rightarrow Q}{\{P\}S\{Q\}}$

Il nous reste à préciser la notion d'interférence relative à ces propriétés

Définition [V.3.1].1 :

Soit P une assertion de l'annotation séquentielle de DIP.

S une instruction de DIP telle que [v:=e], [v:=?], [Skip], [Receive mes when cond], [Send mes to dest], une partie séquentielle s'exécutant atomiquement.

On suppose que P et S sont concurrentes dans DIP ie ils n'appartiennent pas au même processus ou même sous-processus.

On dit que S n'interfère pas avec P, si

$\{P \wedge pré(S)\}S\{P\}$ est vrai, où pré(S) est l'assertion de l'annotation séquentielle se trouvant devant S.

Définition [V.3.1].2 :

Soient $IPS_1, \ldots, IPS_n$ les programmes séquentiels constituant CIP un processus de DIP. On dit que $\{P_1\}IPS_1\{Q_1\}, \ldots, \{P_n\}IPS_n\{Q_n\}$ n'interfèrent pas, si, pour chaque i de $\{1, \ldots, n\}$, pour chaque instruction S de IPS_j où $j \neq i$, S n'interfère pas avec chaque annotation de IPS_i.

Soit la règle suivante proposée par OWICKI et GRIES [OWG] :

R5 : $\dfrac{\{P_1\}IPS_1\{Q_1\}, \ldots, \{P_n\}IPS_n\{Q_n\} \quad \text{n'interfèrent pas}}{\{\bigwedge_{i=1}^{n} P_i\}\underline{Cobegin}\ IPS_1 \| \ldots \| IPS_n\ \underline{coend}\{\bigwedge_{i=1}^{n} Q_i\}}$

Cette règle nous permet de déduire des propriétés au niveau du processus et il nous faut ensuite prouver que ces propriétés vérifiées localement par processus restent vraies au niveau du programme distribué.

Ainsi nous proposons une règle relative à l'interférence possible entre processus.

Définition [V.3.1].3 :

Soient $\mathbb{P}_1,\ldots,\mathbb{P}_m$ les processus de DIP. On dit que $\{P_1\}\mathbb{P}_1\{Q_1\},\ldots,\{P_m\}\mathbb{P}_m\{Q_m\}$ n'interfèrent pas si, pour chaque i de $\{1,\ldots,m\}$, pour chaque instruction S d'un autre processus, S n'interfère pas avec chaque annotation de $\mathbb{P}_i$.

$$R6 : \frac{\{P_1\}\mathbb{P}_1\{Q_1\},\ldots,\{P_m\}\mathbb{P}_m\{Q_m\} \quad \text{n'interfèrent pas}}{\{\bigwedge_{i=1}^{m} P_i\}\underline{\text{Disbegin}}\ \mathbb{P}_1 \| \ldots \| \mathbb{P}_m\ \underline{\text{disend}}\{\bigwedge_{i=1}^{m} Q_i\}}$$

V.3.2.- Preuve de l'exclusion mutuelle pour l'algorithme de RICART et AGRAWALA [RIA] amélioré par CARVAHLO et ROUCAIROL [CAR]

V.3.2.1.- Description de l'algorithme

RICART et AGRAWALA [RIA] présentent un algorithme permettant de réaliser l'exclusion mutuelle dans un réseau de N sites communiquant par des messages de façon asynchrone. Leur algorithme exige à chaque invocation de section critique exactement 2*(N-1) messages et ils en donnent une preuve informelle, quant aux propriétés exigées pour de tels algorithmes. Le principe de cet algorithme est le suivant : afin de garantir une certaine priorité au niveau des requêtes de section critique, on définit localement un système d'estampillage de style LAMPORT ; cet estampillage local possède une référence globale liée aux messages reçus ; globalement, tout site, qui demande sa section critique et qui n'y est pas encore, est ordonné totalement par rapport aux autres sites faisant la même requête. La priorité ainsi définie est stricte et permet de garantir l'exclusion mutuelle et l'absence de famine. CARVAHLO et ROUCAIROL [CAR] proposent une amélioration de l'algorithme de RICART et AGRAWALA [RIA],

en diminuant le nombre de messages par invocation de section critique et ce nombre est alors inférieur ou égal à 2*(N-1). En effet, ils remarquent que, si un site i a obtenu l'autorisation d'entrer en section critique de la part d'un autre site j et si ce dernier j n'a pas demandé la sienne, alors le site i peut se passer de l'autorisation de j. Nous adopterons alors cette nouvelle version qui ne demande que de considérer un nouveau type de message supplémentaire. Nous décrivons plus précisément l'algorithme et la structure de donnée utilisée par la suite.

Nous désignons par RA l'algorithme, qui se compose de N processus parallèles appelés noeuds et on dit que RA est un réseau. Chaque noeud du réseau RA contient la même copie des processus qui invoquent la section critique, la rendent ou traitent les messages.

Chaque noeud de RA se réfère à son numéro ME ; chaque noeud ME dispose d'une copie du même processus parallèle noté CP(ME) qui est composé de deux processus qui bouclent indéfiniment et qui se partagent les variables OSN, HSN et WAITING permettant d'établir l'ordre de priorité : OSN est le numéro accordé au site ME, HSN est le plus grand numéro accordé aux sites ayant envoyés un message à ME et tel que ME l'ait reçu, WAITING signale que ME attend sa section critique. Le premier processus est composé de deux parties : une qui demande la section critique et une qui la rend. Le second processus traite les différents messages présents dans le tampon du site ME. Le scénario de demande de section critique est alors le suivant :

Le noeud ME demande sa section critique et, pour cela, initialise OSN et WAITING. Puis il envoie des messages de requêtes aux autres sites, qui ne lui ont pas auparavant accordé l'entrée et pour cela on utilise le tableau $(A[j])_{j\neq ME}$ qui est vrai en j, si j a autorisé ME. ME attend les N-1

réponses ie il attend que A soit vrai. Au cours de cette attente il diffère les requêtes de noeuds de priorité moindre, en utilisant l'ordre total d'estampillage défini par OSN et ME car chaque message contient un numéro et un numéro de site ; il positionne ainsi un tableau $(R\text{-}D[j])_{j \neq ME}$ vrai en j si ME diffère sa réponse à j. Enfin, si ME entre en section critique, il positionne USING à vrai. Au cours de la restitution de sa section critique, ME envoie des réponses à tout noeud j tel que R-D[j] est vrai et positionne ceci à faux. Ceci permet à j de progresser vers sa section critique. Nous donnons quelques remarques sur l'algorithme de CARVAHLO et ROUCAIROL [CAR] relatifs à sa correction. Une première tentative de correction nous a conduit à un algorithme conduisant à un blocage total et ceci était dû à une mauvaise spécification de l'emboîtement des divers processus qui n'avait pas été précisé par CARVAHLO et ROUCAIROL [CAR]. Nous avons donc modifié la présentation de l'algorithme RA pour utiliser notre syntaxe ; nous proposons l'algorithme suivant.


```
CP(ME)::= Cobegin
        while true do
            REQUEST-RESOURSE;
            C-S[ME];
            RELEASE-RESOURSE
        od
        ||
        while true do
            Receive mes when ⋁_{j=1}^{N} [(mes='Reply(j)')
                                          (mes="Reply-Request(HSN,j)')
                                          (mes='Request(HSN,j)')];
            if(mes="Reply(j)')then REPLY-MESSAGE
            else if (mes="Reply-Request(HSN,j)')then
                        REPLY-REQUEST-MESSAGE
                 else REQUEST-MESSAGE
                 fi
            fi
        od
        Coend
```

On note ME un site quelconque du réseau de N sites.

Structure de donnée au noeud ME

CONSTANT ME,N ;

INTEGER OSN initial (0) ;

HSN initial (0) ;

BOOLEAN A[1..N] initial false ;

USING initial false ;

WAITING initial false ;

R-D[1..N] initial false ;

Nous ne donnerons pas les parties en caractères majuscules, mais nous les préciserons dans les annotations séquentielles par la suite.

V.3.2.2.- Preuve de l'algorithme : exclusion mutuelle

Un premier critère de correction pour cet algorithme est l'exclusion mutuelle. Il s'agit de prouver que l'assertion suivante est invariante pour cet algorithme :

$$MUTEX \equiv \bigwedge_{\substack{(i,j)\in[N]^2 \\ i\neq j}} [\sim(CS_i \wedge CS_j)]$$ où CS_k signifie que k est en section critique et, en se reportant aux preuves ci-après et notations, on définit CS_k par :

$$CS_k \equiv [(\underline{in}\ C\text{-}S[k]) \vee (\underline{at}_k 18) \vee (\underline{at}_k 19) \vee (\underline{at}_k 20) \vee (\underline{at}_k 21)]$$

$\underline{at}_k S$ signifie que le contrôle est devant l'instruction S du site k.

V.3.2.2.1.- Preuve séquentielle de RA

Nous divisons la preuve séquentielle et donnons dans la suite la preuve séquentielle associée aux diverses parties de RA.

On notera $M_{ME,R}$, le multiensemble contenant les messages reçus par ME et $M_{ME,S}$, le multiensemble contenant les messages envoyés à ME. L'état du tampon τ de ME est donc tel que :

$$\tau = M_{ME,S} \ominus M_{ME,R} .$$

Au cours de notre propos, nous utiliserons les prédicats suivants : W_{ME} , U_{ME} , A_{ME} , $R\text{-}D_{ME}$ dont le sens est :

W_{ME} est vrai ssi $(WAITING_{ME} = \underline{True})$

U_{ME} est vrai ssi $(USING_{ME} = \underline{True})$

A_{ME} est vrai ssi $\bigwedge_{\substack{j=1 \\ j\neq ME}}^{N} \left[A_{ME}[j] = \underline{True}\right]$.

$R\text{-}D_{ME}$ est vrai ssi $\bigwedge_{\substack{j=1 \\ j\neq ME}}^{N} \left[R\text{-}D_{ME}[j] = \underline{True}\right]$.

[N] désigne le sous-ensemble de $\mathbb{N}$, $\{1,\ldots,N\}$.

Afin de repérer chaque variable de chaque site ME, nous indiquerons celle-ci à l'aide de ME, si ME est le numéro du site concerné : OSN_{ME} désignera la variable du site ME, lorsqu'on l'utilisera comme variable dans les assertions.

V.3.2.2.1.1.- Preuve séquentielle de RA

On définit les notations auxiliaires suivantes pour les assertions dont nous aurons besoin de définir par la suite.

$M\text{-}RQ_{ME}$ est un multiensemble qui contient les numéros de sites n'ayant

pas autorisé ME à entrer en section critique mais à qui ME a envoyé une requête qui est dans le tampon du site :

$$i \in \text{M-RQ}_{ME} \quad \text{ssi} \sim A_{ME}[i] \wedge (\text{REQUEST(OSN,ME)} \in M_{i,S} \ominus M_{i,R})$$

M-RP_{ME} est un multiensemble contenant le numéro des sites n'ayant pas autorisé ME à entrer en section critique et à qui ME n'a pas autorisé à entrer en section critique mais la réponse de chaque site est à traiter par ME :

$$i \in \text{M-RP}_{ME} \quad \text{ssi} \sim A_{ME}[i] \wedge \sim A_i[ME] \wedge (\text{REPLY(i)} \in M_{ME,S} \ominus M_{ME,R})$$

M-AUT_{ME} est un multiensemble contenant le numéro des sites qui ont autorisé ME à entrer en section critique et desquels ME a reçu le message de réponses :

$$i \in \text{M-AUT}_{ME} \quad \text{ssi} \quad A_{ME}[i] \wedge [(\text{REQUEST(OSN,ME)} \in M_{i,R}) \Rightarrow (\text{REPLY(i)} \in M_{ME,R})].$$

M-RR_{ME} est un multiensemble contenant le numéro des sites à qui ME a envoyé une réponse-requête à la suite d'une priorité inférieure à celle d'un site demandant sa section critique :

$$i \in \text{M-RR}_{ME} \quad \text{ssi} \quad (\text{REPLY-REQUEST(OSN,ME)} \in M_{i,S} \ominus M_{i,R}).$$

M-RD_{ME} est un multiensemble contenant le numéro des sites différant leur réponse à ME et à qui ME a autorisé l'entrée en section critique :

$$i \in \text{M-RD}_{ME} \quad \text{ssi} \quad \text{R-D}_i[ME] \wedge \sim A_{ME}[i] \wedge A_i[ME].$$

M-TRAIT_{ME} est un multiensemble qui contient le numéro des sites qui ont répondu à ME et dont on traite la réponse ou à qui ME a envoyé une requête traitée par ce site ou à qui ME a envoyé une réponse-requête traitée par ce site :

$$\begin{aligned} i \in \text{M-TRAIT}_{ME} \quad \text{ssi} \; \{ & [(\text{REPLY(J)} \in M_{ME,R}) \wedge (J=i)] \vee \\ & [(\text{REQUEST(OSN,ME)} \in M_{i,R}) \wedge (\text{TSN=OSN}) \wedge (J=ME)] \\ & \vee [(\text{REPLY-REQUEST(OSN,ME)} \in M_{i,R}) \wedge (\text{TSN=OSN}) \wedge (J=ME)] \} \end{aligned}$$

Nous utilisons les assertions suivantes dans la preuve séquentielle :

$$\begin{aligned} *I_{ME} \equiv \{ & [\text{M-RP}_{ME} = \text{M-RQ}_{ME} = \text{M-AUT}_{ME} = \text{M-RR}_{ME} = \text{M-RD}_{ME} = \text{M-TRAIT}_{ME} = \emptyset] \wedge \\ & \sim \text{R-D}_{ME} \wedge \sim U_{ME} \wedge \sim W_{ME} \wedge [\forall \ell \in [N] - \{ME\}, W_\ell \Rightarrow (ME \in \text{M-RD}_\ell)] \wedge \\ & [\forall \ell \in [N] \setminus \{ME\}, (\ell \notin \text{M-RP}_{ME} \cup \text{M-AUT}_{ME} \cup \text{M-RR}_{ME})] \wedge [M_{ME,R} = \emptyset] \wedge \\ & [\forall \ell \in [N] \setminus \{ME\}, W_\ell \Rightarrow (\text{REQUEST}(OSN_\ell, \ell) \notin M_{ME,R})] \}. \end{aligned}$$

Cette assertion décrit les conditions initiales du processus de chaque site ME de RA.

$$\begin{aligned} *I_1 \equiv \{ & [\text{M-RP}_{ME} = \text{M-RQ}_{ME} = \text{M-AUT}_{ME} = \text{M-RR}_{ME} = \text{M-RD}_{ME} = \text{M-TRAIT}_{ME} = \emptyset] \wedge \\ & \sim \text{R-D}_{ME} \wedge \sim U_{ME} \wedge \sim W_{ME} \wedge [\forall \ell \in [N] \setminus \{ME\}, W_\ell \Rightarrow (ME \notin \text{M-RD}_\ell)] \wedge \\ & [\forall \ell \in [N] \setminus \{ME\}, (\ell \notin \text{M-RP}_{ME} \cup \text{M-AUT}_{ME} \cup \text{M-RR}_{ME})] \}. \end{aligned}$$

Cette assertion décrit les variables du sous-processus de RA, qui demande la section critique, au début du cycle : aucune demande n'a été faite encore,

$$*I_2 \equiv \{(M_{ME,R} = \emptyset) \wedge (\bigwedge_{J=1}^{N} \sim A_J[ME])\}$$

I_2 décrit les variables du deuxième sous-processus de RA qui traite les messages et avant tout traitement, aucune autorisation n'est accordée.

$$*I_3 \equiv \{[(mes = REPLY(j)) \Rightarrow (K=j)] \wedge$$
$$[(mes = REPLY\text{-}REQUEST(n,j)) \Rightarrow (TSN=n) \wedge (J=j)] \wedge$$
$$[(mes = REQUEST(n,j)) \Rightarrow (TSN=n) \wedge (J=j)]\}.$$

I_3 décrit l'état des variables après réception d'un message et avant son traitement par ME.

$$*I_4 \equiv \{[(mes = REPLY(j)) \Rightarrow ((W_{ME} \vee U_{ME} \vee (\bigwedge_{j=1}^{N} A_{ME}[j]) \wedge \sim W_{ME} \wedge \sim U_{ME}))$$
$$\Rightarrow (j \in M\text{-}AUT_{ME}))] \wedge$$
$$[(mes = REPLY\text{-}REQUEST(n,j)) \Rightarrow [((W_{ME} \vee U_{ME} \vee (\bigwedge_{j=1}^{N} A[j]) \wedge \sim W_{ME} \wedge \sim U_{ME}))$$
$$\Rightarrow (j \in M\text{-}AUT_{ME})) \wedge (W_j \Rightarrow (ME \in M\text{-}RD_j))]] \wedge$$
$$[(mes = REQUEST(n,j)) \Rightarrow (W_{ME} \Rightarrow (ME \notin M\text{-}RQ_j))]\}.$$

I_4 décrit l'état des variables, après réception et traitement des messages.

On propose alors la preuve séquentielle suivante globale décrivant tout RA

$$0:\{\bigwedge_{J=1}^{N} [M_{J,R}=M_{J,S}=\emptyset) \wedge \sim A_J \wedge \sim R\text{-}D_J \wedge (OSN_J=HSN_J=0) \wedge \sim W_J \wedge \sim U_J]\}$$

DISBEGIN

$$\Downarrow$$

$$\Diamond$$

$$ME:\{(OSN_{ME}=HSN_{ME}=0) \wedge \sim W_{ME} \wedge \sim U_{ME} \wedge \sim A_{ME} \wedge \sim R\text{-}D_{ME} \wedge (M_{ME,R}=\emptyset) \wedge I_{ME}\}$$

COBEGIN

$$1:\{(OSN_{ME}=0) \wedge \sim W_{ME} \wedge \sim U_{ME} \wedge \sim A_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_1\}$$

while true do

{REQUEST-RESOURSE};

C-S[ME];

{RELEASE-RESOURSE}

od

01:{false}

[]

$$2:\{(HSN_{ME}=0) \wedge \sim A_{ME} \wedge \sim R\text{-}D_{ME} \wedge \sim U_{ME} \wedge I_2\}$$

while true do

$$21:\{(HSN_{ME}=Sup^{*}_{ME})\}$$

$$\text{Receive mes when}\{\bigvee_{\substack{J=1 \\ j\neq ME}}^{N} [(mes=REPLY(j)) \vee (mes=REPLY\text{-}REQUEST(n,j)) \vee (mes=REQUEST(n,j))]\};$$

$$22:\{(HSN_{ME}=Sup^{+}_{ME}) \wedge I_3\}$$

If mes=REPLY(J)then 2210:REPLY-MESSAGE

else if mes=REPLY-REQUEST(n,J)then 2220:REPLY-REQUEST-MESSAGE

else 2230:REQUEST-MESSAGE

fi fi

$$23:\{(HSN_{ME}=Sup^{*}_{ME}) \wedge I_4\}$$

od

02:{false}

COEND

0.ME:{false}

$$\Diamond$$

$$\}$$

DISEND

00:{false}

V.3.2.2.1.2.- Preuve séquentielle de {REQUEST-RESOURSE}

Comme précédemment nous aurons besoin de notations auxiliaires et nous les donnons dans ce qui suit,

$*I_{10} \equiv I_1$ (cf : la preuve précédente).

$$*I_{11} \equiv \{[\forall \ell \in [N] \setminus \{ME\}, (W_\ell \wedge (OSN_{ME}, ME) < (OSN_\ell, \ell)) \Rightarrow (ME \notin M\text{-}RP_\ell \cup M\text{-}AUT_\ell \cup M\text{-}RR_\ell)] \wedge [\forall \ell \in [N] \setminus \{ME\}, (W_\ell \wedge (OSN_\ell, \ell) < (OSN_{ME}, ME)) \Rightarrow (\ell \notin M\text{-}RP_{ME} \cup M\text{-}AUT_\ell \cup M\text{-}RR_\ell)] \wedge [(M\text{-}TRAIT_{ME} = M\text{-}RP_{ME} = M\text{-}AUT_{ME} = M\text{-}RQ_{ME} = M\text{-}RR_{ME} = M\text{-}RD_{ME} = \emptyset)]\}$$

I_{11} permet de décrire l'état, après que ME a fait sa demande de section critique et ajusté son numéro OSN_{ME}. A ce moment, aucune requête n'est envoyée et on établit ME dans l'ordre total d'estampillage.

$$*I_{12}[J] \equiv \{[\forall \ell \in [N] \setminus \{ME\}, [(W_\ell \wedge (OSN_{ME}, ME) < (OSN_\ell, \ell)) \Rightarrow (ME \notin M\text{-}RP_\ell \cup M\text{-}AUT_\ell \cup M\text{-}RR_\ell)] \wedge [(\ell \in [J]) \Rightarrow (\ell \in M\text{-}RQ_{ME} \cup M\text{-}AUT_{ME} \cup M\text{-}TRAIT_{ME} \cup M\text{-}RP_{ME} \cup M\text{-}RR_{ME})]] \wedge [\forall \ell \in [N] \setminus \{ME\}, (W_\ell \wedge (OSN_\ell, \ell) < (OSN_{ME}, ME)) \Rightarrow ((\ell \notin M\text{-}RP_{ME} \cup M\text{-}AUT_{ME} \cup M\text{-}RR_{ME}) \wedge ((\ell \in [J]) \Rightarrow (\ell \in M\text{-}RQ_{ME} \cup M\text{-}RD_{ME} \cup M\text{-}TRAIT_{ME})))] \wedge [(M\text{-}TRAIT_{ME} \cup M\text{-}RP_{ME} \cup M\text{-}RR_{ME} \cup M\text{-}AUT_{ME} \cup M\text{-}RQ_{ME} \cup M\text{-}RD_{ME} = [J] \setminus \{ME\})]$$

Cette assertion décrit l'état d'avancement des envois de requête par ME aux sites [J] et l'ordre total est toujours valable.

$$*I_{13} \equiv \{[\forall \ell \in [N] \setminus \{ME\}, (W_\ell \wedge (OSN_{ME}, ME) < (OSN_\ell, \ell)) \Rightarrow [(ME \notin M\text{-}RP_\ell \cup M\text{-}AUT_\ell \cup M\text{-}RR_\ell) \wedge (\ell \in (M\text{-}RQ_{ME} \cup M\text{-}AUT_{ME} \cup M\text{-}RR_{ME} \cup M\text{-}RP_{ME}) \setminus M\text{-}TRAIT_{ME})] \wedge [\forall \ell \in [N] \setminus \{ME\}, (W_\ell \wedge (OSN_\ell, \ell) < (OSN_{ME}, ME)) \Rightarrow [(\ell \notin M\text{-}RP_{ME} \cup M\text{-}AUT_{ME} \cup M\text{-}RR_{ME}) \wedge (\ell \in M\text{-}RQ_{ME} \cup M\text{-}RD_{ME} \cup M\text{-}TRAIT_{ME})]] \wedge [(M\text{-}TRAIT_{ME} \cup M\text{-}RQ_{ME} \cup M\text{-}RR_{ME} \cup M\text{-}AUT_{ME} \cup M\text{-}RP_{ME} \cup M\text{-}RD_{ME} = [N] - \{ME\})]\}.$$

Tous les messages de requête sont envoyés par ME aux autres sites et celui-ci attend les réponses modulo l'ordre total d'estampillage.

$$I_{14} \equiv \{[\bigwedge_{J=1}^{N} A_{ME}[J]] \wedge [\forall \ell \in [N] \setminus \{ME\}, (W_\ell \Rightarrow (OSN_{ME}, ME) < (OSN_\ell, \ell))] \wedge [\forall \ell \in [N] \setminus \{ME\}, (W_\ell \Rightarrow (ME \notin (M\text{-}RP_\ell \cup M\text{-}AUT_\ell \cup M\text{-}RR_\ell)))] \wedge [(M\text{-}AUT_{ME} = [N] \setminus \{ME\}) \wedge (M\text{-}RQ_{ME} = M\text{-}RR_{ME} = M\text{-}TRAIT_{ME} = M\text{-}RD_{ME} = M\text{-}RQ_{ME} = \emptyset)] \wedge [W_{ME}]\}.$$

ME n'attend aucune réponse ; il les a toutes eues. Il lui reste à entrer effectivement en section critique.

Nous donnons maintenant la preuve séquentielle suivante à l'aide des assertions ci-dessus.

11 :{$\sim W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_{10}$}
WAITING:=TRUE;
OSN:=HSN+1;

12 : $\{W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_{11}\}$

for J:=1 to N do

13 : $\{W_{ME} \wedge \sim U_{ME} \wedge I_{12}[J-1]\}$

if (J≠ME) and not A[J] then

14 : $\{W_{ME} \wedge \sim U_{ME} \wedge \sim A_{ME}[J] \wedge I_{12}[J-1]\}$

Send REQUEST(OSN,ME) to J ;

15 : $\{W_{ME} \wedge \sim U_{ME} \wedge I_{12}[J]\}$

fi

16 : $\{W_{ME} \wedge \sim U_{ME} \wedge I_{12}[J]\}$

end-for

17 : $\{W_{ME} \wedge \sim U_{ME} \wedge I_{13}\}$

wait for $(\bigwedge_{J=1}^{N} A[J])$;

18 : $\{W_{ME} \wedge \sim U_{ME} \wedge I_{14}\}$

WAITING:=false;

19 : $\{W_{ME} \wedge \sim U_{ME} \wedge I_{14}\}$

USING:=true;

1.10 : $\{\sim W_{ME} \wedge U_{ME} \wedge I_{14}\}$.

V.3.2.2.1.3.- Preuve séquentielle de {RELEASE-RESOURSE}

Comme pour les cas précédents nous donnons quelques assertions dont nous aurons besoin dans l'annotation séquentielle de RELEASE-RESOURSE.

$$*I_{15} \equiv \Big[\forall \ell \in [N] \setminus \{ME\}, (W_\ell \wedge [(REQUEST(OSN_\ell,\ell) \in M_{ME,S} \ominus M_{ME,R}) \vee ((REQUEST(OSN_\ell,\ell) \in M_{ME,R}) \wedge (ME \in M\text{-}TRAIT_\ell))]) \Rightarrow (ME \in M\text{-}AUT_\ell \cup M\text{-}RP_\ell \cup M\text{-}RQ_\ell \cup M\text{-}TRAIT_\ell)\Big].$$

Le site ME n'utilise plus sa section critique ; il en est sorti.

Les messages de requête traités alors sont favorablement reçus et ME répond alors au site demandant :

$$*I_{16}[J] \equiv \Big\{\Big[\bigwedge_{K=1}^{J-1} \sim R\text{-}D_{ME}[K]\Big] \wedge \Big[\forall \ell \in [J-1] \setminus \{ME\}, (W_\ell \wedge ((OSN_{ME},ME) < (OSN_\ell,\ell))) \Rightarrow ((ME \in M\text{-}RP_\ell \cup M\text{-}AUT_\ell \cup M\text{-}TRAIT_\ell \cup M\text{-}RQ_\ell) \wedge (ME \notin M\text{-}RD_\ell))\Big] \wedge \Big[\forall \ell \in [J-1] \setminus \{ME\}, \ell \notin M\text{-}RP_{ME} \cup M\text{-}RR_{ME} \cup M\text{-}AUT_{ME}\Big]\Big\}.$$

Cette assertion décrit l'avancement des réponses aux sites bloqués à cause de ME au niveau de J.

Nous proposons alors la preuve séquentielle suivante de la partie RELEASE-RESOURSE.

1.11 : $\{\sim W_{ME} \wedge U_{ME} \wedge I_{14}\}$

USING:=false;

1.12 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{J=1}^{N} [R\text{-}D_{ME}[J] \Rightarrow ((ME \in M\text{-}RD_J) \wedge A_{ME}[J])] \wedge I_{15}\}$

for J:=1 to N do

1.13 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{K=J}^{N} [R\text{-}D_{ME}[K] \Rightarrow ((ME \in M\text{-}RD_K) \wedge A_{ME}[K])] \wedge I_{16}[J]\}$

if $R\text{-}D_{ME}[J]$ then

1.14 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{K=J}^{N} [R\text{-}D_{ME}[K] \Rightarrow ((ME \in M\text{-}RD_K) \wedge A_{ME}[K]] \wedge I_{16}[J]\}$

A[J]:=false;

1.15 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{K=J+1}^{N} [R\text{-}D_{ME}[K] \Rightarrow ((ME \in M\text{-}RD_K) \wedge A_{ME}[K]] \wedge \sim A_{ME}[J] \wedge (ME \in M\text{-}RD_J) \wedge R\text{-}D_{ME}[J] \wedge I_{16}[J+1]$

R-D[J]:=false;

1.16 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{K=J+1}^{N} [R\text{-}D_{ME}[K] \Rightarrow ((ME \in M\text{-}RD_K) \wedge A_{ME}[K]] \wedge I_{16}[J+1])$

Send REPLY(ME) to J;

1.17 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{K=J+1}^{N} [R\text{-}D_{ME}[K] \Rightarrow ((ME \in M\text{-}RD_K) \wedge A_{ME}[K]] \wedge (W_J \Rightarrow (ME \in M\text{-}RD_J \cup M\text{-}AUT_J \cup M\text{-}TRAIT_J)) \wedge I_{16}[J+1]$

fi

1.18 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{K=J+1}^{N} [R\text{-}D_{ME}[K] \Rightarrow ((ME \in M\text{-}RD_K) \wedge A_{ME}[K])] \wedge I_{16}[J+1]\}$

end-for;

1.19 : $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_1\}$

V.3.2.2.1.4.- Preuves séquentielles relatives aux parties traitant les messages

Soit $Sup^*_{ME} = Sup\{n \in \mathbb{N} / ((n,j) \in M_{ME,R}) \wedge (j \in [N] \setminus \{ME\})\}$

et $Sup^+_{ME} = Sup\{n' \in \mathbb{N} / ((n',j) \in M_{ME,R}) \wedge (j \in [N] \setminus \{ME\}) \wedge (n' \neq n) \wedge (j \neq ME)\}$.

2.2.1.0 : $\{(K=j) \wedge (j \neq ME) \wedge (j \in M\text{-}TRAIT_{ME}) \wedge (HSN=Sup^*_{ME}) \wedge (REPLY(j) \in M_{ME,R})\}$

$A_{ME}[K]$:=true;

2.2.1.1 : $\{(K=j) \wedge (REPLY(j) \in M_{ME,R}) \wedge (HSN=Sup^*_{ME}) \wedge (W_{ME} \Rightarrow (j \in M\text{-}AUT_{ME})\}$.

Ceci constitue la partie traitement des REPLY et nous donnons maintenant la partie concernant le traitement des REPLY-REQUEST.

2.2.2.0 : $\{(J=j) \wedge (TSN=n) \wedge (REPLY\text{-}REQUEST(n,j) \in M_{ME,R}) \wedge (HSN=Sup^+_{ME}) \wedge (j \in M\text{-}TRAIT_{ME}) \wedge (j \neq ME)\}$.

HSN:=Max(HSN,TSN);

A[J]:=true;

R-D[J]:=false;

2.2.2.1 : $\{(J=j) \wedge (TSN=n) \wedge (REPLY\text{-}REQUEST(n,j) \in M_{ME,R}) \wedge (HSN=Sup^*_{ME}) \wedge (W_{ME} \Rightarrow (J \in M\text{-}AUT_{ME}))\}$.

Enfin, il nous reste à donner la partie traitant les messages de REQUEST. Nous utilisons les instructions await afin d'exprimer l'atomicité choisie et le fait d'avoir true revient à exiger que l'instruction à l'intérieur soit exécutée atomiquement.

2.2.3.0 : {(J=j)∧(TSN=n)∧(REQUEST(n,j)$\in M_{ME,R}$)∧
(HSN=Sup^{+}_{ME})∧(ME∈M-$TRAIT_J$)}

HSN:=Max(HSN,TSN);
OUR-PRIORITY:=[(OSN<TSN)OR((TSN=OSN)and(ME<J)];
if USING ØR(WAITING AND OUR-PRIORITY) then
R-D[J]:=true;
fi;
if[~ USING and~WAITING]OR[WAITING and~A[J]and OUR-PRIORITY]then

A[J]:=false;
Send REPLY(ME) to J;
fi;
if WAITING and A[J] and ~ OUR-PRIORITY then
A[J]:=false;
Send REPLY-REQUEST(OSN ,ME)to J ;
fi

2.2.3.1. : {(J=j)∧(TSN=n)∧(REQUEST(n,j)$\in M_{ME,R}$)∧(HSN=Sup^{*}_{ME})∧
(W_J⇒(ME∈M-RD_J)∨(ME∈M-RP_J)∨(ME∈M-AUT_J)∨(ME∈M-$TRAIT_J$)∨(ME∈M-RR_J))}.

Ceci constitue la preuve séquentielle de RA et il reste maintenant à prouver que les autres sites et processus n'interfèrent pas avec cette preuve séquentielle : ce sera le propos du paragraphe suivant.

V.3.2.2.2.- Preuves d'absence d'interférence

Nous allons prouver deux types d'absence d'interférence :

- une interférence locale au noeud ME.
- une interférence globale au réseau RA pour ME.

Ceci revient à considérer deux types possibles d'interférence :

- interférence due au partage d'une variable
- interférence due à une communication.

V.3.2.2.2.1.- Interférence locale à ME

Il s'agit du même type de preuve que celui d'OWICKI et GRIES [OWG], plus des axiomes sur les communications.

CP(ME) est formé de deux processus parallèles notés $\mathbb{P}_1$ et $\mathbb{P}_2$ tels que :

$\mathbb{P}_1$ demande et rend la section critique.

$\mathbb{P}_2$ traite les messages dans le tampon.

$\mathbb{P}_2$ est formé de quatre instructions atomiques :

- $St_2 1$ ≡ Receive mes when $\bigvee_{\substack{J=1 \\ J\neq ME}}^{N}$ [(mes=REPLY(J))∨(mes=REPLY-REQUEST(n,J)) ∨(mes=REQUEST(n,J)]
- $St_2 2$ ≡ REPLY-MESSAGE;
- $St_2 3$ ≡ REPLY-REQUEST-MESSAGE;
- $St_2 4$ ≡ REQUEST-MESSAGE;

Afin de prouver que $\mathbb{P}_2$ n'interfère pas dans la preuve de $\mathbb{P}_1$, on prouve les relations suivantes :

① $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_{10} \wedge pre(St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge \sim I_{10}\}, k \in \{1,2,3,4\}$

② $\{ W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge \sim I_{11} \wedge pre(St_2k)\}$

St_2k

$\{ W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_{11}\}$

③ $\{ W_{ME} \wedge \sim U_{ME} \wedge I_{12}[J\text{-}1] \wedge pre(St_2k)\}$

St_2k

$\{ W_{ME} \wedge \sim U_{ME} \wedge I_{12}[J\text{-}1]\}$

④ $\{ W_{ME} \wedge \sim U_{ME} \wedge \sim A_{ME}[J] \wedge I_{12}[J\text{-}1] \wedge pre(St_2k)\}$

St_2k

$\{ W_{ME} \wedge \sim U_{ME} \wedge \sim A_{ME}[J] \wedge I_{12}[J\text{-}1]\}$

⑤ $\{ W_{ME} \wedge \sim U_{ME} \wedge I_{12}[J] \wedge pre(St_2k)\}$

St_2k

$\{ W_{ME} \wedge \sim U_{ME} \wedge I_{12}[J]\}$

⑥ $\{ W_{ME} \wedge \sim U_{ME} \wedge I_{13} \wedge pre(St_2k)\}$

St_2k

$\{ W_{ME} \wedge \sim U_{ME} \wedge I_{13}\}$

⑦ $\{ W_{ME} \wedge \sim U_{ME} \wedge I_{14} \wedge pre\ (St_2k)\}$

St_2k

$\{ W_{ME} \wedge \sim U_{ME} \wedge I_{14}\}$

⑧ $\{\sim W_{ME} \wedge \sim U_{ME} \wedge I_{14} \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge I_{14}\}$

⑨ $\{\sim W_{ME} \wedge \sim U_{ME} \wedge I_{14} \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge I_{14}\}$

⑩ $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{J=1}^{N} (R\text{-}D_{ME}[J] \Rightarrow \big((ME \in M\text{-}RD_J) \wedge A_{ME}[J]\big)) \wedge I_{15} \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{J=1}^{N} (R\text{-}D_{ME}[J] \Rightarrow \big((ME \in M\text{-}RD_J) \wedge A_{ME}[J]\big)) \wedge I_{15}\}$

⑪ $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J}^{N} (R\text{-}D_{ME}[k] \Rightarrow \big((ME \in M\text{-}RD_k) \wedge A_{ME}[k]\big) \wedge I_{16}[J]) \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J}^{N} \big((R\text{-}D_{ME}[k] \Rightarrow (ME \in M\text{-}RD_k) \wedge A_{ME}[k]\big) \wedge I_{16}[J])\}$

⑫ $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow ((ME \in M\text{-}RD_k) \wedge A_{ME}[k]) \wedge A_{ME}[J]$

$\wedge (ME \in M\text{-}RD_J) \wedge R\text{-}D[J] \wedge I_{16}[J+1] \wedge pre\ (St_2k)\}$

St_2k

(12) $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow (ME \in M\text{-}RD_k) \wedge A_{ME}[k]$

$A_{ME}[J] \wedge (ME \in M\text{-}RD_J) \wedge R\text{-}D[J] \wedge I_{16}[J+1]\}$

(13) $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow ((ME \in M\text{-}RD_k) \wedge A_{ME}[k])) \wedge I_{16}[J+1] \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow (ME \in M\text{-}RD_k) \wedge A_{ME}[k]) \wedge I_{16}[J+1]\}$

(14) $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow (ME \in M\text{-}RD_k) \wedge A_{ME}[k]) \wedge$

$[W'_J \Rightarrow ME \in M\text{-}RP_J \cup M\text{-}AUT_J \cup M\text{-}TRAIT_J)] \wedge I_{16}[J+1] \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow (ME \in M\text{-}RD_k) \wedge A_{ME}[k]) \wedge$

$[W'_J \Rightarrow (ME \in M\text{-}AUT_J \cup M\text{-}RD_J \cup M\text{-}TRAIT_J)] \wedge I_{16}[J+1]\}$

(15) $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow (ME \in M\text{-}RD_k) \wedge A_{ME}[k]) \wedge I_{16}[J+1] \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge \sim U_{ME} \wedge \bigwedge_{k=J+1}^{N} (R\text{-}D_{ME}[k] \Rightarrow (ME \in M\text{-}RD_k) \wedge A_{ME}[k]) \wedge I_{16}[J+1]\}$

(16) $\{\sim W_{ME} \wedge \sim U_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_1 \wedge pre\ (St_2k)\}$

St_2k

$\{\sim W_{ME} \wedge U_{ME} \wedge \sim R\text{-}D_{ME} \wedge I_1\}$

Nous prouvons maintenant que $\mathbb{P}_1$ n'interfère pas avec la preuve de $\mathbb{P}_2$. Les instructions atomiques de $\mathbb{P}_1$ sont notées St_1k et définies comme suit :

$St_1 1 \equiv \left[\begin{array}{l} WAITING:=true; \\ OSN:=HSN+1 \end{array}\right]$

$St_1 2 \equiv$ Send Request (OSN,ME) to J

$St_1 3 \equiv$ Wait-for$\left(\bigwedge_{J=1}^{N} A_{ME}[J]\right)$

$St_1 4 \equiv$ WAITING:=false

$St_1 5 \equiv$ USING:=True

$St_1 6 \equiv$ USING:=false

$St_1 7 \equiv$ A[J]:=false

$St_1 8 \equiv$ R-D[J]:=false

$St_1 9 \equiv$ Send Reply (ME) to J

Soit $k \in \{1,\ldots,9\}$,

(17) $\{(k=j) \wedge (Reply(j) \in M_{ME,R}) \wedge (j \neq ME) \wedge (HSN=Sup^*_{ME}) \wedge (j \in M\text{-}TRAIT_{ME}) \wedge pre\ (St_1k)\}$

St_1k

$\{(k=j) \wedge (Reply(j) \in M_{ME,R}) \wedge (j=ME) \wedge (HSN=Sup^*_{ME}) \wedge (j \in M\text{-}TRAIT_{ME})\}$

(18) $\{(k=j) \wedge (Reply(j) \in M_{ME,R}) \wedge (j \neq ME) \wedge (HSN=Sup^*_{ME}) \wedge (W_{ME} \Rightarrow j \in M\text{-}AUT_{ME}) \wedge pre\ (St_1k)\}$

St_1k

$\{(k=j) \wedge (Reply(j) \in M_{ME,R}) \wedge (j \neq ME) \wedge (HSN=Sup^*_{ME}) \wedge (W_{ME} \Rightarrow j \in M\text{-}AUT_{ME})\}$

(19) $\{(J=j)\wedge(j\neq ME)\wedge(TSN=n)\wedge(\text{Reply-Request}(n,j)\in M_{ME,R})$

$\wedge(HSN=Sup^{*}_{ME})\wedge(j\in M\text{-}TRAIT_{ME})\wedge pre\ (S_1k)\}$

St_1k

$\{(J=j)\wedge(j\neq ME)\wedge(TSN=n)\wedge(\text{Reply-Request}(n,j)\in M_{ME,R})\wedge$

$(HSN=Sup^{+}_{ME})\wedge(j\in M\text{-}TRAIT_{ME})\}$

(20) $\{(J=j)\wedge(j\neq ME)\wedge(\text{Reply-Request}(n,j)\in M_{ME,R})\wedge$

$(TSN=n)\wedge(HSN = Sup^{*}_{ME})\wedge(W_{ME}\Rightarrow j\in M\text{-}AUT_{ME})\wedge pre\ (St_1k)\}$

St_1k

$\{(J=j)\wedge(j\neq ME)\wedge(\text{Reply-Request}(n,j)\in M_{ME,R})\wedge$

$(TSN=n)\wedge(HSN=Sup^{*}_{ME})\wedge(W_{ME}\Rightarrow j\in M\text{-}AUT_{ME})\}$

(21) $\{(J=j)\wedge(TSN=n)\wedge(\text{Request}(n,j)\in M_{ME,R})\wedge(HSN=Sup^{+}_{ME})\wedge(ME\in M\text{-}TRAIT_j)\wedge pre\ (St_1k)\}$

St_1k

$\{(J=j)\wedge(TSN=n)\wedge(\text{Request}(n,j)\in M_{ME,R})\wedge(HSN=Sup^{+}_{ME})\wedge(ME\in M\text{-}TRAIT_j)\}$

(22) $\{(J=j)\wedge(TSN=n)\wedge(\text{Request}(n,j)\in M_{ME,R})\wedge(HSN=Sup^{*}_{ME})$

$\wedge(W_j\Rightarrow(ME\in M\text{-}RD_j\cup M\text{-}RP_j\cup M\text{-}AUT_j\cup M\text{-}TRAIT_j\cup M\text{-}RR_j)\wedge pre\ (St_1k)\}$

St_1k

$\{(J=j)\wedge(TSN=n)\wedge(\text{Request}(n,j)\in M_{ME,R})\wedge(HSN=Sup^{*}_{ME})$

$\wedge(W_j\Rightarrow(ME\in M\text{-}RD_j\cup M\text{-}RD_j\cup M\text{-}AUT_j\cup M\text{-}TRAIT_j\cup M\text{-}RR_j))\}$

(23) $\{(HSN_{ME}=Sup^{*}_{ME})\wedge I_3\wedge pre\ (St_1k)\}$

St_1k

$\{(HSN_{ME}=Sup^{*}_{ME})\wedge I_3\}$

(24) $\{(HSN_{ME}=Sup^{*}_{ME})\wedge I_4\wedge pre\ (St_1k)\}$

St_1k

$\{(HSN_{ME}=Sup^{*}_{ME})\wedge I_4\}$

(25) $\{(HSN_{ME}=Sup^{*}_{ME})\wedge pre\ (St_1k)\}$

St_1k

$\{(HSN_{ME}=Sup^{*}_{ME})\}$

(26) $\{(HSN_{ME}=0)\wedge\sim A_{ME}\wedge\sim R\text{-}D_{ME}\wedge\sim U_{ME}\wedge I_2\wedge pre\ (St_1k)\}$

St_1k

$\{(HSN_{ME}=0)\wedge\sim A_{ME}\wedge\sim R\text{-}D_{ME}\wedge\sim U_{ME}\wedge I_2\}$

(27) $\{(OSN_{ME}=0)\wedge\sim W_{ME}\wedge\sim U_{ME}\wedge\sim A_{ME}\wedge\sim R\text{-}D_{ME}\wedge I_1\wedge pre\ (St_2k)\}$

St_2k

$\{(OSN_{ME}=0)\wedge\sim W_{ME}\wedge\sim U_{ME}\wedge\sim A_{ME}\wedge\sim R\text{-}D_{ME}\wedge I_1\}$

V.3.2.2.2.2.- Interférence globale de RA

Soient ME, ME' deux noeuds distincts de RA. On doit montrer que l'envoi d'un message de ME' n'interfère pas avec toute assertion de ME et la réception d'un message de ME' n'interfère pas avec toute assertion de ME.

Nous utilisons les notations suivantes :

$St_3 1 \equiv$ Send Request(OSN,ME') to ME

$St_3 2 \equiv$ Send Reply(ME') to ME

$St_3 3 \equiv$ Send Reply(ME') to ME

$St_3 4 \equiv$ Send Reply-Request($OSN_{ME'}$,ME') to ME

① $\{(HSN_{ME}=Sup^{*}_{ME}) \wedge pre\ (St_3 k)\}$

$St_3 k$

$\{(HSN_{ME}=Sup^{*}_{ME})\}$

② $\{(HSN_{ME}=Sup^{*}_{ME}) \wedge pre(St_3 k)\}$

$St_3 k$

$\{(HSN_{ME}=Sup^{*}_{ME})\}$

On prouve la même chose de ME vers ME', en utilisant ME à ME' et ME' à ME dans ① et ② .

V.3.2.2.3.- Exclusion mutuelle

L'assertion d'exclusion mutuelle peut s'exprimer de la façon suivante :

$$MUTEX' \equiv \left[\exists ?\, J \in [N], \left[\bigwedge_{K=1}^{N} A_J[K]\right] \wedge \left[W_J \vee U_J \vee (\sim W_J \wedge \sim U_J)\right]\right]$$

$\exists ?$ signifie "il existe au plus un" "tel que".

On vérifie aisément mais longuement que chaque assertion de l'annotation implique MUTEX'. Donc MUTEX' est invariant pour RA.

V.4.- PREUVE DE PROPRIETES DE FATALITE DE PROGRAMMES DISTRIBUES

V.4.1.- Présentation

Dans la partie V.3, nous avons donné une méthode de preuve de propriétés d'invariance que nous avons ensuite utilisée pour le cas de l'algorithme de RICART et AGRAWALA. Cependant, les propriétés d'invariance décrivent partiellement l'exécution et le comportement du programme et nous porterons notre attention sur les propriétés de fatalité qui ont été définies pour DIP $\in \mathfrak{D}$ dans la partie V.2. Une propriété intéressante au sujet de l'algorithme RA est l'accessibilité en section critique pour tout noeud de RA qui la demande. Cette propriété peut s'exprimer en utilisant la terminologie de la fatalité :

Soit $\mathfrak{A}$[RA] un langage d'assertions pour RA,

$\mathcal{M}$ un modèle de fatalité pour RA.

ME un noeud de RA ; on exprime cette propriété par :

$$\mathcal{M} \models WAITING_{ME} \rightsquigarrow USING_{ME}$$

où $WAITING_{ME}$ signifie que WAITING est vrai pour ME,

$USING_{ME}$ signifie que USING est vrai pour ME.

Cette propriété est exprimée dans un modèle de fatalité $\mathcal{M}$ que nous voulons faiblement équitable.

Nous utiliserons dans notre étude l'opérateur temporel $\Box$ dont la signification est la suivante :

$\mathcal{M} \models \Box P$ ssi $\forall e \in S[D\mathbb{P}], P(e) \Rightarrow [\forall e' \in S[D\mathbb{P}], t^*[D\mathbb{P}](e,e') \Rightarrow P(e')]$

Nous supposons que tout message sur le point d'être envoyé, le sera au bout d'un temps fini, que tout message envoyé est délivré et que tout message en attente de réception pendant un temps arbitrairement grand est reçu au bout d'un temps fini. Nous allons maintenant donner une axiomatique adaptée à ce genre de programmes sous ces hypothèses.

V.4.2.- Système de preuve de fatalité TD

Comme OL, TD est composé de trois parties dont nous présentons succinctement le contenu :

- une première partie contenant les axiomes propres à exprimer l'hypothèse d'équité faible et de fiabilité.
- une seconde partie pareille à celle de OL contenant les règles de déduction qui permettent de construire les preuves et, donc, la relation $\rightsquigarrow$.
- enfin, une troisième partie disposant des mêmes règles auxiliaires que OL.

Partie 1 : Equité faible et fiabilité

D1 : $\underline{at}[v:=e] \rightsquigarrow \underline{jafter}[v:=e]$ où $v \in \mathcal{V}[D\mathbb{P}]$, $e \in \mathcal{E}$.

D2 : $\underline{at}[v:=?] \rightsquigarrow \underline{jafter}[v:=?]$ où $v \in \mathcal{V}[D\mathbb{P}]$.

D3 : $\underline{at}[\underline{skip}] \rightsquigarrow \underline{jafter}[\underline{skip}]$

D4 : $\underline{at}[\underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}] \rightsquigarrow (\underline{jat}\ S_1 \wedge B \vee \underline{jat}\ S_2 \wedge \sim B)$
où $B \in \mathcal{B}$, $S_1, S_2 \in \mathcal{P}$.

D5 : $\underline{after}\ S_1 \rightsquigarrow \underline{jafter}[\underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}]^{(1)}$

D6 : $\underline{after}\ S_2 \rightsquigarrow \underline{jafter}[\underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}]^{(2)}$

D7 : $\underline{at}[\underline{while}\ B\ \underline{do}\ S\ \underline{od}] \rightsquigarrow (\underline{jat}\ S \wedge B) \vee (\underline{jafter}[\underline{while}\ B\ \underline{do}\ S\ \underline{od}])$

D8 : $\underline{after}\ S \rightsquigarrow (\underline{jat}\ S \wedge B) \vee (\underline{jafter}[\underline{while}\ B\ \underline{do}\ S\ \underline{od}] \wedge \sim B)$.

D9 : $\underline{at}[\underline{Send}\ mes\ \underline{to}\ dest] \rightsquigarrow \underline{jafter}[\underline{Send}\ldots dest] \wedge (mes \in T_{dest})$

D10 : $(\underline{at}[\underline{Receive}\ mes\ \underline{when}\ cond] \wedge \Box\Diamond[(mes \in T) \wedge cond[mes]]) \rightsquigarrow \ldots$
$\rightsquigarrow (\underline{jafter}[\underline{Receive}\ mes\ \underline{when}\ cond] \wedge cond)$.

D11 : $\underline{at}[\underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend}] \rightsquigarrow (\underline{jat}\ \mathbb{P}S_1 \wedge \ldots \wedge \underline{jat}\ \mathbb{P}S_n)$.

D12 : $(\underline{after}\ \mathbb{P}S_1 \wedge \ldots \wedge \underline{after}\ \mathbb{P}S_n) \rightsquigarrow \underline{jafter}[\underline{Cobegin}\ \mathbb{P}S_1 \| \ldots \| \mathbb{P}S_n\ \underline{coend}]$.

D13 : $\underline{at}[\underline{Disbegin}\ C\mathbb{P}_1 \| \ldots \| C\mathbb{P}_m\ \underline{disend}] \rightsquigarrow (\underline{jat}\ C\mathbb{P}_1 \wedge \ldots \wedge \underline{jat}\ C\mathbb{P}_m)$.

D14 : $(\underline{after}\ C\mathbb{P}_1 \wedge \ldots \wedge \underline{after}\ C\mathbb{P}_{i-1} \wedge \underline{jafter}\ C\mathbb{P}_i \wedge \ldots \wedge \underline{after}\ C\mathbb{P}_m) \rightsquigarrow \ldots$
$\rightsquigarrow \underline{jafter}[\underline{Disbegin}\ C\mathbb{P}_1 \| \ldots \| C\mathbb{P}_m\ \underline{disend}]$.

Partie 2 : Règles d'inférences : P, Q, R, S, $R(\alpha)$, $R(\beta)$ sont des assertions de $\mathcal{A}[D\mathbb{P}]$.

R1 : si $_{ID} \vdash P \rightarrow q$, alors $_{TD} \vdash P \rightsquigarrow Q$.

R2 : si $_{TD}\vdash P \rightsquigarrow Q$,

$$_{TD}\vdash \left[\begin{array}{l} \exists i \in \mathcal{A}[D\!\!\!P], P\wedge R \Rightarrow i, \\ \qquad\qquad i \Rightarrow Oi, \\ \qquad\qquad Q\wedge i \Rightarrow S \end{array}\right]$$

alors

$_{TD}\vdash P\wedge R \rightsquigarrow Q\wedge S$.

R3 : si $_{TD}\vdash P \rightsquigarrow \exists\alpha R(\alpha)$,

$\forall\alpha \in Ord,\ \alpha > 0$,

$_{TD}\vdash R(\alpha) \rightsquigarrow (\exists\beta < \alpha,\ R(\beta))$,

$_{TD}\vdash R(0) \rightsquigarrow Q$,

alors $_{TD}\vdash P \rightsquigarrow Q$.

R4 : si $_{TD}\vdash P \rightsquigarrow Q$, $_{TD}\vdash Q \Rightarrow R$,

alors $_{TD}\vdash P \rightsquigarrow R$.

Partie 3 : Règles auxiliaires

R1A : si $P \Rightarrow Q$ est vrai pour $D\!\!\!P$ alors $_{TD}\vdash P \Rightarrow Q$.

R2A : si $\forall(s,s')\in S^2[D\!\!\!P],\ (P\wedge R)(s)\wedge t^*[D\!\!\!P](s,s')\wedge Q(s') \Rightarrow S(s')$,

alors

$$_{TD}\vdash \left[\begin{array}{l} \exists I \in [D\!\!\!P],\ P\wedge R \Rightarrow I \\ \qquad\qquad I \Rightarrow OI \\ \qquad\qquad Q\wedge I \Rightarrow S \end{array}\right]$$

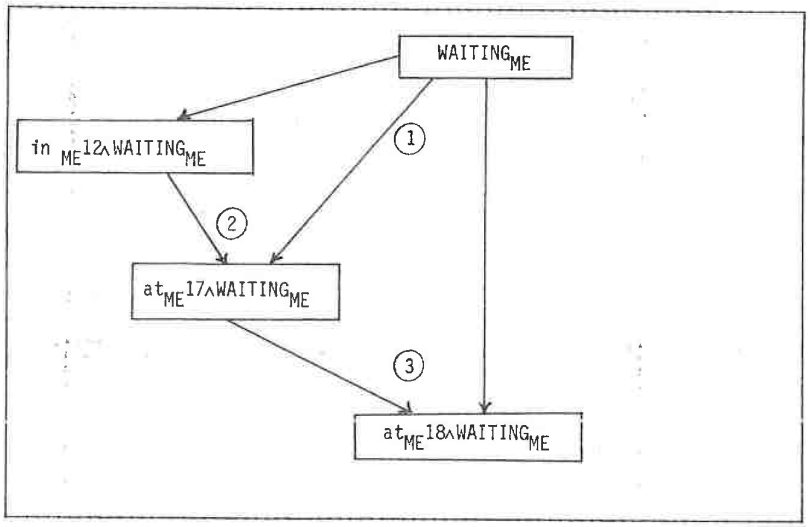
R3A : si $P \rightsquigarrow Q$ est une hypothèse de travail, alors

$_{TD}\vdash P \rightsquigarrow Q$.

Les axiomes D1 à D14 expriment les propriétés des suites ou traces d'états considérées. Nous appellerons $\mathcal{M}_{WF}$ le modèle de fatalité validant l'ensemble de ces axiomes et nous ne considèrerons que le plus grand modèle vérifiant cette propriété. Notre propos ici n'est pas de prouver ni la correction, ni la complétude sémantique de TD mais de donner une généralisation de la méthode d'OWICKI-LAMPORT. Il nous paraît raisonnable de généraliser la méthode des treillis de preuve à TD de façon à l'utiliser par la suite pour prouver la propriété que nous avions donnée au paragraphe V.3.1. La méthode reste la même, si ce n'est qu'il faut remplacer OL par TD. Nous donnons maintenant les treillis de preuve correspondant à la preuve de l'accessibilité de ME en section critique :

$M \in \{1,\ldots,N\}$

$WAITING_{ME} \rightsquigarrow USING_{ME}$.



Treillis de preuve : TP1

Commentaires sur TP1

$(in_{ME}12 \equiv at_{ME}12 \vee at_{ME}13 \vee at_{ME}14 \vee at_{ME}15 \vee at_{ME}16)$.

(1) Nous avons prouvé précédemment que RA réalise l'exclusion mutuelle.

Ceci a été mené par l'intermédiaire d'un invariant $I \equiv \bigwedge_{\alpha \in [RA]} (at\ \alpha \Rightarrow I_\alpha)$

où [RA] est l'ensemble des points de contrôle de RA.

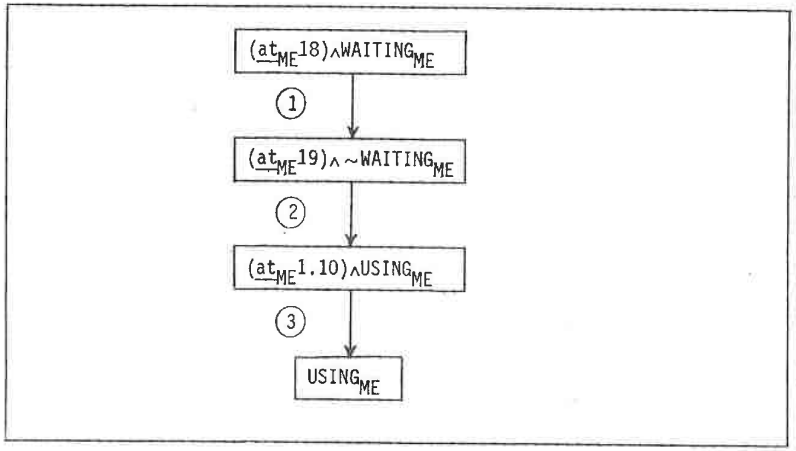
WAITING est vraie en 3 points, on en déduit :

$WAITING_{ME} \Rightarrow (in_{ME}12 \vee at_{ME}17 \vee at_{ME}18) \wedge WAITING_{ME}$.

Par les règles 1A, puis 1 on en déduit (1) .

(2) On verra TP1.1

(3) On verra TP1.2.



Treillis de preuve TP2

Commentaires sur TP2

(1) $I_1 \equiv (at_{ME}18 \Rightarrow WAITING_{ME}) \wedge (at_{ME}19 \Rightarrow \sim WAITING_{ME})$

est invariant par RA (on le déduit de l'invariant plus fort de la

preuve d'exclusion mutuelle).

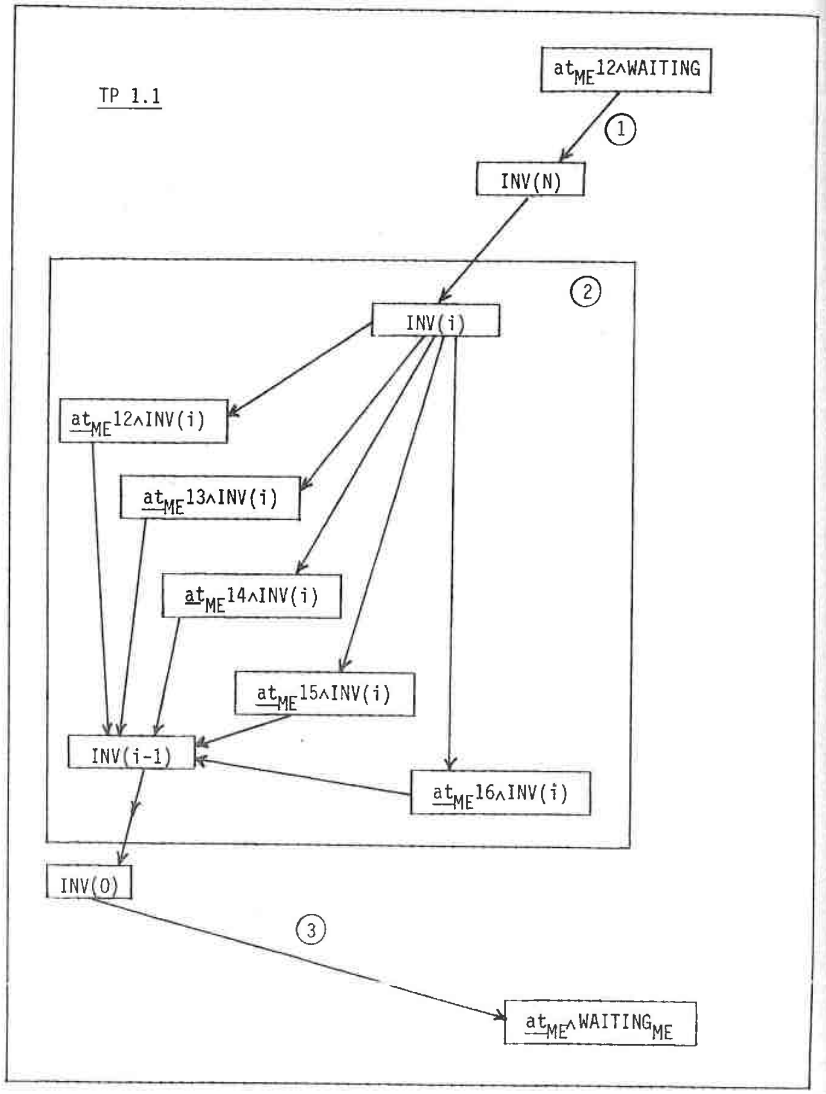
$at_{ME}18 \rightsquigarrow jafter_{ME}18$ est un axiome de TD.

De plus, $jafter_{ME}18 \rightsquigarrow after_{ME}18$ est trivial par définition de jafter et after.

On utilise la règle 2 pour déduire alors (1) .

(2) On fait le même raisonnement que ci-dessus et on considère l'invariant $I_2 \equiv (at_{ME}19 \Rightarrow \sim WAITING_{ME}) \wedge (at_{ME}1.10 \Rightarrow USING_{ME})$.

(3) Puisque $(at_{ME}1.10) \wedge USING_{ME} \Rightarrow USING_{ME}$, on en déduit par la règle 1A puis 1, (3) .



Commentaires sur TP1.1

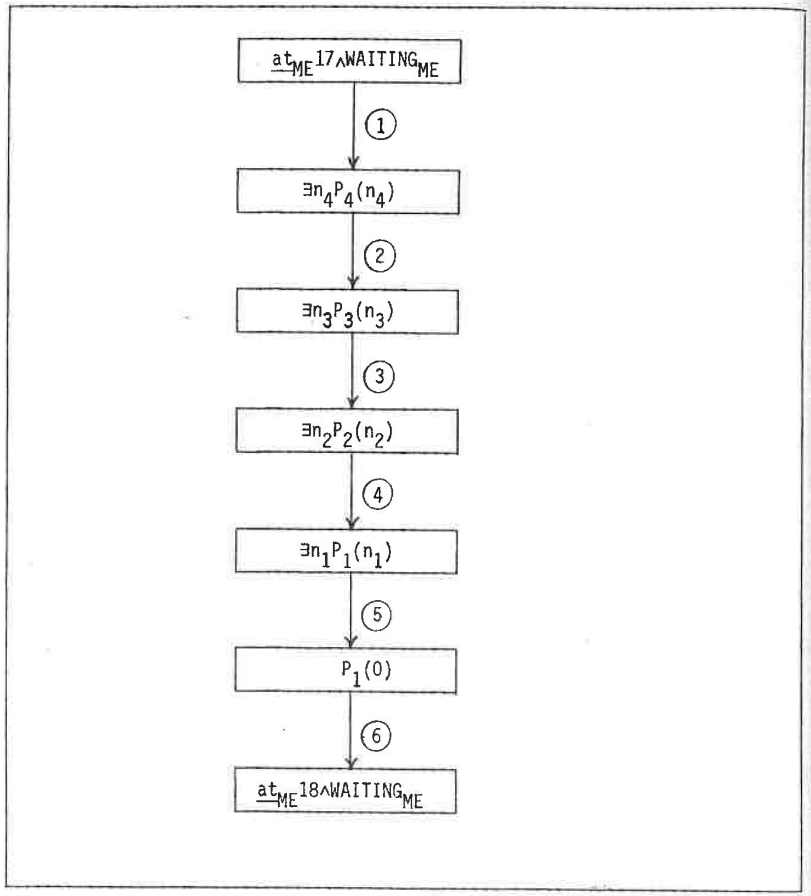
$$Inv(m) \equiv [\underline{at}_{ME}13 \wedge (J=N-m) \wedge (J \leqslant N) \wedge I_{12}(J-1)] \vee$$
$$[\underline{at}_{ME}14 \wedge (J=N-m) \wedge (J \leqslant N) \wedge \sim A[J] \wedge (J \neq ME) \wedge I_{12}(J-1)] \vee$$
$$[\underline{at}_{ME}15 \wedge (J=N-m) \wedge (J \leqslant N) \wedge (J<N) \wedge I_{12}(J)] \vee$$
$$[\underline{at}_{ME}16 \wedge (J=N=m) \wedge (J \leqslant N) \wedge I_{12}(J)] \vee$$
$$[\underline{at}_{ME}12 \wedge (J=0) \wedge (m=N) \wedge I_{11}]$$

$$INV(u) = \bigvee_{m \leqslant u} Inv(m).$$

(1) Lorsque le contrôle est $in_{ME}12$, les états de RA sont caractérisés par I_{11} et $I_{12}[J]$, soit par INV(N) de façon grossière.
L'implication suivante est donc évidente :
$\underline{in}_{ME}12 \wedge WAITING_{ME} \Rightarrow INV(N)$.
D'où par les règles 1Z et 1, on déduit (1).

(2) On utilise les axiomes d'équité pour chaque point en transformant le for en while. Ainsi l'entier i sera décrémenté.

(3) Il suffit pour cela d'utiliser l'axiome du while pour ce cas.



TP 1.2

Commentaires au sujet de TP1.2

On utilise pour cette preuve les notations suivantes :

$|M\text{-}RQ|$ est le cardinal de M-RQ,

$|M\text{-}RP|$ est le cardinal de M-RP,

$|M\text{-}AUT|$ est le cardinal de M-AUT,

$|M\text{-}RER|$ est le cardinal de M-RER,

M-RER est l'ensemble qui contient les numéros de tout noeud distinct de ME tel qu'un message REPLY-REQUEST a été envoyé par ME à ces noeuds.

$|M\text{-}RED|$ est le cardinal de M-RED,

M-RED est l'ensemble suivant :

$j \in M\text{-}RED$ ssi $(j \neq ME) \wedge (R\text{-}D_j[ME] = true)$.

Nous utilisons les variables d'entiers suivantes :

n_0, n_1, n_2, n_3, n_4 et nous définissons alors l'assertion suivante :

$$P(n_4,n_3,n_2,n_1,n_0) \equiv \Big[W_{ME} \wedge \underline{at}_{ME}17 \wedge (|M\text{-}RQ| = n_4) \wedge (|M\text{-}REP| = n_1) \wedge (n_0 = N-1-|M\text{-}AUT_{ME}|) \wedge (|M\text{-}RED| = n_2) \wedge (|M\text{-}RER| = n_3) \wedge (M\text{-}RQ \cup M\text{-}REP \cup M\text{-}AUT \cup M\text{-}RED \cup M\text{-}RER = [N] \setminus \{ME\}) \wedge (n_4+n_3+n_2+n_1+n_0' = N-1) \wedge (n_0' = |M\text{-}AUT_{ME}|) \wedge (n_0 = N-1-n_0') \Big]$$

On définit alors une suite de 5 types d'assertions : pour $i \in \{0,\ldots,4\}$,

$$P_i(n_i) \equiv \bigvee_{(n_4,n_3,n_2,n_1,n_0) \in [(i)]} P(n_4,n_3,n_2,n_1,n_0)$$

$$[(i)] = \{(n_4,n_3,n_2,n_1,n_0) \in \mathbb{N}^5 / (\forall j>i, n_j=0) \wedge (\forall j \leq i, n_j \leq N-1)\}$$

Ce qui permet de donner un ordre lexicographique pour la preuve et ceci constitue une notion de variable auxiliaire qui explicite les ordinaux néces-

saires. Dans le cas présent, les ordinaux sont :

$$\alpha = \sum_{i=0}^{4} n_i.\omega^i.$$

(1) La justification de cette fatalité est évidente à cause du choix des différentes variables auxiliaires et de la définition de $P_i(n_i)$. Ainsi, quoiqu'il advienne, au pire des cas, si on se trouve en 17, il reste N-1 réponses en attente de sollicitation ie des requêtes.

D'où on en déduit :

$$\underline{at}_{ME}17 \wedge W_{ME} \Rightarrow \bigvee_{n_4 \leq N-1} P_4(n_4).$$

Par les règles 1A et 1, on déduit (1).

(2) Il faut utiliser la règle 3 et prouver que le nombre de requêtes en attente décroît strictement. Pour cela, soit $j \in M\text{-}RQ_{ME}$ telle que $j \neq ME$.

On utilise l'axiome d'équité relatif à la réception du message :

$$(\underline{at}_j[\underline{Receive}]) \wedge \Box \Diamond[(mes \in T_j) \wedge (mes = REQUEST(OSN_{ME}, ME))] \rightsquigarrow$$

$$\rightsquigarrow (\underline{jafter}[Receive]) \wedge (mes = REQUEST(OSN_{ME}, ME)) \wedge (mes \notin T_j).$$

Donc :

$$[(\underline{at}_j Receive) \wedge \Box \Diamond(\sim) \wedge P_4(n)] \rightsquigarrow [\underline{jafter}_j Receive \wedge P_4(n') \wedge n' < n]$$

par la règle 2, en utilisant le fait que $P_4(n') \wedge n' < n$ est invariant par rapport à $\underline{at}_j Receive \wedge \Box \Diamond(\sim) \wedge P_4(n)$ et $\underline{jafter}_j Receive$.

On procède de même pour chaque j et on prouve que n décroît. On en déduit que :

$$\exists n_4 P_4(n_4) \rightsquigarrow P_4(0) \quad \text{et} \quad P_4(0) \Rightarrow \exists n_3 P_3(0).$$

On utilise les règles 1A, 1 et la règle de transitivité pour déduire (2)

(3) (4) (5)

Ces preuves sont identiques à celles ci-dessus et utilisent notamment l'hypothèse suivante :

$\underline{in}$ C-S(j) $\rightsquigarrow$ $\underline{after}$ C-S(j) pour chaque j de [N].

Par la règle 3A, ceci devient une hypothèse.

(6) Enfin, la preuve de cette propriété repose sur l'hypothèse à ajouter comme règle supplémentaire :

$$\left(\underline{at}_{ME}17\right) \wedge \left(\bigwedge_{J=1}^{N} A_{ME}[J]\right) \rightsquigarrow \left(\underline{at}_{ME}18\right) \wedge W_{ME}.$$

Ceci est une hypothèse raisonnable puisque nous avons montré l'exclusion mutuelle auparavant.

V.5.- REMARQUES COMPLEMENTAIRES

V.5.1.- Absence de blocage

Puisque nous avons prouvé que tout noeud, qui demande sa section critique, l'obtient au bout d'un temps fini, il ne peut y avoir blocage global ou mutuel. Ainsi, le seul point de blocage est le point 17 et dans ce cas le noeud concerné demande sa section critique ; nous avons prouvé qu'il se trouvera un jour fatalement en 18.

V.5.2.- Complexité des preuves

Les preuves de fatalité comme celle d'invariance sont fort complexes et demandent des efforts considérables. Nous pensons qu'une assistance pseudo-mécanisée serait bien venue pour ce type de problèmes. Les difficultés sont

plus importantes car elles exigent un outil interactif pour le cas des propriétés d'invariance. La schématisation offerte par les treillis de preuve facilite considérablement la résolution du problème.

V.5.3.- Correction et complétude sémantique

Nous n'avons pas exposé le problème de la correction et de la complétude sémantique de TD mais nous pensons que c'est un problème trop complexe pour être exposé brièvement. Une étude future nous permettra éventuellement de compléter TD, afin d'assurer complétude et correction de TD.

Le problème de fiabilité a été facilement résolu et il reste de nombreux choix et cas à considérer. La logique temporelle a été utilisée pour spécifier de tels programmes notamment SCHWARTZ et MELLIAR-SMITH [SCM1] en font une démonstration.

VI - CONCLUSION

VI - CONCLUSION

Au terme de ce travail, il convient de faire le point sur les principaux résultats obtenus et d'esquisser les perspectives que nous envisageons pour nos travaux futurs.

La notion de structure abstraite pour des langages infinitaires nous a permis d'associer des ordinaux plus petits que l'ordinal de la structure abstraite à toute propriété de fatalité abstraite sans hypothèse d'équité faible, c'est-à-dire pour tout l'ensemble de traces du programme abstrait. Le problème qui nous paraît important est celui du choix du langage d'assertions, qui doit être suffisamment expressif, afin de garantir l'expression de toute propriété dont nous aurions besoin : les fonctions d'abstraction et de concrétisation utilisées antérieurement pour ce type de problème par COUSOT R. [COU] sont ici essentielles pour garantir un formalisme propre. La notion de modèle de fatalité permet de prendre en compte de nombreuses hypothèses et, comme OWICKI et LAMPORT [OWL], nous caractérisons cette propriété que nous appelons l'équité faible, en spécifiant, tout d'abord, la classe des programmes parallèles, la sémantique opérationnelle, les modèles de fatalité déduits de la sémantique opérationnelle et des traces, les assertions de contrôle, les actions atomiques ; notre démarche est indépendante de la logique temporelle qui n'offre pas toutes les hypothèses requises pour une base formelle à cette étude. Cependant, les résultats obtenus pour les propriétés de fatalité sans hypothèse ne purent être généralisés systématiquement à celles avec hypothèse d'équité faible. Nous avons mis en évidence un opérateur monotone Γ_Q pour chaque assertion Q de CIP, un programme parallèle quelconque et nous avons donné, à l'aide de Γ_Q, une suite d'assertions intermédiaires pour prouver que P conduit à Q fatalement

sous hypothèse d'équité faible. Des actions atomiques semblaient jouer un rôle essentiel pour conduire à Q sous cette condition et nous avons défini l'ensemble des actions critiques en un état donné par un multiensemble ; cette approche suggère de chercher quelques actions déterminantes au cours d'une preuve de fatalité sous hypothèse d'équité faible. Ainsi on mis en évidence l'existence de l'ordinal de fatalité sous hypothèse d'équité faible pour deux assertions P et Q telles que $\mathcal{M}_{WF} \models P \rightsquigarrow Q$, en supposant la relation de transition dénombrable, on prouve que l'ordinal de fatalité est au plus dénombrable (ie : $<\omega_1$) et dans le cas général, l'ordinal était plus petit que celui de la structure abstraite.

L'étude préalable nous a permis de prouver la correction et la complétude sémantique du système de preuves de fatalité de programmes parallèles (appartenant à la classe définie) sous hypothèse d'équité faible. La correction est faite par le biais du modèle de fatalité faiblement équitable défini antérieurement et la notion de validité est spécifiée. La preuve de complétude sémantique est déduite de l'étude préalable et le choix du langage d'assertions est très important quant à l'existence des assertions nécessaires pour la preuve. Ces résultats ont été obtenus par un enrichissement du système axiomatique d'OWICKI et LAMPORT [OWL] mais nous y avons ajouté une règle d'induction et un système de preuve d'invariance correct et sémantiquement complet, ce qui n'avait pas été fait par OWICKI et LAMPORT [OWL].

Cette preuve est constructive dans la mesure où elle propose une suite d'assertions intermédiaires pour la règle d'induction mais ces assertions sont inutilisables en pratique et il n'y a pas, comme APT et DELPORTE [APD] le proposent, de véritable heuristique associée à la preuve. Le point original de la méthode d'OWICKI et LAMPORT [OWL] est la notion de treillis de preuve et nous

avons trivialement donné la complétude de cette méthode. La correction de cette méthode avait été faite par OWICKI et LAMPORT [OWL] pour leur système et la généralisation fut aisée. Cependant l'illustration de cette méthode par la preuve d'une propriété de fatalité du ramasse-miettes de BEN ARI [BEN2] montre l'élégance de cette technique de preuve due particulièrement à l'aspect schématique des preuves. Cette technique permet de simplifier les preuves de fatalité de programmes très complexes et le ramasse-miettes n'en est pas un des moindres, d'autant plus que nous y avons trouvé une erreur.

Au lieu d'implanter des variables auxiliaires, afin de transformer un programme parallèle sous hypothèse d'équité faible en un programme parallèle faiblement équitable, comme APT et OLDEROG [APO], nous implantons, au niveau de la sémantique, des variables auxiliaires, sans les mentionner dans la syntaxe du programme, ayant valeur dans des structures bien fondées et telles que le modèle de fatalité soit faiblement équitable. Notre approche permet de mettre en évidence un fait rencontré en pratique : au cours d'une preuve de fatalité, les ordinaux sont toujours polynomiaux ($<\omega^\omega$) et ceci est dû au choix de la structure bien-fondée, domaine des variables auxiliaires d'implantation.

Nous n'avons fait qu'esquisser une étude formelle des treillis de preuve dont le nom formel est diagramme ordinal et, sur ce point, nous n'avons donné que quelques résultats symboliques intuitifs. Il nous reste à poursuivre afin d'étendre la structure mathématique de ces objets. L'algorithme de RICART et AGRAWALA [RIA] amélioré par CARVAHLO et ROUCAIROL [CAR] nous a conduit à généraliser la méthode de preuves d'invariance d'OWICKI et GRIES [OWG] et la méthode des treillis de preuve précédemment étudiée, à une classe de programmes distribués communiquant par message de façon asynchone dans un tampon.

Les propriétés prouvées nous ont montré que la méthode des treillis de

preuve facilitait considérablement la vérification et la compréhension de tels programmes complexes ; elle permet d'introduire la notion de modularité dans les preuves, ce qui est suggéré par OWICKI et HAILPERN [HA01]. Ce type de programmes est très complexe à cause de la présence du tampon et des hypothèses supplémentaires de fiabilité qu'il faut émettre à son intention. Cependant les méthodes proposées offrent un cadre de base au développement de méthodes correctes et sémantiquement complètes pour des programmes distribués et une étude future consistera en l'étude de tels systèmes de preuve.

Nous n'avons pas donné de système de preuve prenant en compte la présence de sémaphore dans la syntaxe des programmes et une étude complète basée en partie sur les embryons donnés par OWICKI et LAMPORT [OWL] reste à faire par la suite. Cette étude nous permettra d'envisager les instructions await d'OWICKI et GRIES [OWG]. La notion de rendrez-vous est une technique de synchronisation qui nous semble plus intéressante que celle de communication par messages de façon asynchrone dans un tampon et nous voulons généraliser notre étude des propriétés de fatalité à ce type de synchronisation. Nos preuves ont été longues et difficiles ; il convient donc de construire un outil d'aide à la preuve qui sera interactif. Ce projet sera, sans doute, long à réaliser mais permettra de constituer un ensemble d'algorithmes parallèles ou distribués, corrects.

Nous avons parlé d'équité faible et APT et OLDEROG [APO] traitent la question de l'équité forte dont la définition ne nous paraît pas assez générale et est à étudier formellement. Cette voie est à suivre en parallèle avec les projets cités ci-dessus. Certains s'attardent sur la logique temporelle et nous voulons mener dans cette voie une recherche approfondie, afin de donner un cadre équivalent nouveau de spécification et de preuves de propriétés quelconques de programmes quelconques.

NOTATIONS UTILISÉES

BIBLIOGRAPHIE

[APT1] APT K.R. — Ten years of HOARE's logic : a survey Part II : nondeterminism. Rapport de recherche 82-32, Juin 1982 LITP - LA 248, Université P. et M. CURIE, Université PARIS 7.

[APT2] APT K.R. — Formal Justification of a Proof System for Communicating Sequential Processes. J.A.C.M., January 1983, volume 80, Number 1, pages 197-216.

[APD] APT K.R. DELPORTE C. — An Axiomatization of the Intermittent Assertion Method (extended abstract). Rapport de Recherche 82-70, LITP - LA 248, Université P. et M. CURIE, Université PARIS 7.

[AFD] APT K.R. FRANCEZ N. DE ROEVER W.P. — A Proof System for Communicating Sequential Processes. ACM Trans. on Prog. Lang. and Sys., Vol. 2, n° 3, July 1980, Pages 359-385.

[APO] APT K.R. OLDEROG E.R. — Proof Rules and Transformations Dealing with Fairness. Rapport de Recherche 82-47, Octobre 1982, LITP - LA 248, Université P. et M. CURIE, Université PARIS 7.

[APP] APT K.R. PLOTKIN G.D. — Countable Nondeterminism and Random Assignment. Rapport de Recherche 82-7, Février 1982, LITP - LA 248, Université P. et M. CURIE, Université PARIS 7.

[APS] APT K.R. PNUELI A. STAVI J. — Fair Termination Revisited with Delay. Rapport de Recherche 82-51, Octobre 1982, LITP - LA 238, Université P. et M. CURIE, Université PARIS 7.

[BEN1] BEN-ARI M. Complexity of Proofs and Models in Programming Logics. Ph. D. Thesis, May 1981, TEL-AVIV UNIVERSITY.

[BEN2] BEN-ARI M. On-the-Fly Garbage Collection : New algorithms Inspired by Programs Proofs. ICALP 82.

[BMP] BEN-ARI M. MANNA Z. PNUELI A. The Temporal Logic of Branching Time. Eighth Annual ACM Symposium on Principles of Programming Languages. (Williamsburg), January 1981, p. 164-176.

[BOC] BOCHMANN G.V. Hardware Specification with Temporal Logic : An Example. IEEE Transactions on Computers, Vol. C-31, N° 3, March 1982.

[BUR] BURSTALL R.M. Program Proving as Hand Simulation with a Little Induction. Proceedings IFIP 74, North-Holland, pages 308-312, Amsterdam 1974.

[CAR] CARVAHLO O.S.F.- ROUCAIROL G. Une Amélioration de l'Algorithme d'Exclusion Mutuelle de RICART et AGRAWALA. Rapport de Recherche 81-58, Novembre 1981, LITP - LA 248, Université P. et M. CURIE, Université PARIS 7. et CACM, février 1983.

[CES] CLARKE E.M. EMERSON E.A. SISTLA A.P. Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications : A Practical Approach. 10 th Annual Symposium on Principles of Programming Languages, Austin, Texas, January 1983.

[COO] COOK S.A. Soundness and Completeness of an Axiom System for Program Verification. SIAM J. COMPUTING, Vol. 7, N° 1, February 1978.

[COC1] COUSOT P. COUSOT R. Reasoning About Invariance Proof Methods. Proc. Int. Workshop on Program Construction, Château de BONAS, FRANCE, Tome 1, INRIA eds. (sept. 1980).

[COC2] COUSOT P. COUSOT R. "A la FLOYD" Induction Principles for Proving Inevitability Properties of Programs. Joint US-Franco-British Seminar, June 1982, FONTAINEBLEAU.

[COU] COUSOT R. Proving Invariance Properties of Parallel Programs by Backward Induction. Rapport de Recherche CRIN - 81 - P026, mars 1981, CRIN - LA 262, Université de NANCY I.

[DLMS] DIJKSTRA E.W. LAMPORT L. MARTIN A.J. SCHOLTEN C.S. STEFFENS E.F.M. On-the-Fly Garbage Collection : An Exercise in Cooperation. C.A.C.M., November 1978, Volume 21, Number II.

[EMA1] EMERSON E.A. HALPERN J.Y. Decision Procedure and Expressiveness in the Temporal Logic of Branching Time. 14 th Annual ACM Symposium on Theory of Computing, 1982.

[EMA2] EMERSON E.A. HALPERN J.Y. "SOMETIMES" and "NOT NEVER" Revisited : On Branching versus Linear Time (Preliminary Report). 10 th Annual Symposium on Principles of Programming Languages, Austin, Texas, January 1983.

[FLOY] FLOYD R.W. Assigning Meanings to Programs. Proc. Symp. in Applied Math., AMS, Vol. 19, Providence, R.I., USA, 1967, 19-32.

[GRF] GRÜMBERG O. FRANCEZ N. A Complete Proof Rule For (Weak) Equifair Termination. Research Report, Computer Science, RC 9634 (# 42539), 10/15/82. IBM Research Division Yorktown Heights, N.Y.

[HAO1] HAILPERN B. OWICKI S. Modular Verification of Computer Communication Protocols. Research Report, Computer Sciences, RC 8726, March 1981, IBM T.J. Watson Research Center Yorktown Heights, N.Y. 10598.

[HA02] HAILPERN B. OWICKI S. Modular Verification of Concurrent Programs. Research Report, Computer Science RC 9130, November 1981, IBM T.J. Watson Research Center Yorktown Heights, N.Y. 10598.

[HKP] HAREL D. KOZEN D. PARIKH R. Process Logic : Expressiveness, Decidability, Completeness. Proceedings of the 21th Symposium on Foundations of Computer Science. Syracuse, New-York, October 1980.

[HIN] HINMAN P.G. Recursion - Theoretic Hierarchies. Perspectives in Mathematical Logic, Springer - Verlag.

[HOA] HOARE C.A.R. An Axiomatic Basis for Computer Programming. C.A.C.M., October 1969, Volume 12, Number 10, pages 576-580.

[HUC] HUGHES G.E. CRESSWELL M.J. An Introduction to Modal Logic. Methuenen and Co Ltd (1971).

[KAM] KAMP J.A.W. Tense Logic and the Theory of Linear Order. Ph. D. Thesis, 1968, University of California, Log Angeles.

[KLE] KLEENE Introduction to metamathematics. Van Nostrand, Princeton, New-Jersey 1952.

[KUD] KUIPER R. DE ROEVER W.P. Fairness Assumptions for CSP in a Temporal Logic Framework. TC2 Working Conference on the Formal Description of Programming Concepts, Garmisch-Partenkirchen, June 1982.

[LAM1] LAMPORT L. A New Approach to Proving the Correctness of Multiprocess Programs. ACM Transactions on Programming Languages and Systems, Vol. 1, N° 1, July 1979, Pages 84-97.

[LAM2] LAMPORT L. The "HOARE" Logic of Concurrent Programs. Acta Informatica 14, 1980, pages 21-37.

[LAM3] LAMPORT L. Reasoning About Nonatomic Operations. 10 th Annual ACM - SIGACT - SIGPLAN Symposium on Principles of Programming Languages in January 1983, Austin, Texas.

[LAM4] LAMPORT L. "Sometime" Is Sometimes "Not Never", On the Temporal Logic of Programs. 7th Annual Symposium on Principles of Programming Languages, 1980.

[LAS] LAMPORT L. SCHNEIDER F.B. The "HOARE Logic" of CSP, and All That. Technical Report, TR 82-490, May 1982, Department of Computer Science, CORNELL University.

[LPS] LEHMAN D. PNUELI A. STAVI J. Impartiality, Justice and Fairness The Ethics of Concurrent Termination. Proc. 8 th Coll. on Automata Languages and Programming, Springer-Verlag, Lecture Notes in Computer Science, 115, p.p. 264-277, 1981.

[MAN] MANNA Z. Verification of Sequential Programs : Temporal Axiomatization. Report. N) STAN-CS-81-877, Department of Computer Sciences, Stanford University.

[MAP1] MANNA Z. PNUELI A. Verification of Concurrent Programs, Part. I : The Temporal Framework. Report. N° STAN - CS - 8'- 836, Department of Computer Science, Stanford University.

[MAP2] MANNA Z. PNUELI A. Verification of Concurrent Programs, Part. II : Temporal Proof Principles. Report, N° STAN - CS - 81 - 843, Department of Computer Science, Stanford University.

[MAP3] MANNA Z. PNUELI A. How To Cook A Temporal Proof System For Your Pet Language. 10 th Annual ACM - SIGACT - SIGPLAN Symposium on Principles of Programming Languages in January 1983, Austin Texas.

[MAW] MANNA Z. WOLPER P. Synthesis of Communicating Processes from Temporal Logic Specifications. Report N° STAN CS 81 - 872, Department of Computer Science, Stanford University.

[MOS] MOSCHOVAKIS Y.N. Elementary Induction On Abstract Structures. Studies in Logic and the Foundations of Mathematic, North-Holland Publishing Company.

[NIS] NISHIMURA H. HOARE Method for Concurrent Programs I. Proceedings of the Sixth IBM Symposium on Mathematical Foundations of Computer Science Logic aspects of Programs. May 1981, Japan.

[OWG] OWICKI S. GRIES D. An Axiomatic Proof Technique for Parallel Programs I . Acta Informatica 6, 1976, pages 319-340.

[OWL] OWICKI S. LAMPORT L. Proving Liveness Properties of Concurrent Programs. SRI International.

[PLO] PLOTKIN G.D. A Powerdomain For Countable Non-Determinism. November 1981, Department of Computer Science, University of Edinburgh.

[PNU] PNUELI A. The Temporal Logic of Programs. 18 th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, Oct. 31 - Nov 2, 1977, pages 46-57.

[PRA] PRATT V.R. Semantical Considerations on Floyd - Hoare Logic. 17 th Symposium on Foundations on Computer Science, IEEE, October 1976.

[RIA] RICART G. AGRAWALA A.K. An Optimal Algorithm for Mutual Exlcusion in Computer Net-works. C.A.C.M., January 1981, Volume 24, Number 1.

[ROG] ROGERS H.Jr. Theory of Recursive Functions and Effective Computability. Mc Graw-Hill Book Company.

[SAL] SALWICKI A. BANACHOVSKI L. KRECZMAR A. MIRKOWSKA G. RASIOWA H. An Introduction to Algorithmic Logic Metamathematical Investigations in the Theory of Programs. "Un grain de Sem n° 1", Poitiers, décembre 1980. Compte rendu de GROSSEM.

[SCH] SCHLICHTING R.D. Axiomatic Verification to Enhance Software Reliability. Technical Report, TR 82-480, Jaunary 1982, Department of Computer Science, CORNELL University.

[SCS] SCHLICHTING R.D. SCHNEIDER F.B. Using Message Passing for Distributed Programming : Proof Rules and Disciplines. Technical Report, TR 82-5, Department of Computer Science, University of Arizona.

[SCM1] SCHWARTZ R.L. MELLIAR-SMITH P.M. Temporal Logic Specification of Distributed Systems. Proceedings of the Second International Conference on Distributed Systems, PARIS, April 1981.

[SCM2] SCHWARTZ R.L. MELLIAR-SMITH P.M. From State Machines to Temporal Logic : Specification Methods for Protocol Standard IEEE Transactions on Communications, Vol.Con-30. n° 12. december 1982.

[SHO] SHOENFIELD J.R. Mathematical Logic. Addison - Wesley Publishing Company.

[SIC] SISTLA A.P. The Complexity of Propositional Temporal Logics. Technical Report, TR 05-82. HARVARD University.

[SCFG] SISTLA A. CLARKE E. FRANCEZ N. GUREVICH Y. Can Message Buffers Be Characterized In Linear Temporal Logic ? Technical Report. HARVARD University

[SPE] SPECTOR Inductively defined sets of natural numbers. Infinistic methods (Pergamon, New-York, 1961, p. 97-102.

[VAN] VAN BENTHEM J.F.A.K. The logic of Time Cours du Filosofisch Institut, Rijk universiteit, GRONINGEN, 1980.

[WOL] WOLPER P. Temporal Logic Can Be more Expressive. Rapport de Recherche de l'Université de Stanford, Californie.

institut national polytechnique de lorraine

Le Président,

N/Réf. : Scol.

AUTORISATION DE SOUTENANCE DE THESE DE DOCTORAT 3ème CYCLE

-:-:-:-:-:-:-:-:-:-:-:-:-

VU LE RAPPORT ETABLI PAR :

Monsieur le Professeur COUSOT P.

le Président de l'Institut National Polytechnique de Lorraine autorise :

Monsieur MERY Dominique

à soutenir, devant l'I.N.P.L., une thèse de Doctorat intitulée :

"UNE METHODE AXIOMATIQUE DE PREUVE DE PROPRIETES DE FATALITE DE PROGRAMMES PARALLELES AVEC HYPOTHESE D'EXECUTION EQUITABLE."

en vue de l'obtention du titre de DOCTEUR 3ème CYCLE

Spécialité *"INFORMATIQUE"*

Fait à NANCY, le

Le Président de l'I.N.P.L.

M. LUCIUS