VERIFICATION STATIQUE DE LA COHERENCE DYNAMIQUE DES PROGRAMMES

-:-:-:-:-:-:-:-:-:-:-:-:-:-

(1) Patrick COUSOT
(2) Radhia COUSOT

(1) - Attaché de Recherche au CNRS
(2) - Chercheur sous contrat IRIA-SESORI N° 75-035

23 Septembre 1975

VERIFICATION STATIQUE DE LA COHERENCE DYNAMIQUE DES PROGRAMMES

-:-:-:-:-:-:-:-:-:-:-:-:-:-

1°) - PRESENTATION DU PROBLEME -

La notion de type ou mode dans les langages de programmation, permet généralement une vérification statique (à la compilation), de la cohérence dynamique (à l'exécution) des programmes. Toute valeur a un mode unique, qui caractérise les actions qui peuvent être exécutées sur ou avec elle. En ALGOL 68 par exemple, les déclarations :

```
struct (int jour, string mois, int an) t = (11, "juillet", 1971) ;
struct (long bits c) u ;
```

permettent au compilateur de détecter que les écritures

```
mois of u
```

ou

```
c of t
```

sont erronées.

Pour certains types toutefois, certaines opérations sur des objets de ce type, ne sont pas définies pour certaines valeurs ayant ce type. Par exemple, en PASCAL [1] on peut déclarer :

```
type personne = record
                    nom : alfa ;
                    père, mère : ↑ personne ;
                end ;
var x, y, z : ↑ personne ;
...
```

mais l'opération x ↑ • nom n'est définie que si x ≠ nil. Les techniques de compilation actuelles ne permettent pas de déterminer que x n'est pas nil quand on accède à un champ de l'enregistrement repéré par x. De ce fait, certains compilateurs* insèrent des tests dans le code généré pour détecter ce genre d'erreur à l'exécution du programme.

* La plupart des compilateurs utilisent le mécanisme de protection mémoire (à un coût pratiquement nul), mais cette technique n'est pas utilisable pour des langages d'implémentation de système, comme LIS.

x • ↑ nom sera compilé comme :

```
if x ≠ nil then
     ... x • ↑ nom ...
else
     erreur à l'exécution ;
fi ;
```

Ces tests sont généralement redondants avec ceux que le programmeur a placés dans son programme d'après la logique de son problème.

Un deuxième exemple de vérification dynamique de la cohérence des programmes, est celui des objets de type discret numérique ("Subrange Type" [1, 6.1.2], type discret numérique en français).

Ce type est défini comme un intervalle fermé du type entier, par indication de la plus petite et de la plus grande valeur dans cet intervalle.

Exemple 1 : var v : -10 .. +23 ;

La définition du langage PASCAL est telle qu'un compilateur ne peut pas vérifier statiquement, qu'une variable de type discret numérique t ne prendra en cours d'exécution que des valeurs situées dans l'intervalle défini par t.

Le point de vue de l'écrivain de compilateur est alors le suivant [2] :

a) - En Pascal, le type d'un objet 0 déclaré de type discret numérique t est le type entier.

b) - On teste à l'exécution que les valeurs de l'objet 0 sont dans les limites de l'intervalle défini par t.

Exemple 2 : Dans l'instruction d'affectation :

```
v := v + 5
```

on doit tester à l'exécution, que la valeur délivrée par l'expression est dans l'intervalle -10 .. 23.

Malheureusement le coût de ces tests de validité insérés systématiquement par le compilateur dans les programmes est généralement jugé excessif par les utilisateurs. Ils utilisent donc une version du compilateur qui n'effectue pas l'insertion de vérifications dynamiques, ce qui les conduit à exploiter des programmes faux, à découvrir leurs erreurs après une longue période

d'exploitation dans le meilleur cas, à obtenir des résultats erronés mais tenus pour corrects dans le pire. Pour obtenir un compromis entre fiabilité et coût d'exploitation des programmes, nous pensons que le programme objet doit tenir compte de toutes les déclarations de l'auteur du programme source, et que pour cela, le compilateur doit insérer un nombre minimum (et suffisant) de vérifications dans le programme objet.

Ce document présente une technique de compilation, permettant de déterminer statiquement en chaque point d'un programme, un domaine de valeurs possibles pour les objets de ce programme. Par exemple, on peut déterminer que dans le programme PASCAL :

```
     var j : integer
         t : array [1 .. 11] of real ;
     j := 0 ;
     while j ≤ 10 do
          begin
              j := j+1 ;
[α]           ... T(j) ... ;
          end
```

la valeur de j en [α] est comprise entre 1 et 11, donc, en particulier qu'il est inutile de tester dynamiquement que j est compris entre les bornes du tableau T.

La détermination statique des domaines des valeurs dynamiques des objets d'un programme, est basée sur l'interprétation abstraite des programmes telle qu'elle a été utilisée par SINTZOFF pour faire des preuves de programmes [3], par KILDALL pour l'optimisation globale [4], par WEGBREIT pour l'extraction de propriétés des programmes [5], et par KARR pour découvrir des relations linéaires ($\alpha x + \beta y + \ldots = \gamma$) entre des variables (x, y, ...) d'un programme [6].

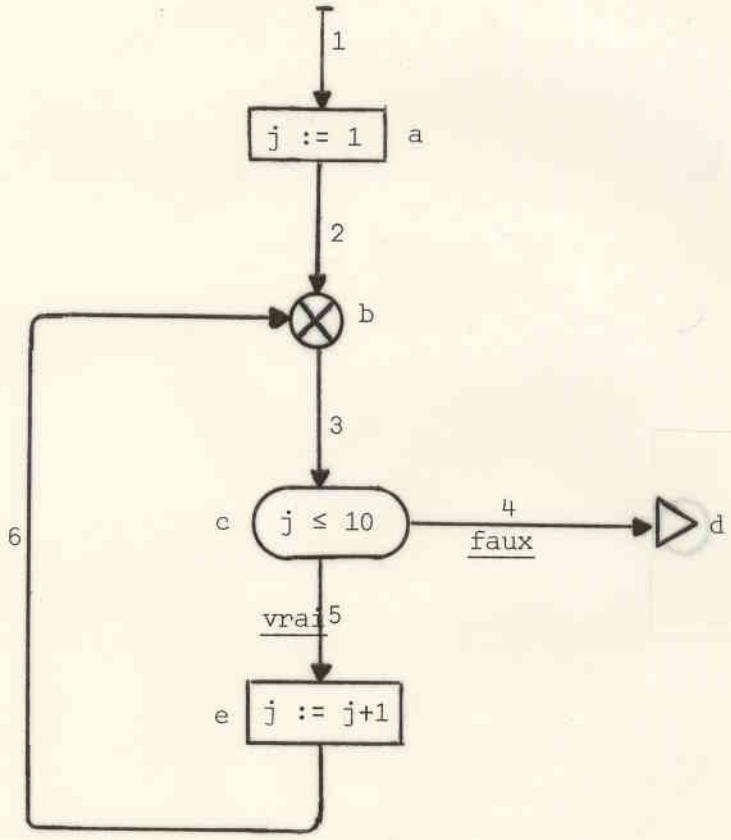
Dans notre cas, cette interprétation abstraite des programmes consiste à "exécuter" ces programmes, non pas avec des valeurs associées aux variables, mais avec des domaines de valeurs associées aux variables. On peut ainsi déterminer quel est le domaine des valeurs possible pour les variables en chaque point d'un programme, ce qui permet d'assurer statiquement la cohérence dynamique des programmes.

2) - EXEMPLE -

L'exemple que nous choisissons pour la présentation informelle de la méthode est volontairement très simple. Il permet de présenter la méthode, sans prétendre illustrer sa puissance :

```
var j : integer ;
j := 1 ;
while j ≤ 10 do
     j := j+1 ;
```

sous forme d'organigramme, on obtient :



Pour faire l'interprétation abstraite, on associe à chaque arc de l'organigramme, un contexte abstrait, qui est un ensemble de couples (identificateur, valeur abstraite), différant par leurs identificateurs. Une valeur abstraite, est un objet qui désigne un ensemble de valeurs concrètes. C'est ainsi, que dans notre exemple, les valeurs abstraites seront des intervalles fermés sur les entiers, [a, b], qui désignent les valeurs concrètes entières, a, a+1, a+2, ..., b ($b \geq a$).

A la fin de l'interprétation abstraite, les contextes abstraits ont la propriété suivante :

(P) quelle que soit la valeur concrète "v_c" affectée à un identificateur "i" sur un arc "a" d'un programme "p", lors d'une exécution réelle (quelconque mais correcte) de "p", "v_c" appartient à l'ensemble des valeurs concrètes désignées par la valeur abstraite "v_a" de "i" dans le contexte abstrait "C" associé à l'arc "a" à la fin de l'interprétation abstraite.

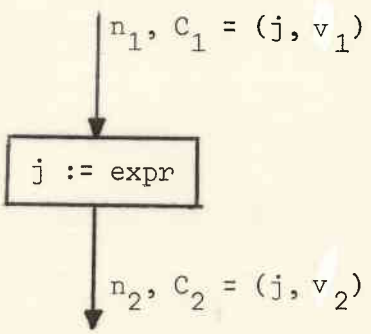
La construction des contextes abstraits associés aux arcs du programme se fait pas à pas, comme suit : on commence par l'arc d'entrée du programme, avec un certain contexte initial, et le contexte vide sur les autres arcs. Pour chacun des trois types de noeuds (affectation, décision, jonction), on décrit une transformation qui spécifie le(s) contexte(s) abstrait(s) sur l'(es) arc(s) de sortie de ce noeud en fonction :

- des contextes abstraits sur les arcs d'entrée du noeud,
- et quand c'est nécessaire, du contenu de ce noeud.

Ces transformations ont la propriété que si les contextes d'entrée satisfont (P), alors le(s) contexte(s) de sortie satisfait(ont) (P). Ces transformations sont appliquées, jusqu'à ce que les contextes abstraits soient"stabilisés", c'est-à-dire que l'application d'une transformation sur un noeud quelconque ne modifie pas le(s) contexte(s) sur l'(es) arc(s) de sortie.

Nous décrivons maintenant sommairement ces transformations pour les différents types de noeuds. Elles constituent les opérations de base ou élémentaires de l'interprétation abstraite. Nous ne faisons pas une présentation générale, mais nous utilisons l'exemple de valeurs abstraites de type intervalle d'entiers, pour déterminer les domaines de valeurs des variables entières.

L'interprétation abstraite élémentaire d'un noeud affectation :



consiste à évaluer l'expression "expr" dans le contexte C_1, ce qui délivre

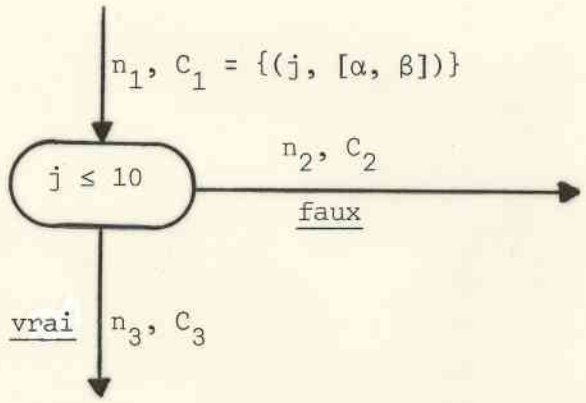
une valeur "v", qui est affectée à j, c'est-à-dire que la valeur "v_2" associée à j dans le contexte C_2 est remplacée par "v" (ou un couple (j, v) est introduit dans C_2 si j n'avait pas de valeur dans le contexte C_2).

Une expression réduite à une constante c s'évalue en l'intervalle [c, c].

Une expression réduite à un identificateur s'évalue dans un contexte C, en la valeur associée à cet identificateur dans le contexte C. (Pour simplifier à ce point, notons que si l'identificateur n'est pas défini dans le contexte C, son évaluation peut conduire à la valeur par défaut $[-\infty, +\infty]$).

Une expression e somme de deux expressions $e_1 + e_2$ s'évalue, en évaluant e_1 ce qui donne $[\alpha, \beta]$, en évaluant collatéralement e_2 ce qui donne $[\gamma, \delta]$ puis en délivrant la valeur abstraite $[\alpha + \gamma , \beta + \delta]$.

L'interprétation d'un noeud test comme :



s'effectue comme suit :

1er cas : $\beta \leq 10$: Pour toutes les valeurs possibles de j comprises entre α et β, le test est vrai, donc on continue l'interprétation abstraite par la branche n_3, avec le contexte C_3 égal à C_1, puisqu'aucun élément du contexte C_1 n'a été modifié par la comparaison.

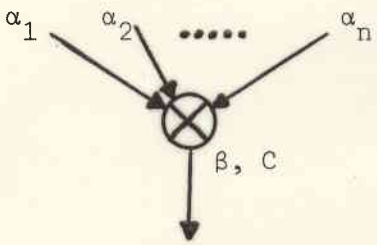
2ème cas : $\alpha > 10$, on continue l'interprétation par n_2, avec C_2 égal à C_1.

3ème cas : $\alpha \leq 10 < \beta$

Dans une interprétation réelle, l'arc n_3 sera suivi si $j \leq 10$, donc pour les valeurs de j comprises entre α et 10, tandis que l'arc n_2 sera suivi pour les valeurs de j comprises entre 11 et β. Dans l'interprétation abstraite, les deux chemins sont suivis de façon quasi-parallèle n_3, $C_3 = \{(j, [\alpha, 10])\}$ et n_2, $C_2 = \{(j, [11, \beta])\}$. Le choix du chemin à suivre en premier n'a aucune importance. On voit donc, qu'à cause des noeuds de

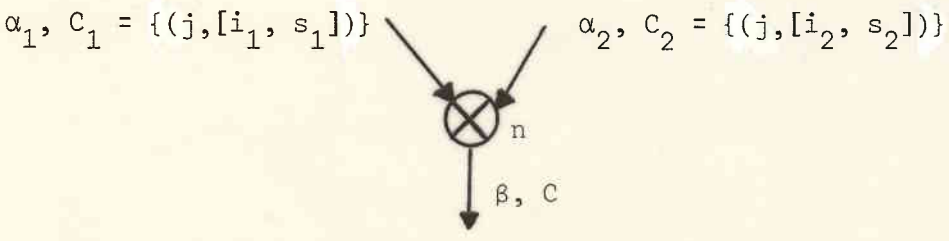
test, l'interprétation abstraite doit suivre, non pas un chemin d'exécution, mais plusieurs, à tour de rôle. Quand on sélectionne un chemin parmi ceux à suivre, on l'exécute, jusqu'à arriver à un noeud de sortie

(auquel cas, on s'arrête et l'on choisit de poursuivre avec un autre chemin candidat à l'exécution) ;
ou à un noeud de jonction :



(auquel cas, on suspend l'exécution de ce chemin, pour continuer avec un des chemins candidats à l'exécution). Les noeuds de jonctions permettent donc la synchronisation des exécutions quasi-parallèles. Ceci est nécessaire puisque l'information dont on dispose sur l'arc β, est l'"union" des informations dont on dispose sur les arcs α_1, ..., α_n. Avant de continuer l'exécution par l'arc de sortie du noeud de jonction il est naturel d'attendre d'avoir rassemblé toutes les informations disponibles sur les arcs d'entrée. Quand toutes les exécutions quasi-parallèles sont bloquées sur les noeuds de jonction, on peut calculer l'information disponible sur l'arc de sortie de chacun de ces noeuds, puis reprendre les exécutions quasi-parallèles.

Supposons que les exécutions quasi-parallèles soient bloquées sur un noeud de jonctions dans l'état :



Sur la branche α_1, j est compris entre i_1 et s_1, tandis que sur la branche α_2, j est compris entre i_2 et s_2. On dispose donc sur la branche de sortie β, de l'information que j est compris dans l'union des intervalles $[i_1, s_1]$ et $[i_2, s_2]$. On peut définir cette union, comme suit :

$[i_1, s_1] \bar{\cup} [i_2, s_2] = [\min(i_1, i_2), \max(s_1, s_2)]$

Noter que dans le cas d'intervalles disjoints, cette opération correspond à une perte d'information puisque si par exemple $s_1 < i_2$, on affirme sur la branche β de sortie que j peut être compris entre s_1 et i_2 ce qui n'est pas le cas sur la branche α_1 ou α_2. Cette perte d'information se justifie, puisque la notion d'appartenance à des intervalles disjoints n'existe pas comme notion de base du langage PASCAL. De façon générale l'interprétation abstraite peut ignorer toutes les informations qui ne sont pas utiles pour l'analyse particulière du programme interprété.

Dans l'interprétation abstraite d'un programme, on est amené à la suite de chaque itération dans un cycle, à passer par l'arc β. Supposons qu'une première itération ait associé le contexte C à l'arc β. (Eventuellement $C = \Phi$ contexte vide à l'initialisation). Quand cette exécution s'arrête à nouveau sur le noeud n de jonction, elle le fait avec C_1 sur α_1 et C_2 sur α_2. Soit $C' = C_1 \bar{\bar{\cup}} C_2$, union des contextes C_1 et C_2. Supposons pour simplifier que $C' = \{(j, [i',s'])\}$ et $C = \{(j, [i, s])\}$. On remarque tout de suite que si $C' = C$, il est inutile de continuer l'interprétation abstraite, qui avec les mêmes hypothèses ne peut conduire qu'aux mêmes conclusions. De même si $C' \bar{\bar{\leq}} C$, (C' est inclus dans C), c'est-à-dire que $i \leq i' \leq s' \leq s$, il est également inutile de continuer l'interprétation abstraite sur la branche β, avec la justification intuitive que si le domaine de j est plus petit sur l'arc β qu'il n'était auparavant, il en sera de même sur tous les arcs des chemins d'origine β. Enfin, si le domaine de j dans C' n'est pas inclus ou égal à celui de j dans C, il faut continuer l'interprétation abstraite sur l'arc β, parce que les conclusions que l'on a obtenues à l'itération précédente ne sont pas les plus larges que l'on peut obtenir. Dans ce cas l'interprétation continue en remplaçant C par $C \bar{\bar{\cup}} C'$. (A la première itération avec $\Phi \bar{\bar{\cup}} C' = C'$). Noter, qu'après m passages sur l'arc β, on utilise le contexte $(\ldots ((\Phi \bar{\bar{\cup}} C'_1) \bar{\bar{\cup}} C'_2) \ldots \bar{\bar{\cup}} C'm)$ sur cet arc. Pour que le processus termine, il faut que la suite $Co = \Phi$, $C_1 = (\Phi \bar{\bar{\cup}} C'_1)$, $C_2 = ((\Phi \bar{\bar{\cup}} C'_1) \bar{\cup} C'_2)$, $\ldots C_m = (\ldots((\Phi \bar{\bar{\cup}} C'_1) \bar{\bar{\cup}} C'_2) \ldots \bar{\bar{\cup}} C'_m)$ converge, plus précisément, qu'il existe ℓ tel que $C_{\ell+1}$ est inclus ou égal à C_ℓ. Noter, que notre opération $\bar{\cup}$ d'union, n'assure pas cette convergence dans le cas général : si par exemple $C'_i = \{(j, [1, i])\}$, on a $C_o = \Phi$, $C_1 = C'_1 = \{(j, [1, 1])\}$, ..., $C_i = C'_i \bar{\bar{\cup}} C'_{i-1} = \{(j, [1, i])\}$, suite non convergente. Pour assurer cette convergence, on remplacera C par $C \bar{\bar{\nabla}} C'$, avec

$$[i_1, s_1] \bar{\nabla} [i_2, s_2] = [(\underline{si}\ i_2 < i_1\ \underline{alors}\ -\infty\ \underline{sinon}\ i_1),$$
$$(\underline{si}\ s_2 > s_1\ \underline{alors}\ +\infty\ \underline{sinon}\ s_1)]$$

On a maintenant :

$C_o = \Phi$

$C_1 = \Phi \bar{\bar{\nabla}} \{(j, [1, 1])\} = \{(j, [1, 1])\}$

$C_2 = C_1 \bar{\bar{\nabla}} C'_1$

$= \{(j, [1, 1])\} \bar{\bar{\nabla}} \{(j, [1, 2])\}$

$= \{(j, [1, 1] \bar{\nabla} [1, 2])\} = \{(j, [1, +\infty])\}$

$C_3 = C_2 \bar{\bar{\nabla}} C'_3$

$= \{(j, [1, +\infty])\} \bar{\bar{\nabla}} \{(j, [1, 3])\}$

$= \{(j, [1, +\infty] \bar{\nabla} [1, 3])\} = \{(j, [1, +\infty])\}$

$= C_2$

On a évidemment une perte d'information, mais les exemples que nous donnerons confirment que ce choix est judicieux, en particulier, nous démontrerons que l'interprétation abstraite de tous les programmes termine, même si ces programmes ne terminent pas dans les exécutions réelles.

Avant de traiter notre exemple introductif, il reste à expliquer pourquoi, sur un arc de sortie d'un noeud affectation ou test, on ne fait pas l'union des contextes obtenus à chaque itération :

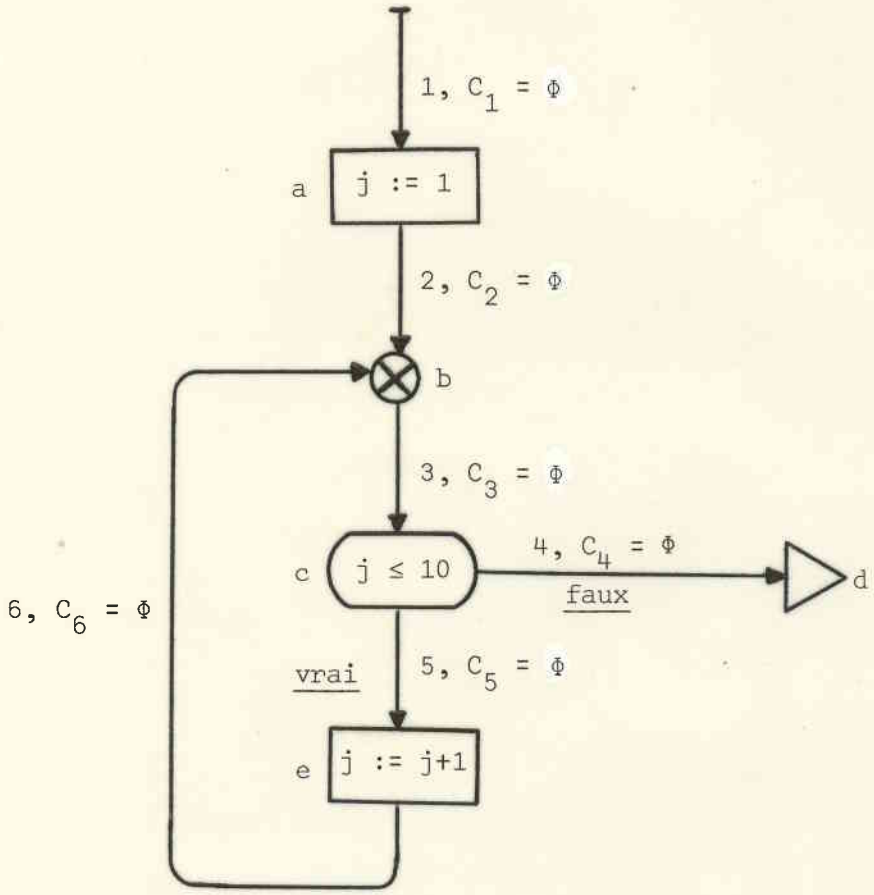
si une itération supplémentaire a été nécessaire c'est que C'_{n+1} n'est pas inclus dans C'_n.

Mais d'après le traitement fait aux noeuds de jonctions C'_n est inclus dans C'_{n+1}, puisque C'_{n+1} est de la forme $C'_n \bar{\bar{\nabla}} C'$. Dans ces conditions l'interprétation du noeud affectation dans le contexte C'_{n+1} qui inclut C'_n, produit

un contexte C qui inclut C_n, par conséquent C_{n+1} = C qui inclut $C_n \bar{\cup} C$. Il en va de même pour les noeuds tests.

Une présentation plus rigoureuse de l'algorithme d'interprétation abstraite suit cette introduction informelle, et complète cette explication.

Le programme que nous avons choisi en exemple, est interprété en commençant par les conditions initiales :



l'interprétation commence par le noeud a, l'expression 1 dans l'instruction j := 1, a la valeur abstraite [1, 1] et par conséquent $C_2 = \{(j, [1, 1])\}$. Le noeud de jonction b, synchronise les chemins d'exécution quasi-parallèles. Le contexte C' de sortie est :

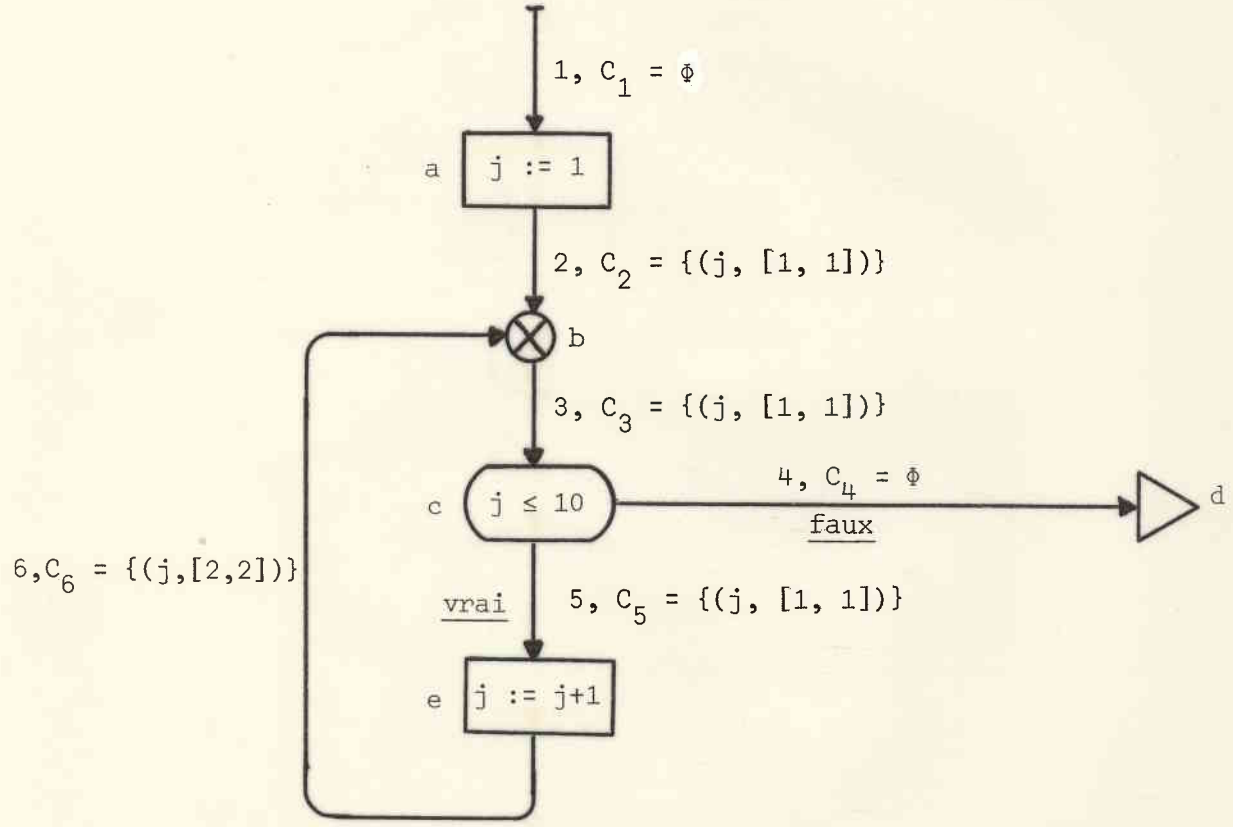
$$C' = C_6 \bar{\cup} C_2$$
$$= \Phi \bar{\cup} \{(j, [1, 1])\} = \{(j, [1, 1])\}$$

puis $C_3 := \Phi \bar{\nabla} C' = \{(j, [1, 1])\}$

l'interprétation du noeud c, nous conduit à prendre la branche vraie, car si j est égal à 1 il est inférieur ou égal à 10, donc $C_5 = \{(j, [1, 1])\}$. L'interprétation du noeud e, j := j+1, conduit à :

$$C_6 = \{(j, [1, 1] + [1, 1])\}$$
$$= \{(j, [2, 2])\}$$

On obtient, en attente sur le noeud b :



Le contexte de sortie sur le noeud b, est

$$C' = C_2 \bar{\cup} C_6 = \{(j, [1, 1])\} \bar{\cup} \{(j, [2, 2])\}$$
$$= \{(j, [1, 2])\}$$

On remplace donc C_3 par $C_3 \bar{\nabla} C'$ soit :

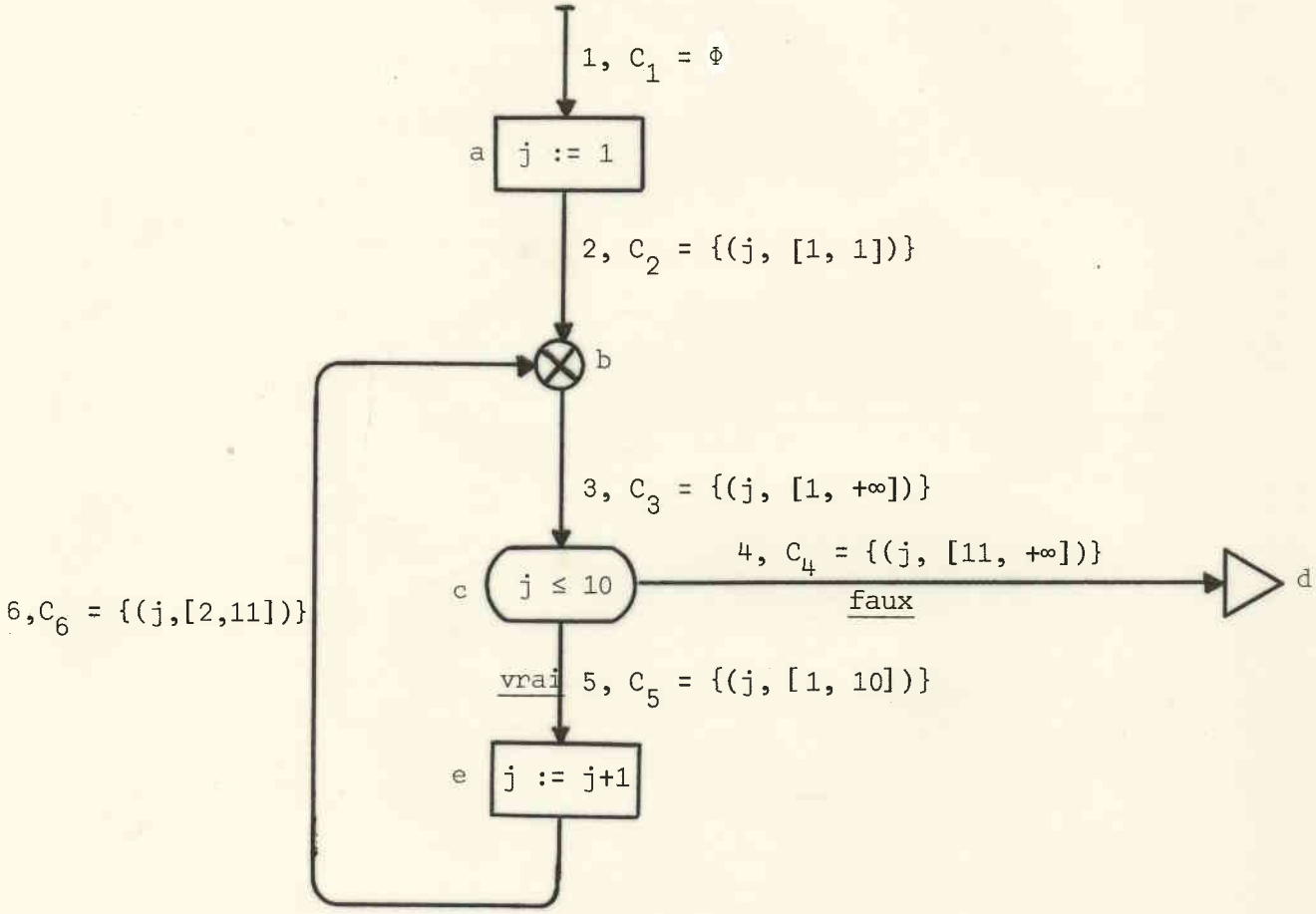
$$C_3 = \{(j, [1, 1])\} \bar{\nabla} \{(j, [1, 2])\}$$
$$= \{(j, [1, 1] \bar{\nabla} [1, 2])\} = \{(j, [1, +\infty])\}$$

l'interprétation du noeud c, nous conduit alors à deux chemins d'exécution :

- D'une part, sur l'arc 4, avec $C_4 = \{(j, [11, +\infty])\}$ qui conduit au noeud d et termine,
- D'autre part, sur l'arc 5, avec $C_5 = \{(j, [1, 10])\}$ l'interprétation du noeud e, conduit alors à :

$$C_6 = \{(j, [1, 10] + [1, 1])\}$$
$$= \{(j, [2, 11])\}$$

On obtient, en attente sur le noeud b :



Le contexte de sortie sur le noeud b, est :

$$C' = C_2 \bar{\cup} C_6$$
$$= \{(j, [1, 1])\} \bar{\cup} \{(j, [2, 11])\}$$
$$= \{(j, [1, 1] \bar{\cup} [2, 11])\} = \{(j, [1, 11])\}$$

Mais $C' = \{(j, [1, 11])\}$ est inclus dans C_3 égal à $\{(j, [1, +\infty])\}$ et l'interprétation abstraite s'arrête. On a ainsi obtenu un domaine de valeurs pour j sur chaque arc du graphe de programme. Noter que la convergence est très

rapide, mais que, en contre partie, l'inteprétation abstraite conduit à des résultats faibles, par exemple, sur l'arc 4 on a trouvé que j est compris entre 11 et $+\infty$, tandis que lors des exécutions réelles, j sera égal à 11. Toutefois, l'intervalle trouvé dans l'interprétation abstraite, inclut le domaine des valeurs réelles de j.

Par extrapolation des résultats obtenus sur cet exemple, on s'aperçoit que l'inteprétation abstraite permet :

- la détection des constantes : une variable ayant la même valeur [c, c] sur toutes les branches de l'organigramme est une constante égale à c.
- la détection de certaines variables non initialisées, sur les arcs où cette variable ne figure pas dans le contexte associé (comme l'arc 1 de notre exemple).
- la détection de certaines branches mortes, c'est-à-dire celles qui ont un contexte final vide. (Reprendre l'exemple, en remplaçant le noeud e pour j := j-1 et la branche 4 est morte). Noter que certaines branches, pour la détection des erreurs de programmation, du genre :

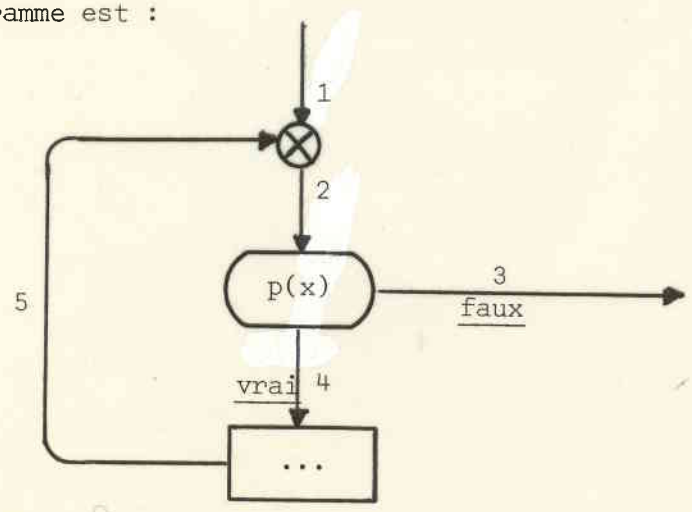
```
if (j < 2) ∨ (j < 11) then
    write "erreur de programmation" ;
    stop ;
fi ;
```

sont redondantes, mais utiles à la lisibilité des programmes et peuvent être éliminées automatiquement par une analyse statique du programme.

- La détection de certaines boucles infinies, par exemple, une boucle comme

```
while p(x) do
    begin
        ...
    end ;
```

dont l'organigramme est :



est infinie, si l'évaluation du prédicat p(x) dans le contexte associé à la branche 5 est toujours "vrai".

Bien sûr, ce résultat étant indécidable, il n'est pas sûr que l'on détectera ainsi toutes les branches mortes, boucles infinies, etc... du programme.

- La détermination du type des variables dans des langages "sans-types" comme LISP ou APL ....
- Etc....

Après cette introduction, il nous reste à présenter de façon rigoureuse l'algorithme d'interprétation abstraite, et l'interprétation des opérations élémentaires, puis à donner quelques exemples convaincants.

2°) - ALGORITHME D'INTERPRETATION ABSTRAITE -

La correction de l'algorithme d'interprétation abstraite repose sur un certain nombre de propriétés mathématiques des valeurs et contextes abstraits. Un préliminaire introduit ces propriétés.

2.1 - Préliminaires mathématiques -

Soit un ensemble E et deux opérations $\cup$ et ∇ définies sur E, satisfaisant les axiomes suivants :

A1 : $E \times E \xrightarrow{\cup} E$

$\forall (x, y, z) \in E^3$, on a :

A2 : $(x \cup (y \cup z)) = ((x \cup y) \cup z)$ associativité

A3 : $(x \cup y) = (y \cup x)$ commutativité

A4 : $(x \cup x) = x$ idempotence

A5 : $\{\exists\ \emptyset \in E \mid x \cup \emptyset = \emptyset \cup x = x\}$ élément neutre

PDEF1 : $\{x \leq y\} \overset{def}{\Longleftrightarrow} \{x \cup y = y\}$

PDEF2 : $\{x < y\} \overset{def}{\Longleftrightarrow} \{(x \leq y) \wedge (x \neq y)\}$

A6 : $E \times E \xrightarrow{\nabla} E$

A7 : $(x \cup y) \leq (x \nabla y)$

Tout triplet $(E, \cup, \nabla)$ satisfaisant les axiomes A1 à A7, possède les propriétés suivantes :

lemme P1

la relation $\leq$ est une relation d'ordre partiel.

réflexivité :

(A4) $\Rightarrow$ $(x \cup x) = x \Rightarrow x \leq x$

antisymétrie :

$$(x \leq y) \wedge (y \leq x)$$

(PDEF1) $\Rightarrow ((x \cup y) = y) \wedge ((y \cup x) = x)$

(A3) $\Rightarrow ((x \cup y) = y) \wedge ((x \cup y) = x)$

$\Rightarrow y = x$

transitivité :

$$(x \leq y) \wedge (y \leq z)$$

(PDEF1) $\Rightarrow ((x \cup y) = y) \wedge ((y \cup z) = z)$ (1)

$\Rightarrow ((x \cup y) \cup z) = z$

(A2) $\Rightarrow (x \cup (y \cup z)) = z$

(1) $\Rightarrow (x \cup z) = z$

(PDEF1) $\Rightarrow x \leq z$

lemme P2

$\{\forall (x, y) \in E^2 \quad , \quad x \leq x \cup y\}$

$$((x \cup y) = (x \cup y)) \wedge ((x \cup x) = x) \qquad (A4)$$

$\Rightarrow ((x \cup x) \cup y) = (x \cup y)$

(A2) $\Rightarrow (x \cup (x \cup y)) = (x \cup y)$

(PDEF1) $\Rightarrow x \leq (x \cup y)$

lemme P3

$\{\forall (x, y) \in E^2 \quad , \quad (x \leq x \nabla y) \wedge (x \leq y \nabla x)\}$

(P2) $\Rightarrow x \leq x \cup y$ (1)

(A7) $\Rightarrow (x \cup y) \leq (x \nabla y)$

(P1, transitivité) $\Rightarrow x \leq x \nabla y$

(1) et (A3) $\Rightarrow x \leq (y \cup x)$

(A7) $\Rightarrow (y \cup x) \leq (y \nabla x)$

(P1, transitivité) $\Rightarrow x \leq y \nabla x$

lemme P4

$\{\forall (x, y) \in E^2 \quad \{x \nmid x \cup y\} \Rightarrow \{x \nmid x \nabla y\}\}$

Par l'absurde, supposons $x = x \nabla y$ (1)

(P2) $\Rightarrow x \leq x \cup y$ (2)

(A7) $\Rightarrow x \cup y \leq x \nabla y$

(1) $\Rightarrow x \cup y \leq x$ (3)

(2), (3) (P1, antisymétrie) $\Rightarrow x = x \cup y$, négation de l'hypothèse $x \neq x \cup y$, donc absurdité.

lemme P5

$\forall (x, y) \in E^2 \quad , \quad \{x = x \nabla y\} \Rightarrow \{x = x \cup y\}$

(A7) $\Rightarrow (x \cup y) \leq (x \nabla y)$

$\Rightarrow (x \cup y) \leq x$ (1)

(P2) $\Rightarrow x \leq (x \cup y)$ (2)

(1), (2), (P1, antisymétrie) $\Rightarrow x = x \cup y$

lemme P6

$\{\forall (x, y) \in E^2, (\neg(x \leq y))\} \Rightarrow \{(y < y \cup x) \wedge (y < y \nabla x)\}$

a) - supposons $\neg(x \leq y) \wedge y = y \cup x$

(A3) $\Rightarrow \neg(x \leq y) \wedge y = x \cup y$

(PDEF1) $\Rightarrow \neg(x \leq y) \wedge (x \leq y)$, absurdité, donc $y \neq y \cup x$ (1)

(P2) $\Rightarrow y \leq y \cup x$ (2)

(1), (2), (PDEF2) $\Rightarrow y < y \cup x$

b) - (1), (P4) $\Rightarrow y \neq y \nabla x$ (3)

(P2), (A7) $\Rightarrow y \leq y \cup x \leq y \nabla x$

(P1, transitivité) $\Rightarrow y \leq y \nabla x$ (4)

(3), (4), (PDEF2) $\Rightarrow y < y \nabla x$

lemme P7

a) $\forall x \in E \quad , \quad \{\emptyset \leq x\}$

b) $\forall x \in E \qquad \{x \neq \emptyset\} \Rightarrow \{\emptyset < x\}$

a) - (A5) $\Rightarrow (\emptyset \cup x) = x$

(PDEF1) $\Rightarrow \emptyset \leq x$

b) - $(\emptyset \leq x) \wedge (x \neq \emptyset) \Rightarrow \emptyset < x$ (PDEF2)

lemme P8

$\forall x \in E \quad , \quad \{x \neq \emptyset\} \Rightarrow \{\neg(x \leq 0)\}$

A contrario, supposons $(x \neq \emptyset) \wedge (x \leq \emptyset)$

(PDEF1) $\Rightarrow ((x \cup \emptyset) = \emptyset) \wedge (x \neq \emptyset)$

(A5) $\Rightarrow (x = \emptyset) \wedge (x \neq \emptyset)$, absurdité

lemme P9

$\forall x \in E \quad , \quad \{x \leq \emptyset \nabla x\}$

(A5) $\Rightarrow x = \emptyset \cup x$ (1)

(A7) $\Rightarrow \emptyset \cup x \leq \emptyset \nabla x$ (2)

(1), (2), (P1, transitivité) $\Rightarrow x \leq \emptyset \nabla x$

lemme P10

{l'opération $\cup$ est compatible avec la relation d'ordre partiel $\leq$ } $\overset{\text{def}}{\Leftrightarrow}$

$\{\forall (a, b, c, d) \in E^4 \mid (a \leq b) \wedge (c \leq d)\} \Rightarrow \{(a \cup c) \leq (b \cup d)\}$

$(a \leq b) \wedge (c \leq d)$

(PDEF1) $\Rightarrow ((a \cup b) = b) \wedge ((c \cup d) = d)$

$\Rightarrow (b \cup d) = (a \cup b) \cup (c \cup d)$

(A2) $\Rightarrow (b \cup d) = a \cup (b \cup (c \cup d))$

(A3) $\Rightarrow (b \cup d) = a \cup (b \cup (d \cup c))$

(A2) $\Rightarrow (b \cup d) = a \cup ((b \cup d) \cup c)$

(A3) $\Rightarrow (b \cup d) = a \cup (c \cup (b \cup d))$

(A2) $\Rightarrow (b \cup d) = (a \cup c) \cup (b \cup d)$

(PDEF1) $\Rightarrow (a \cup c) \leq (b \cup d)$

lemme P11

$\{A5\} \Rightarrow \{\{\exists\, x \in E \mid \forall y \in E \quad x \cup y = y\} \Rightarrow \{x = \emptyset\}\}$

Pour $y = \emptyset$, on a $x \cup \emptyset = \emptyset$

(A5) $\qquad x = \emptyset$

Ce lemme rappelle que l'élément neutre $\emptyset$ de E est unique, ce qui est un résultat connu.

lemme P12

$\{(A2), (A4), (A5)\} \Rightarrow$
$\{\forall (x, y) \in E^2 \mid (x \cup y) = \emptyset\} \Rightarrow \{(x = \emptyset) \wedge (y = \emptyset)\}$

$(x \cup y) = \emptyset$ (1)

(A5) $\Rightarrow x \cup \emptyset = x$

(1), (A5) $\Rightarrow (x \cup (x \cup y)) = x$ (2)

(1), (A4) $\Rightarrow ((x \cup x) \cup y) = \emptyset$

(A2) $\Rightarrow (x \cup (x \cup y)) = \emptyset$ (3)

(2), (3) $\Rightarrow x = \emptyset$ (4)

(1), (4) $\Rightarrow (\emptyset \cup y) = \emptyset$

(A5) $\Rightarrow y = \emptyset$

lemme P13

$\forall (x, y) \in E^2,$
$\{\neg (x \leq y)\} \Rightarrow \{\neg ((y \cup x) \leq y)\}$
$\Rightarrow \{\neg ((y \nabla x) \leq y)\}$

$\neg (x \leq y)$

(P6) $\Rightarrow y < (y \cup x)$

(A4) $\Rightarrow y < ((y \cup y) \cup x)$

(A2) $\Rightarrow y < (y \cup (y \cup x))$

(PDEF2) $\Rightarrow y \sharp y \cup (y \cup x)$

(A3) $\Rightarrow y \sharp (y \cup x) \cup y$

(PDEF1) $\Rightarrow \neg ((y \cup x) \leq y)$

Pour le deuxième volet du lemme, on a :

$$\neg(x \leq y)$$
$$(P6) \quad \Rightarrow y < (y \cup x)$$
$$(PDEF2) \Rightarrow y \sharp (y \cup x)$$
$$(P4) \quad \Rightarrow y \sharp (y \nabla x) \qquad (1)$$
$$\text{mais } (P3) \Rightarrow y \leq (y \nabla x)$$
$$(PDEF2) \Rightarrow y \cup (y \nabla x) = (y \nabla x) \qquad (2)$$
$$(1), (2) \Rightarrow y \sharp y \cup (y \nabla x)$$
$$(A3) \quad \Rightarrow y \sharp (y \nabla x) \cup y$$
$$(PDEF2) \quad \neg((y \nabla x) \leq y)$$

2.2 - Valeurs concrètes et valeurs abstraites -

Nous noterons V_c, l'ensemble des valeurs concrètes, définies dans le langage de programmation étudié. Ces valeurs sont des entiers, des booléens, des pointeurs, des tableaux .... Pour simplifier la présentation nous supposerons que les vérifications de types, (telles qu'elles sont faites actuellement dans des langages comme PASCAL), ont été menées à bien, avant de faire l'interprétation abstraite.

Nous noterons V_a l'ensemble des valeurs abstraites, qui pour simplifier, n'est pas structuré par la notion de type.

On introduit l'opération @, d'abstraction des valeurs concrètes, qui, à un ensemble de valeurs concrètes fait correspondre une valeur abstraite. Dans notre exemple introductif, on a vu le cas de l'abstraction d'un ensemble d'entiers :

$$X \subset \mathbb{N} \quad , \quad @(X) = \left[\underset{x \in X}{MIN(x)} \quad , \quad \underset{x \in X}{MAX(x)} \right]$$

L'opération γ, de concrétisation des valeurs abstraites, fait correspondre un ensemble de valeurs concrètes à une valeur abstraite. Dans notre exemple introductif, on a vu que :

$$\gamma([\alpha, \beta]) = \{\alpha, \alpha+1, \alpha+2, \ldots, \beta\}$$

Définition - DEF1 - @ : abstraction d'ensembles de valeurs concrètes

$$2^{V_c} \xrightarrow{@} V_a$$

Définition - DEF2 - $\bar{\cup}$: union de valeurs abstraites

$$V_a \times V_a \xrightarrow{\bar{\cup}} V_a$$

Hypothèse - H3

{@ est un homomorphisme de $(2^{V_c}, \cup)$ dans $(V_a, \bar{\cup})$}

$\overset{def}{\Leftrightarrow}\{\forall (v_1, v_2) \in (2^{V_c})^2, @(v_1 \cup v_2) = @(v_1) \bar{\cup} @(v_2)\}$

Définition - DEF4 - γ : concrétisation de valeurs abstraites

$$V_a \xrightarrow{\gamma} 2^{V_c}$$

Les fonctions @ et γ sont liées par les relations suivantes :

Hypothèse - H5

$$\{\forall v \in 2^{V_c} \quad , \quad \{v \subseteq \gamma(@(v))\}\}$$

Nous utiliserons également cette hypothèse sous la forme :

$$\{\forall v \in 2^{V_c} \quad , \quad x \in v\} \Rightarrow \{x \in \gamma(@(v))\}$$

Hypothèse - H6

$$\{\forall v \in V_a \quad , \quad \{v = @(\gamma(v))\}\}$$

Lemme - L7

a) - $\{@, \text{ est surjective}\} \overset{\text{def}}{\Longleftrightarrow} \{\forall y \in V_a, \exists x \subseteq V_c \mid y = @(x)\}$

b) - $\{\gamma \text{ est injective}\} \overset{\text{def}}{\Longleftrightarrow} \{\{\forall (x, y) \in V_a^2 \mid \gamma(x) = \gamma(y)\} \Rightarrow \{x = y\}\}$
$\overset{\text{def}}{\Longleftrightarrow} \{\{\forall (x, y) \in V_a^2 \mid x \neq y\} \Rightarrow \{\gamma(x) \neq \gamma(y)\}\}$

La fonction composée $V_a \xrightarrow{@\circ\gamma} V_a$ est bijective, à cause de l'hypothèse H6, d'où le lemme L7.

Noter que les hypothèses (H5) et (H6) sont toutes deux indépendantes l'une par rapport à l'autre. En voici la preuve par deux contre-exemples :

Ex1 : $V_c = \{x, y\}$, $V_a = \{\alpha\}$;

$\gamma(\alpha) = \{x\}$, $@(\emptyset) = \alpha$, $@(\{x\}) = \alpha$, $@(\{y\}) = \alpha$ et $@(\{x, y\}) = \alpha$.

On a $@(\gamma(\alpha)) = @(\{x\}) = \alpha$, mais $y \in \{y\}$ et $y \notin \gamma(@(\{y\})) = \gamma(\alpha) = \{x\}$

Ex2 : $V_c = \{x\}$, $V_a = \{\alpha, \beta\}$

$\gamma(\alpha) = \{x\}$, $\gamma(\beta) = \{x\}$, $@(\emptyset) = \alpha$, $@(x) = \alpha$, on a $x \in \{x\}$

et $x \in \gamma(@(\{x\})) = \gamma(\alpha) = \{x\}$, mais $@(\gamma(\beta)) = @(x) = \alpha \neq \beta$.

Lemme L8 — $\bar{\cup}$ admet un élément neutre

$\{\exists \square \in V_a \mid V_x \in V_a, x \bar{\cup} \square = \square \bar{\cup} x = x\}$

$\{\square = @(\emptyset)\}$

Lemme L9 — $\bar{\cup}$ est idempotente

$\{\forall x \in V_a \quad , \quad x \bar{\cup} x = x\}$

Lemme L10 — $\bar{\cup}$ est commutative

$\{\forall (x, y) \in V_a^2 \quad , \quad (x \bar{\cup} y) = (y \bar{\cup} x)\}$

Lemme L11 —— $\bar{\cup}$ est associative

$$\{\forall (x, y, z) \in V_a^3 \quad , \quad ((x \bar{\cup} y) \bar{\cup} z) = (x \bar{\cup} (y \bar{\cup} z))\}$$

Démonstrations :

L'ensemble 2^{V_c} de tous les sous-ensembles de l'ensemble V_c, avec la réunion $\cup$ comme loi de composition interne, et l'ensemble vide $\emptyset$ comme élément neutre est un monoïde abélien unitaire, noté $(2^{V_c}, \cup, \emptyset)$. L'image par @ $(V_a, \bar{\cup}, \square)$ du monoïde $(2^{V_c}, \cup, \emptyset)$ est donc un monoïde abélien unitaire, d'où L8, L9, L10, L11.

Définition - DEF 12 - Comparaison de valeurs abstraites

$$\forall (x, y) \in V_a^2 \quad , \quad \{x \bar{\leq} y\} \overset{\text{def}}{\Longleftrightarrow} \{x \bar{\cup} y = y\}$$

Définition - DEF 13

$$\{\forall (x, y) \in V_a^2 \quad , \quad \{x \bar{<} y\} \overset{\text{def}}{\Longleftrightarrow} \{\{x \bar{\leq} y\} \wedge \{x \neq y\}\}$$

Nous introduisons maintenant la notion d'élargissement de valeurs abstraites. Dans notre exemple introductif les valeurs abstraites étaient des intervalles d'entiers. On avait défini l'union d'intervalles par :

$$[a, b] \bar{\cup} [c, d] = [\underline{\min}(a, c), \underline{\max}(b, d)]$$

De même, l'élargissement de valeurs abstraites était défini par :

$$[a, b] \bar{\nabla} [c, d] = [\underline{\text{si}}\ c < a\ \underline{\text{alors}}\ -\infty\ \underline{\text{sinon}}\ a,$$
$$\underline{\text{si}}\ d > b\ \underline{\text{alors}}\ +\infty\ \underline{\text{sinon}}\ b]$$

Immédiatement : $\{[a, b] \leq [c, d]\} \Longleftrightarrow \{(a \geq c) \wedge (b \leq d)\}$

donc :

$$[a, b] \bar{\cup} [c, d] \bar{\leq} [a, b] \bar{\nabla} [c, d].$$

Cette propriété est générale pour les valeurs abstraites.

Remarque l'application $(2^{V_c}, \cup, \emptyset) \xrightarrow{@} (V_a, \bar{\cup}, \square)$ est isotone, c'est-à-dire :

$$(X \subseteq Y) \Rightarrow @(X) \bar{\leq} @(Y)$$

Définition - DEF 14 — $\bar{\nabla}$: élargissement de valeurs abstraites

$$V_a \times V_a \xrightarrow{\bar{\nabla}} V_a$$

Hypothèse - H14

$$\forall (x, y) \in V_a^2, \quad \{(x \mathbin{\bar{\cup}} y) \mathbin{\bar{\leq}} (x \mathbin{\bar{\nabla}} y)\}$$

Remarque : L'ensemble V_a muni des opérations $\bar{\cup}$ et $\bar{\nabla}$ satisfait les axiomes A1, A2, A3, A4, A5, A6, A7 d'après DEF2, L11, L10, L9, L8, DEF14, H14. On peut donc utiliser pour le triplet $(V_a, \bar{\cup}, \bar{\nabla})$ les lemmes P1, ..., P13. Nous utiliserons également les propriétés suivantes :

Lemme - L15

$$\{\forall x \in V_c, \forall (v_1, v_2) \in V_a^2 \mid x \in \gamma(v_1)\} \Rightarrow \{x \in \gamma(v_1 \mathbin{\bar{\cup}} v_2)\}$$

$x \in \gamma(v_1)$

$\Rightarrow x \in \gamma(v_1) \cup \gamma(v_2)$ (1)

(H5) $(\gamma(v_1) \cup \gamma(v_2)) \subseteq \gamma(@(\gamma(v_1) \cup \gamma(v_2)))$ (2)

(1), (2) $\Rightarrow x \in \gamma(@(\gamma(v_1) \cup \gamma(v_2)))$

(H3) $\Rightarrow x \in \gamma(@(\gamma(v_1)) \mathbin{\bar{\cup}} @(\gamma(v_2)))$

(H6) $\Rightarrow x \in \gamma(v_1 \mathbin{\bar{\cup}} v_2)$

Lemme - L16

$$\forall (v_1, v_2) \in V_a^2$$
$$\{(v_1 \mathbin{\bar{\leq}} v_2) \wedge (x \in \gamma(v_1))\} \Rightarrow \{x \in \gamma(v_2)\}$$

$v_1 \mathbin{\bar{\leq}} v_2 \Rightarrow v_2 = v_2 \mathbin{\bar{\cup}} v_1$ (DEF 12)

$x \in \gamma(v_1) \Rightarrow x \in \gamma(v_1 \mathbin{\bar{\cup}} v_2)$ (L15)

$\Rightarrow x \in \gamma(v_2 \mathbin{\bar{\cup}} v_1)$ (L10)

$\Rightarrow x \in \gamma(v_2)$

Hypothèse - H17

$\{\forall(v_1, \ldots, v_n, \ldots) \in V_a^\infty ,$

$\forall(s_o, \ldots, s_n, \ldots) \in V_a^\infty \mid$

$(s_o = \square, s_1 = s_o \bar{\nabla} v_1, \ldots, s_n = s_{n-1} \bar{\nabla} v_n, \ldots\}$

$\Rightarrow$ {la suite $s_o, s_1, \ldots, s_n, \ldots$ n'est pas infinie et strictement croissante pour $\bar{<}$ }

Dans notre exemple introductif, on pourrait avoir :

$\square < \square \bar{\nabla} [1, 3] = [1, 3]$

$< [1, 3] \bar{\nabla} [2, 5] = [1, +\infty]$

$< [1, +\infty] \bar{\nabla} [0, 1] = [-\infty, +\infty]$

et les termes de rang 4, 5, 6 ... ne sont plus strictement croissants.

2.3 - Contextes abstraits -

Nous noterons I, l'ensemble des identificateurs du programme étudié. (Nous supposerons que toutes les ambiguïtés syntaxiques, (découlant par exemple de la structure de bloc), ont été levées par des changements de noms appropriés). Le nombre d'identificateurs dans le programme est fini :

Hypothèse - (H100) - I : ensemble des identificateurs

I est un ensemble fini

Les contextes abstraits sont des ensembles de couples (i, v), où i est un identificateur et v une valeur abstraite. Ces couples différent deux à deux par leur identificateur. On notera $\mathcal{C}$, l'ensemble des contextes abstraits pour I donné.

Définition (DEF 101) - : ensemble des contextes abstraits

a) - $\mathcal{C} \subset 2^{I \times (V_a - \{\square\})}$

b) - $\mathcal{C}$ contient l'ensemble vide comme élément particulier que l'on appellera le contexte vide, noté Φ

c) - $\{\forall C \in \mathcal{C}, \forall(i, j) \in I^2, \forall(v, u) \in (V_a - \{\square\})^2,$

$\{(i, v) \in C, (j, u) \in C, (i, v) \neq (j, u)\} \Rightarrow \{i \neq j\}\}$

D'après l'hypothèse précédente, les couples d'un contexte différent deux à deux par leur identificateur ce qui permet d'introduire la notation suivante :

Définition (DEF 102)

la valeur abstraite d'un identificateur i dans un contexte C est notée C(i),

$C(i) \overset{def}{\Longleftrightarrow} \{$ si $(\exists v \in (V_a - \{\square\}) \mid (i, v) \in C)$ alors v

sinon $\square$

finsi ;$\}$

Soient v_1 et v_2 tel que $C(i) = v_1$ et $C(i) = v_2$,

1°) - $v_1 \neq \square$ et $v_2 \neq \square$

$\Rightarrow (i, v_1) \in C \wedge (i, v_2) \in C$

$\Rightarrow$ si $v_1 \neq v_2$, alors $(i, v_1) \neq (i, v_2) \Rightarrow i \neq i$, absurdité, donc $v_1 = v_2$.

2°) - ($v_1 \neq \square$ et $v_2 = \square$) ou ($v_1 = \square$ et $v_2 \neq \square$)

Ce cas est impossible, car si $v_1 \neq \square$, et $C(i) = v_1$ alors $(\exists v_1 \in (v_a - \{\square\}) \mid (i, v) \in C)$ est toujours vrai.

3°) - $v_1 = \square$ et $v_2 = \square$, dans ce cas $v_1 = v_2$.

Finalement, la valeur abstraite d'un identificateur i dans un contexte C est unique, ce qui permet de poser la définition de C(i).

Définition - (DEF 103) - $\bar{\bar{\cup}}$: union de contextes abstraits

$\forall (C_1, C_2) \in \mathcal{C}^2$

$\{C_1 \bar{\bar{\cup}} C_2\} \overset{def}{\Longleftrightarrow}$

$\{(i, v) \mid (i \in I) \wedge (v \in (V_a - \{\square\})) \wedge (v = C_1(i) \bar{\cup} C_2(i))\}$

Définition - (DEF 104) - $\bar{\bar{\nabla}}$: élargissement de contextes abstraits

$$\forall (C_1, C_2) \in \mathcal{C}^2$$
$$\{C_1 \bar{\bar{\nabla}} C_2\} \overset{\text{def}}{\Longleftrightarrow}$$
$$\{(i, v) \mid (i \in I) \wedge (v \in (V_a - \{\square\})) \wedge (v = C_1(i) \bar{\nabla} C_2(i))\}$$

Ces définitions étant posées, nous pouvons étudier la structure mathématique du triplet $(\mathcal{C}, \bar{\bar{\cup}}, \bar{\bar{\nabla}})$, sur laquelle repose la correction de notre algorithme d'interprétation abstraite.

Lemme - L 105

$$\mathcal{C} \times \mathcal{C} \xrightarrow{\bar{\bar{\cup}}} \mathcal{C}$$

Remarquons tout d'abord que $C_1 \bar{\bar{\cup}} C_2$ est définie pour C_1 et C_2 quelconques appartenant à $\mathcal{C}$. $C_1(i)$ et $C_2(i)$ sont des applications de I dans V_a, et

$V_a \times V_a \xrightarrow{\bar{\cup}} V_a$, donc $C_1(i) \bar{\cup} C_2(i)$ est toujours défini, à résultat élément de V_a. Si $v = C_1(i) \bar{\cup} C_2(i) = \square$, alors $(i, \square) \notin C_1 \bar{\bar{\cup}} C_2$, ce qui satisfait (DEF 101, a).

Il nous reste à montrer (DEF 101, c) c'est-à-dire que $v_1 = C_1(i) \bar{\cup} C_2(i)$ et $v_2 = C_1(i) \bar{\cup} C_2(i) \Rightarrow v_1 = v_2$, trivial.

Lemme - L 106

$$\forall (C_1, C_2) \in \mathcal{C}^2, \{C_1 \bar{\bar{\cup}} C_2 = \Phi\} \Rightarrow \{C_1 = \Phi \wedge C_2 = \Phi\}$$

a) - $C_1 \bar{\bar{\cup}} C_2 = \{(j, u) \mid (j \in I) \wedge (u \in V_a - \{\square\}) \wedge (u = C_1(j) \bar{\cup} C_2(j))\}$

donc $C_1 \bar{\bar{\cup}} C_2 = \Phi \Rightarrow \{\forall j \in I, \nexists u \in (V_a - \{\square\}) \mid u = C_1(j) \bar{\cup} C_2(j)\}$

$\Rightarrow \{\forall j \in I, C_1(j) \bar{\cup} C_2(j) \notin (V_a - \{\square\})\}$

mais d'après DEF2, $\{\forall j \in I, C_1(j) \bar{\cup} C_2(j) \in V_a\}$

par conséquent $\{\forall j \in I, C_1(j) \bar{\cup} C_2(j) = \square\}$

Comme $\bar{\cup}$ satisfait L11, L9, L8 donc (A2), (A4), (A5), elle satisfait le lemme P12, ce qui entraîne :

$$\{\forall j \in I, (C_1(j) = \square) \wedge (C_2(j) = \square)\}$$

D'après la définition DEF 102, on déduit :

$\{\forall j \in I, \nexists v \in (V_a - \{\square\}) \mid (j, v) \in C_1\}$

D'après DEF 101, on en déduit $C_1 = \Phi$ et de même $C_2 = \Phi$.

Lemme L107

$\forall (C_1, C_2) \in \mathcal{C}^2, \forall i \in I, \{(C_1 \bar{\bar{\cup}} C_2)(i) = C_1(i) \bar{\cup} C_2(i)\}$

$(C_1 \bar{\bar{\cup}} C_2)(i) =$ si $\exists v \in (V_a - \{\square\}) \mid (i, v) \in (C_1 \bar{\bar{\cup}} C_2)$ alors v sinon $\square$ finsi ;

et $(C_1 \bar{\bar{\cup}} C_2) = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C_1(i) \bar{\cup} C_2(i))\}$

1er cas : Pour i donné $(\exists v \in (V_a - \{\square\}) \mid (i, v) \in (C_1 \bar{\bar{\cup}} C_2))$ est vrai, entraîne

$(i, v) \in (C_1 \bar{\bar{\cup}} C_2) \wedge (i, C_1(i) \bar{\cup} C_2(i)) \in C_1 \bar{\bar{\cup}} C_2$

(DEF 101, c) $\Rightarrow v = (C_1 \bar{\bar{\cup}} C_2)(i) = C_1(i) \bar{\cup} C_2(i)$

2ème cas : Pour i donné $(\exists v \in (V_a - \{\square\}) \mid (i, v) \in (C_1 \bar{\bar{\cup}} C_2))$ est faux, donc

$\forall v \in (V_a - \{\square\}), (i, v) \notin C_1 \bar{\bar{\cup}} C_2$

$\Rightarrow \{\nexists u \in (V_a - \{\square\}) \mid u = C_1(i) \bar{\cup} C_2(i)\}$

mais $(C_1(i) \bar{\cup} C_2(i)) \in V_a$, donc $C_1(i) \bar{\cup} C_2(i) = \square$ et dans ce cas encore

$$(C_1 \bar{\bar{\cup}} C_2)(i) = \square = C_1(i) \bar{\cup} C_2(i)$$

Lemme L108

$\bar{\bar{\cup}}$ est 1) associative
2) commutative
3) idempotente
4) Φ est son élément neutre

1°) - DEF 103 $\Rightarrow C_1 \bar{\bar{\cup}} (C_2 \bar{\bar{\cup}} C_3) =$

$\{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C_1(i) \bar{\cup} (C_2 \bar{\bar{\cup}} C_3(i))\}$

DEF 103 $\Rightarrow (C_1 \bar{\bar{\cup}} C_2) \bar{\bar{\cup}} C_3 =$

$\{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = (C_1 \bar{\bar{\cup}} C_2)(i) \bar{\cup} C_3(i))\}$

il faut montrer $C_1(i) \bar{\cup} (C_2 \bar{\bar{\cup}} C_3)(i) \stackrel{?}{=} (C_1 \bar{\bar{\cup}} C_2)(i) \bar{\cup} C_3(i)$

(L107) $\Rightarrow$

$C_1(i) \bar{\cup} (C_2(i) \bar{\cup} C_3(i)) \stackrel{?}{=} (C_1(i) \bar{\cup} C_2(i)) \bar{\cup} C_3(i)$ vrai, car $\bar{\cup}$ est associative (L11).

2°) - $C_1 \bar{\bar{\cup}} C_2 = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C_1(i) \bar{\cup} C_2(i))\}$

commutativité de $\bar{\cup}$ (L10) $\Rightarrow$

$= \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C_2(i) \bar{\cup} C_1(i))\}$

par définition (DEF 103), on a :

$= C_1 \bar{\bar{\cup}} C_2$

3°) - $\forall C \in \mathcal{C}$

$C \bar{\bar{\cup}} C = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C(i) \bar{\cup} C(i))\}$

idempotence de $\bar{\cup}$ (L9) $\Rightarrow$

$C \bar{\bar{\cup}} C = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C(i))\}$

$\{(u = C(i)) \wedge (u \neq \square)\} \Leftrightarrow \{(i, u) \in C\}$, donc

$C \bar{\cup} C = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (i, u) \in C\}$

$= \{(i, u) \mid (i, u) \in C\}$

$= C$

4°) - $\forall C \in \mathcal{C}$,

$\Phi \bar{\bar{\cup}} C = \{(i, v) \mid (i \in I) \wedge (v \in (V_a - \{\square\})) \wedge (v = \Phi(i) \bar{\cup} C(i))\}$

Mais $\Phi(i) = \square$, $\forall i \in I$, donc

$\Phi \bar{\bar{\cup}} C = \{(i, v) \mid (i \in I) \wedge (v \in (V_a - \{\square\})) \wedge (v = \square \bar{\cup} C(i))\}$

$\square$ élément neutre de $\bar{\cup}$ (L8) $\Rightarrow$

$\Phi \bar{\bar{\cup}} C = \{(i, v) \mid (i \in I) \wedge (v \in (V_a - \{\square\})) \wedge (v = C(i))\}$

$(v = C(i)) \wedge (v \neq \square) \Leftrightarrow (i, v) \in C$

$\Phi \bar{\bar{\cup}} C = \{(i, v) \mid (i \in I) \wedge (v \in (V_a - \{\square\})) \wedge (i, v) \in C\}$

$\Phi \bar{\bar{\cup}} C = C$

Ce qui établit, que Φ est élément neutre à gauche, d'après la commutativité, il l'est également à droite.

Définition - DEF 109 - Comparaison de contextes abstraits

$$\forall (C_1, C_2) \in \mathcal{C}^2$$

$$\{C_1 \overline{\overline{\leq}} C_2\} \overset{\text{def}}{\Longleftrightarrow} \{C_1 \overline{\overline{\cup}} C_2 = C_2\}$$

Définition - DEF 110

$$\forall (C_1, C_2) \in \mathcal{C}^2$$

$$\{C_1 \overline{\overline{<}} C_2\} \overset{\text{def}}{\Longleftrightarrow} \{(C_1 \overline{\overline{\leq}} C_2) \wedge (C_1 \neq C_2)\}$$

Lemme - L111

$$\forall (C_1, C_2) \in \mathcal{C}^2$$

$$\{C_1 \overline{\overline{\leq}} C_2\} \Longleftrightarrow \{\forall i \in I,\ C_1(i) \overline{\leq} C_2(i)\}$$

a) - $\{C_1 \overline{\overline{\leq}} C_2\} \Rightarrow \{\forall i \in I,\ C_1(i) \overline{\leq} C_2(i)\}$

DEF 109 $\Rightarrow$ $C_1 \overline{\overline{\leq}} C_2 = (C_1 \overline{\overline{\cup}} C_2) = C_2$

$\forall i \in I,\ (C_1 \overline{\overline{\cup}} C_2)(i) = C_2(i)$

L 107 $\Rightarrow$ $\forall i \in I,\ C_1(i) \overline{\cup} C_2(i) = C_2(i)$

DEF 12 $\Rightarrow$ $\forall i \in I,\ C_1(i) \overline{\leq} C_2(i)$

b) - réciproque.

DEF 103 $\Rightarrow$ $(C_1 \overline{\overline{\cup}} C_2) = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C_1(i) \overline{\cup} C_2(i))\}$

DEF 12 $\Rightarrow$ $C_1(i) \overline{\cup} C_2(i) = C_2(i)$ car $C_1(i) \overline{\leq} C_2(i)$, $\forall i \in I$

donc $(C_1 \overline{\overline{\cup}} C_2) = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C_2(i))\}$

DEF 102 $\Rightarrow$ $= \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (i, u) \in C_2\}$

$= C_2$

Ce lemme fournit un algorithme pour l'opération d'inclusion de contextes : pour tous les identificateurs i de C_1, vérifier que $C_1(i) \bar{\leq} C_2(i)$.

Lemme - L112

$\mathcal{C} \times \mathcal{C} \xrightarrow{\bar{\bar{\nabla}}} \mathcal{C}$

D'après DEF 104, $C_1 \bar{\bar{\nabla}} C_2$ est définie pour C_1 et C_2 quelconques appartenant à $\mathcal{C}$, on a :

$(C_1 \bar{\bar{\nabla}} C_2) = \{(i, v) \mid (i \in I) \wedge (v \in (V_a - \{\square\})) \wedge (v = C_1(i) \bar{\nabla} C_2(i))\}$

Dans cette identité, $C_1(i)$ et $C_2(i)$ sont des applications de I dans V_a, et $V_a \times V_a \xrightarrow{\bar{\nabla}} V_a$, donc $C_1(i) \bar{\nabla} C_2(i)$ est toujours défini, à résultat élément de V_a, éventuellement $\square$. Mais si $v = C_1(i) \bar{\cup} C_2(i) = \square$, alors $(i,\square) \notin C_1 \bar{\bar{\nabla}} C_2$, ce qui satisfait (DEF 101, a)

Il reste à montrer (DEF 101, c), c'est-à-dire que $v_1 = C_1(i) \bar{\nabla} C_2(i)$ et $v_2 = C_1(i) \bar{\nabla} C_2(i) \Rightarrow v_1 = v_2$, trivial.

Lemme - L113

$\forall (C_1, C_2) \in \mathcal{C}^2, \forall i \in I, \{(C_1 \bar{\bar{\nabla}} C_2)(i) = C_1(i) \bar{\nabla} C_2(i)\}$

DEF 102 $\Rightarrow$ $(C_1 \bar{\bar{\nabla}} C_2)(i) =$ si $(\exists v \in (V_a - \{\square\}) \mid (i, v) \in C_1 \bar{\bar{\nabla}} C_2)$ alors v sinon $\square$ finsi ;

DEF 104 $\Rightarrow$ $(C_1 \bar{\bar{\nabla}} C_2) = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge (u = C_1(i) \bar{\nabla} C_2(i))\}$

1er cas : Pour i donné $(\exists v \in (V_a - \{\square\}) \mid (i, v) \in C_1 \bar{\bar{\nabla}} C_2)$ est vrai, entraîne

$\{\exists u \in (V_a - \{\square\})) \mid (u = C_1(i) \bar{\nabla} C_2(i)) \wedge (u = (C_1 \bar{\bar{\nabla}} C_2)(i))\}$

donc $(C_1 \bar{\bar{\nabla}} C_2)(i) = u = C_1(i) \bar{\nabla} C_2(i)$

2ème cas : Pour i donné, $(\exists v \in (V_a - \{\square\}) \mid (i, v) \in (C_1 \bar{\bar{\nabla}} C_2))$ est faux, entraîne

$(C_1 \bar{\bar{\nabla}} C_2)(i) = \square$ et $\{\forall v \in (V_a - \{\square\}), (i, v) \notin (C_1 \bar{\bar{\nabla}} C_2)\}$

donc $\{\nexists u \in (V_a - \{\square\}) \mid u = C_1(i) \bar{\nabla} C_2(i)\}$, mais $C_1(i) \bar{\nabla} C_2(i) \in V_a$ donc

$C_1(i) \bar{\nabla} C_2(i) = \square$, donc à nouveau

$$(C_1 \bar{\bar{\nabla}} C_2)(i) = \square = C_1(i) \bar{\nabla} C_2(i)$$

Lemme - L114

$\forall (x, y) \in \mathcal{C}^2, \{x \bar{\bar{\cup}} (x \bar{\bar{\nabla}} y) = (x \bar{\bar{\nabla}} y)\}$

$\wedge \{(x \bar{\bar{\nabla}} y) \bar{\bar{\cup}} y = (x \bar{\bar{\nabla}} y)\}$

$x \bar{\bar{\cup}} (x \bar{\bar{\nabla}} y) \overset{\text{DEF}}{=} {}^{103} \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge$

$u = x(i) \bar{\cup} (x \bar{\bar{\nabla}} y)(i)\}$

L 113 $\Longrightarrow$

$x \bar{\bar{\cup}} (x \bar{\bar{\nabla}} y) = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge$

$u = x(i) \bar{\cup} (x(i) \bar{\nabla} y(i))\}$

D'après le lemme P3, $x(i) \bar{\leq} x(i) \bar{\nabla} y(i)$, donc

(DEF 12) $x(i) \bar{\cup} (x(i) \bar{\nabla} y(i)) = (x(i) \bar{\nabla} y(i))$

$x \bar{\bar{\cup}} (x \bar{\bar{\nabla}} y) = \{(i, u) \mid (i \in I) \wedge (u \in (V_a - \{\square\})) \wedge u = x(i) \bar{\nabla} y(i)\}$

(DEF 104) $\Rightarrow x \bar{\bar{\cup}} (x \bar{\bar{\nabla}} y) = x \bar{\bar{\nabla}} y$

La deuxième partie du théorème se démontre comme la première, en utilisant le fait (P3) que :

$$(x(i) \bar{\nabla} y(i)) \bar{\cup} y(i) = x(i) \bar{\nabla} y(i)$$

Lemme - L115

$\forall (C_1, C_2) \in \mathfrak{C}^2 \quad (C_1 \bar{\cup} C_2) \bar{\leq} (C_1 \bar{\nabla} C_2)$

$$
\begin{aligned}
C_1 \bar{\nabla} C_2 &= (C_1 \bar{\nabla} C_2) \bar{\cup} C_2 && \text{(L114)}\\
&= (C_1 \bar{\cup} (C_1 \bar{\nabla} C_2)) \bar{\cup} C_2 && \text{(L114)}\\
&= (C_2 \bar{\cup} (C_1 \bar{\cup} (C_1 \bar{\nabla} C_2)) && \text{(L108, 2)}\\
&= (C_2 \bar{\cup} C_1) \bar{\cup} (C_1 \bar{\nabla} C_2) && \text{(L108, 1)}\\
&= (C_1 \bar{\cup} C_2) \bar{\cup} (C_1 \bar{\nabla} C_2) && \text{(L108, 2)}\\
\Rightarrow\ (C_1 \bar{\cup} C_2) &\bar{\leq} (C_1 \bar{\nabla} C_2), && \text{(DEF 109)}
\end{aligned}
$$

Remarque : L'ensemble $\mathfrak{C}$, muni des opérations $\bar{\cup}$ et $\bar{\nabla}$ satisfait les axiomes (A1), (A2), (A3), (A4), (A5), (A6), (A7) d'après (L105), (L108, 1), (L108, 2), (L108, 3), (L108, 4), (L112) et (L115). On pourra donc utiliser pour le triplet $(\mathfrak{C}, \bar{\cup}, \bar{\nabla})$ les lemmes P1 à P13. En plus, nous utiliserons les propriétés suivantes :

Lemme - L116

$\forall (C_1, C_2) \in \mathfrak{C}^2 \quad \{C_1 = C_2\} \Longleftrightarrow \{\forall i \in I,\ C_1(i) = C_2(i)\}$

$$
\begin{aligned}
\{C_1 = C_2\} &\overset{\text{(L108, 3)}}{\Longleftrightarrow} \{(C_1 = C_1 \bar{\cup} C_2) \wedge (C_2 = C_1 \bar{\cup} C_2)\}\\
&\overset{\text{DEF 109}}{\Longleftrightarrow} \{(C_1 \bar{\leq} C_2) \wedge (C_2 \bar{\leq} C_1)\}\\
&\overset{\text{L111}}{\Longleftrightarrow} \{\forall i \in I,\ (C_1(i) \bar{\leq} C_2(i)) \wedge (C_2(i) \bar{\leq} C_1(i))\}\\
&\overset{\text{DEF 12}}{\Longleftrightarrow} \{\forall i \in I,\ (C_1(i) = C_1(i) \bar{\cup} C_2(i))\\
&\qquad \wedge (C_2(i) = C_1(i) \bar{\cup} C_2(i))\}\\
&\Longleftrightarrow \{\forall i \in I,\ C_1(i) = C_2(i)\}
\end{aligned}
$$

Lemme - L117

$\{\forall (C_1, \ldots, C_n, \ldots) \in \mathcal{C}^\infty$

$\forall (S_0, \ldots, S_n, \ldots) \in \mathcal{C}^\infty \mid$

$(S_0 = \Phi$

$S_1 = S_0 \bar{\bar{\vee}} C_1$

$\ldots$

$S_N = S_{n-1} \bar{\bar{\vee}} C_n\}$

$\Rightarrow$ {la suite $S_0, S_1, \ldots, S_n$ n'est pas infinie et strictrement croissante pour $\bar{\bar{<}}$}

Montrons d'abord la proposition préliminaire 1 suivante :

$$\forall (C_1, C_2) \in \mathcal{C}^2$$

$$\{C_1 \bar{\bar{\leq}} C_2\} \Rightarrow \{\exists i \in I \mid C_1(i) \bar{<} C_2(i)\}$$

Par l'absurde supposons que la conclusion soit fausse

$$\{\exists i \in I \mid C_1(i) \bar{<} C_2(i)\}$$

$\Rightarrow$ $\{\forall i \in I, \neg(C_1(i) \bar{<} C_2(i))\}$

DEF 13

$\Rightarrow$ $\{\forall i \in I, \neg(C_1(i) \bar{\leq} C_2(i) \wedge C_1(i) \neq C_2(i))\}$

$\Rightarrow$ $\{\forall i \in I, \neg(C_1(i) \bar{\leq} C_2(i)) \vee (C_1(i) = C_2(i))\}$ (1)

(L 111)

$\Rightarrow$ $C_1 \bar{\bar{<}} C_2 \Rightarrow C_1 \bar{\bar{\leq}} C_2 \Rightarrow \{\forall i \in I, C_1(i) \bar{\leq} C_2(i)\}$ (2)

(1), (2) $\Rightarrow$ $\{\forall i \in I, (C_1(i) \bar{\leq} C_2(i))$

$\wedge [\neg (C_1(i) \bar{\leq} C_2(i)) \vee (C_1(i) = C_2(i))\}$

$\Rightarrow$ $\{\forall i \in I, (C_1(i) \bar{\leq} C_2(i)) \wedge (C_1(i) = C_2(i))\}$

$\Rightarrow$ $\{\forall i \in I, C_1(i) = C_2(i)\}$

(L116)

$\Rightarrow$ $C_1 = C_2$, contraire à $C_1 \neq C_2$.

Si maintenant la suite $S_0, S_1, \ldots, S_n, \ldots$ est infinie strictement croissante, on peut appliquer un nombre infini de fois la proposition préliminaire 1. Comme le nombre d'identificateurs sur lesquels porte cette proposition préliminaire 1 est fini (H 100), c'est qu'elle porte un nombre infini de fois sur au moins un identificateur, disons j. Pour cet identificateur, on a une séquence infinie de la forme :

$$(3)\ \square \bar{\leq} \ldots \bar{\leq} C_{k_1}(j) \bar{<} C_{k_1+1}(j) \bar{\leq} \ldots \bar{\leq} C_{k_2}(j) \bar{<} C_{k_2+1}(j) \bar{\leq} \ldots$$

$$\ldots \bar{\leq} C_{k_n}(j) \bar{<} C_{k_n+1}(j) \bar{\leq} \ldots$$

En fait $\quad C_{k_\alpha + k}(j) \bar{\leq} C_{k_\alpha+k+1}(j)$

et $\quad \neg(C_{k_\alpha+k}(j) \bar{<} C_{k_\alpha+k+1}(j))$

$C_{k_\alpha + k}(j) = C_{k_\alpha+k+1}(j)$, la séquence est donc en fait de la forme :

$$\square \bar{\leq} \ldots \bar{\leq} C_{k_1}(j) \bar{<} C_{k_1+1}(j) = \ldots = C_{k_1+\alpha_1}(j) = C_{k_2}(j) < C_{k_2+1}(j) =$$

$$\ldots = C_{k_{n-1} + \alpha_{n-1}}(j) = C_{k_n}(j) \bar{<} C_{k_n+1}(j) = \ldots$$

Mais $\quad C_{k_n+1}(j) = C_{k_n}(j) \bar{\nabla} V_{k_m+1}$

$$= C_{k_{m-1} + \alpha_{n-1}}(j) \bar{\nabla} V_{k_m+1}$$

$$\ldots$$

$$= C_{k_{n-1}+1} \bar{\nabla} V_{k_m+1}$$

Donc la séquence infinie est de la forme :

$$\square \bar{<} \ldots \bar{<} C_{k_1}(j) \bar{<} C_{k_2}(j) \bar{<} \ldots \bar{<} C_{k_2}(j) \bar{<} \ldots$$

avec $C_{k_2}(j) = C_{k_1}(j) \bar{\nabla} V_{k_2}, \ldots, C_{k_n}(j) = C_{k_{n-1}}(j) \bar{\nabla} V_{k_n}$

Contraire à l'hypothèse (H 17).

2.4 - Graphes de programmes -

2.4.1 - Propriétés du graphe -

Soit N l'ensemble des noeuds du graphe de programme :

Hypothèse - H 200

N est un ensemble fini

On peut distinguer dans N, l'ensemble N_a des noeuds affectation, N_t des noeuds tests ou décisions, N_j des noeuds de jonction, N_s des noeuds de sortie et du noeud n_e d'entrée.

Hypothèse - H 201

$(N_a, N_t, N_j, N_s, \{n_e\})$ est une partition de N

Soit A l'ensemble des arcs du graphe.

Hypothèse - H 202

$A \subseteq N \times N$

Définitions - D 203

$\forall n \in N$

$\underline{\text{arc-sortie}}(n) \stackrel{\text{def}}{=} \{(n, n') \mid (n, n') \in A\}$

$\underline{\text{arc-entrée}}(n) \stackrel{\text{def}}{=} \{(n', n) \mid (n', n) \in A\}$

Notons Card(α), la cardinalité d'un ensemble α.

Hypothèse - H 204

1) - $\forall n \in N_a$, Card(arc sortie(n)) = Card(arc entrée(n)) = 1

2) - $\forall n \in N_t$, Card(arc sortie(n)) = 2, Card(arc entrée(n)) = 1

3) - $\forall n \in N_j$, Card(arc sortie(n)) = 1

4) - $\forall n \in N_s$, Card(arc sortie(n)) = 0, Card(arc entrée(n)) = 1

5) - Card(arc sortie(n_e)) = 1, Card(arc entrée(n_e)) = 0

Définition - DEF 205

Soit G = (N, A), il existe un arc d'entrée de G unique, noté arc initial (G)

D'après (H 204, 5), on a arc initial (G) = successeur (n_e)

Définition - DEF 206 - chemin dans un graphe

Un "chemin" de α à β dans G = (N, A) pour α et $\beta \in A$, est une séquence d'arcs (a_1, ..., a_n) de A, tels que $a_1[1] = \alpha$, $a_n[2] = \beta$ et $\forall_j \in [1, n[, a_j[2] = a_{j+1}[1]$.

Remarque : $\forall k \in [1, n]$, $a_k = (n_k, n'_k)$ et l'on note $a_k[1] = n_k$, $a_k[2] = n'_k$.

Hypothèse - H 207

{Il y a un chemin du noeud d'entrée à tout autre noeud du graphe de programme} $\Leftrightarrow$

$\{\forall a \in A, \exists (a_1, \ldots, a_n) \in \underline{\text{chemins}}\ (G) \mid$

$(a_1 = \underline{\text{arc initial}}\ (G) \wedge (a_n = a)\}$

Définition - DEF 208

{Un chemin $(a_1, \ldots, a_n)$ de G est un cycle}

$\Leftrightarrow \{(a_1, \ldots, a_n) \in \underline{\text{chemins}}(G) \wedge a_1 = a_n\}$

Lemme - L 209

{tout cycle de G contient au moins un noeud de jonction}

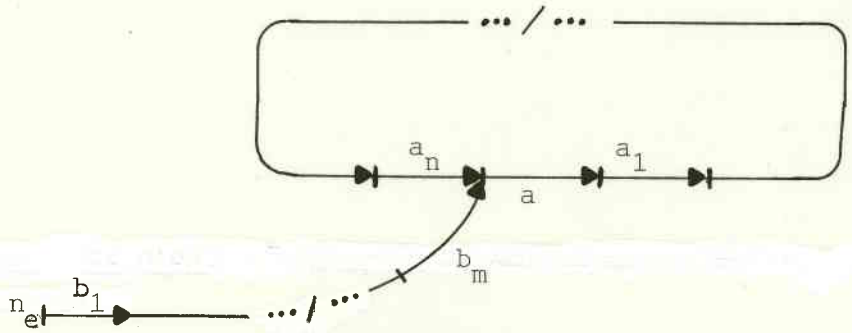
$\Leftrightarrow \{\forall (a_1, \ldots, a_n) \in \underline{\text{cycles}}(G), \exists k \in [1, n] \mid a_k[1] \in N_{j_s} \cup N_{j_b}\}$

- Soit un cycle $(a, a_1, \ldots, a_n, a)$ de G. D'après (D 203),
$a_n \in \underline{\text{arc d'entrée}}\ (a\ [1])$, car $(a_n[1], a_n[2]) = (a_n[1], a[1]) \in A$ (DEF 206)

- D'autre part d'après (H 207), il y a un chemin ($\underline{\text{arc initial}}\ (G)$, b_1, ... b_n, a) de G.

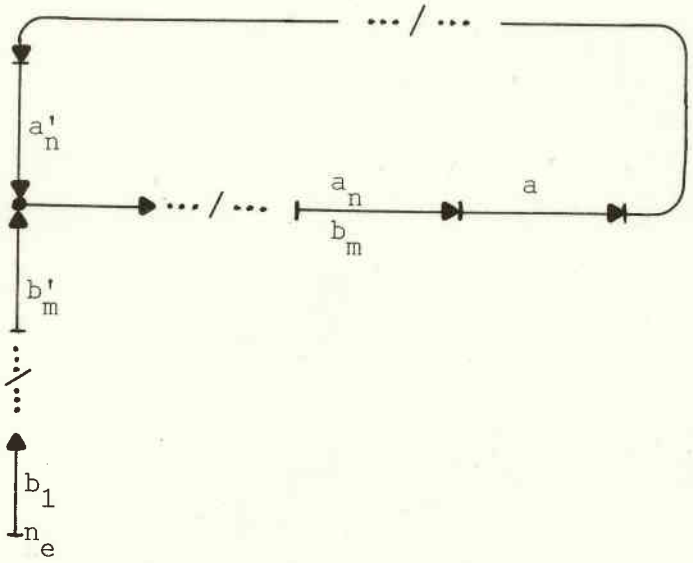
Comme précédemment, $b_n \in \underline{\text{arc d'entrée}}\ (a[1])$

1) - $\underline{\text{si}}\ a_n \neq b_n$, que l'on peut figurer comme suit :



$\underline{\text{card}}\ (\underline{\text{arc d'entrée}}\ (a[1])) \geq 2$, donc d'après (H 204), a[1] est un noeud de jonction.

2) - si $a_n = b_m$



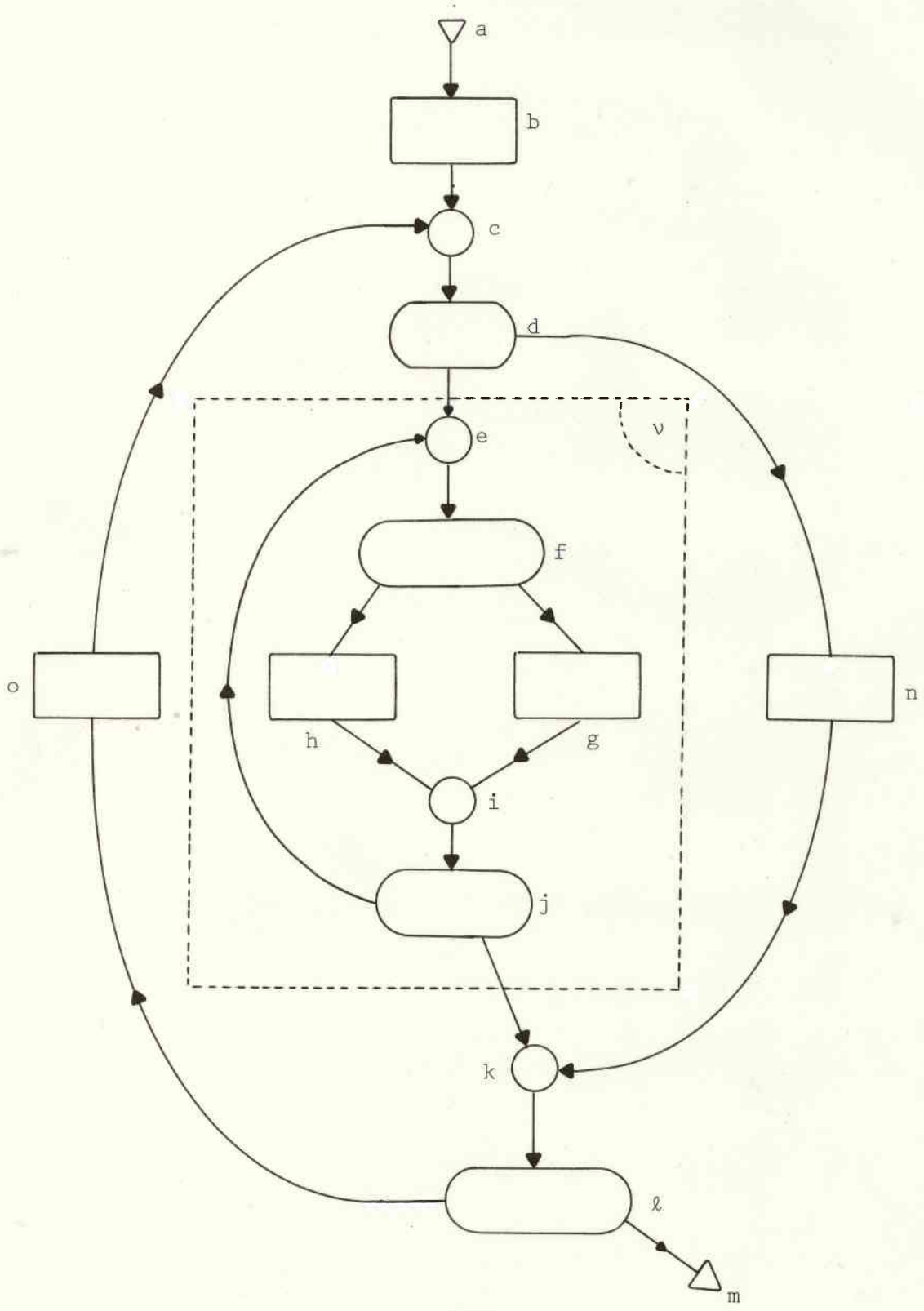
il existe n' et m' uniques, tels que $a_{n'} \neq b_{m'}$ et $a_{n'+1} = b_{m'+1}$, ..., $a_n = b_m$ (car $b_1 \neq a, a_1, \ldots, a_n$ puisque n_e est le noeud d'entrée). Dans ces conditions, comme précédemment $a_{n'}$,[2] est nécessairement un noeud de jonction.

Lemme - L 210

Il existe une partition (N_{j_s}, N_{j_b}) de N_j, telle que tout cycle de G contient au moins un noeud de N_{j_b}.

Les noeuds de N_{j_s}, sont dits de "jonction simple" et ceux de N_{j_b}, de "jonction de boucle". Les noeuds de jonction de boucle, sont des points d'élargissement des contextes abstraits, pour assurer la convergence de l'algorithme d'interprétation abstraite.

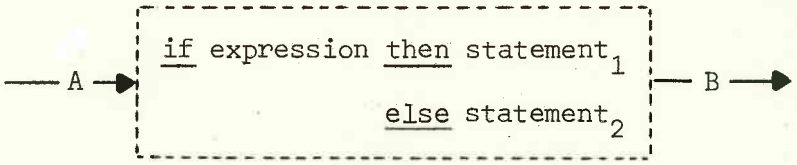
Considérons le graphe de programme suivant :

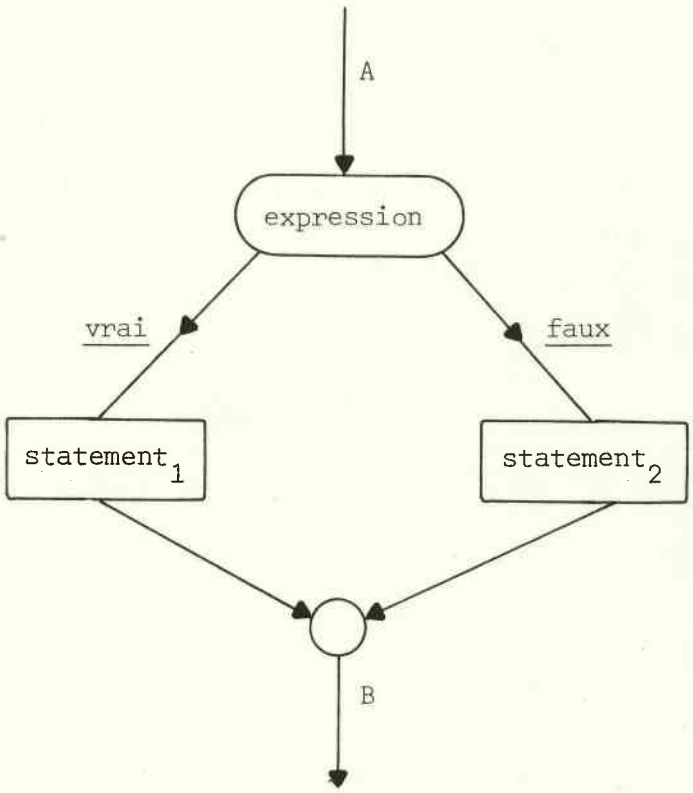


Intuitivement on voit que le noeud de jonction i exprime une alternative entre deux calculs h ou g à la suite du test f. Au contraire, le noeud de jonction e, correspond à une boucle ou calcul itératif au travers de e f h i j e, ou e f g i j e. De même, le noeud k représente la réunion des deux alternatives

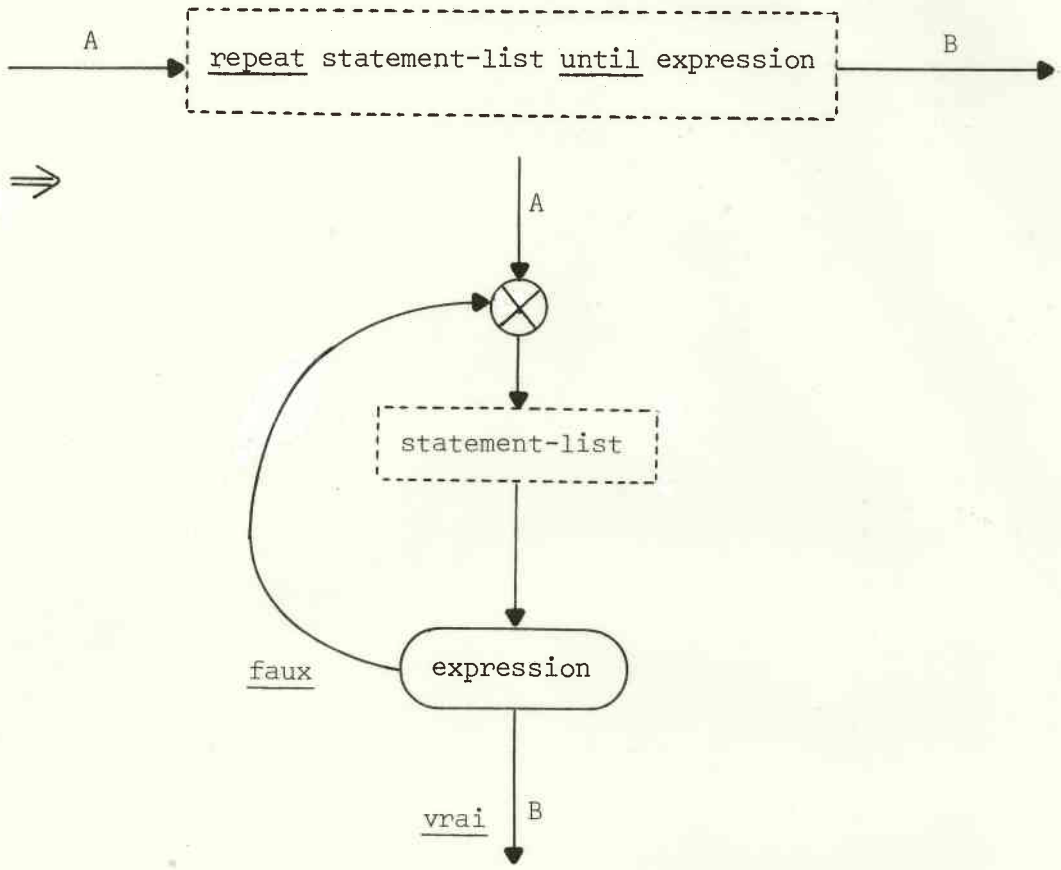
de calcul entre n et le bloc ν à la suite du test d, tandis que le noeud c, ferme les boucles c d n k ℓ σ c, ou c d ν k ℓ σ c.

Dans un langage comme PASCAL, qui possède des instructions de contrôle de haut niveau, on peut faire la distinction entre noeuds de jonctions simples ou de boucle dans l'algorithme qui construit le graphe de programme à partir du texte du programme. Pour une instruction conditionnelle, on aurait un noeud de jonction simple :





Tandis que pour une instruction répétitive, on aurait un noeud de jonction de boucle :



Malheureusement, ce type de macro-expansion qui permet de distinguer d'après la définition du langage entre noeuds de jonction simple ou de boucle n'est pas applicable quand ce langage contient des étiquettes et des constructions de branchement inconditionnel. Dans ce cas, la distinction des utilisations des étiquettes comme jonctions simples ou de boucles, doit être faite pour chaque programme, sur la base d'une étude du graphe de ce programme.

Avant de présenter cette étude du graphe, on rappelle la définition des hypergraphes [11] :

- Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble fini, et $\mathcal{E} = (E_i \mid i \in I)$ une famille de parties de X. On dit [11, page 374] que $\mathcal{E}$ constitue un <u>hypergraphe</u> sur X, si l'on a :

(1) $\quad E_i \neq \emptyset \quad (i \in I)$

(2) $\quad \bigcup_{i \in I} E_i = X$

- $E_1, E_2, \ldots, E_m$ sont les <u>arêtes</u> de l'hypergraphe.
- Un <u>ensemble transversal</u> des arêtes d'un hypergraphe $H = (X ; E_1, E_2, \ldots, E_m)$ est un ensemble $T \subset X$ avec $T \cap E_i \neq \emptyset$ $(i = 1, 2, \ldots, m)$.
- Si $\mathcal{A} = (A_i \mid i \in I)$ et $\mathcal{B} = (B_j \mid j \in J)$ sont deux familles de sous-ensembles de $X = \{x_1, x_2, \ldots, x_n\}$, on pose :

$$\mathcal{A} \vee \mathcal{B} = (A_i \cup B_j \mid (i, j) \in I \times J)$$

- On désigne par $\ddot{\mathcal{A}}$ ("grille" de $\mathcal{A}$), la famille des ensembles traversaux d'une famille $\mathcal{A}$.
- On désigne par <u>Min</u> $\mathcal{A}$, la famille des ensembles minimaux (par rapport à l'inclusion) de $\mathcal{A}$.
- On note <u>Tr</u> $\mathcal{A}$, la famille des ensembles minimaux de la grille $\ddot{\mathcal{A}}$.

BERGE [11, page 405] démontre que l'on peut déterminer <u>Tr</u> $\mathcal{A}$ par un algorithme direct :

```
procédure Tr A , A famille de sous-ensembles de X ;
début
    procédure Tr {A}, A ensemble d'éléments de X ;
    début
        Tr{A} := ({a} | a ∈ A) ;
    fin ;
    {A1, A2, ..., Ak} := Min A ;
    Tr A := Tr {A1} ;
    pour i de 2 à k faire
        Tr A := Min (Tr A ∨ Tr {Ai}) ;
    refaire ;
fin ;
```

Exemple :

$$\mathcal{A} = \{\{c, k\}, \{c, e, i, k\}, \{e, i\}\}$$

$$\{A_1, A_2\} = \{\{c, k\}, \{e, i\}\} = \underline{\text{Min}}\ \mathcal{A}$$

$$\underline{\text{Tr}}_1\ \mathcal{A} = \underline{\text{Tr}}\ \{A_1\} = \{\{c\}, \{k\}\}$$

$$\underline{\text{Tr}}\ \{A_2\} = \{\{e\}, \{i\}\}$$

$$\underline{\text{Tr}}_2\ \mathcal{A} = \underline{\text{Min}}\ (\underline{\text{Tr}}_1\ \mathcal{A} \vee \underline{\text{Tr}}\ \{A_2\})$$

$$= \underline{\text{Min}}\ (\{c, e\}, \{c, i\}, \{k, e\}, \{k, i\})$$

$$= (\{c, e\}, \{c, i\}, \{k, e\}, \{k, i\})$$

$$\underline{\text{Tr}}\ \mathcal{A} = \text{Tr}_2\ \mathcal{A}$$

On peut donner une forme booléenne à cet algorithme [12].

On détermine l'ensemble des noeuds de jonction comme suit :

1 - Recherche de l'ensemble Γ des circuits γ_i ($i \in [1, n]$) (élémentaires) du graphe :

$\Gamma = \{(c, d, n, k, \ell, \sigma, c),$
$(c, d, e, f, h, i, j, k, \ell, \sigma, c),$
$(c, d, e, f, g, i, j, k, \ell, \sigma, c),$
$(e, f, h, i, j, e),$
$(e, f, g, i, j, e)\}$

2 - Construction de l'ensemble $\mathcal{A}$ obtenu en associant à chaque cycle γ_i de Γ, l'ensemble $\bar{\gamma}_j$ ($j \in [1, m]$) des noeuds de jonction de ce cycle :

$$\mathcal{A} = \{\{c, k\}, \{c, e, i, k\}, \{e, i\}\}$$

(Notons, en ce pas, que tout circuit élémentaire ou non dans le graphe passe par un ensemble de noeuds de jonction, qui est un élément de $\mathcal{A}$.

D'autre part, tout cycle contenant au moins un noeud de jonction (d'après L 209), $\mathcal{A}$ constitue un hypergraphe sur $\bigcup_{j \in [1, m]} \bar{\gamma}_j$)

3 - Recherche de l'ensemble Tr $\mathcal{A}$ des ensembles minimaux de la grille de $\mathcal{A}$.

$$\underline{\text{Tr}}\,\mathcal{A} = \{\{c, e\}, \{c, i\}, \{k, e\}, \{k, i\}\}$$

(Notons, en ce pas, que tout circuit élémentaire ou non, dans le graphe de programme, contient au moins un noeud de jonction appartenant à l'un quelconque des ensembles de la famille $\underline{\text{Tr}}\,\mathcal{A}$).

4 - Choix heuristique d'un ensemble transversal N_{j_b} de $\mathcal{A}$ parmi les éléments de la famille $\underline{\text{Tr}}\,\mathcal{A}$. On pose $N_{j_s} = N_j - N_{j_b}$.

(Noter, que le lemme L 210 est satisfait quel que soit ce choix dans $\underline{\text{Tr}}\,\mathcal{A}$).

Pour faire le choix, diverses heuristiques peuvent être retenues :

- Choisir les ensembles de $\underline{\text{Tr}}\,\mathcal{A}$ contenant le nombre maximum de "têtes d'intervalles" au sens de ALLEN et COCKE [7]. En effet, étant donné un noeud h, l'intervalle I(h) est le sous-graphe maximal à une seule entrée, pour lequel h est le seul noeud d'entrée et pour lequel tous ses circuits contiennent h. Intuitivement, h est la tête d'un certain nombre de boucles dans I(h), ce qui justifie cette heuristique.

Dans notre exemple, les têtes d'intervalles sont a, c, e, k et

$$\underline{\text{Tr}}\,\mathcal{A} = \{\{c, e\}, \{c, i\}, \{k, e\}, \{k, i\}\}$$

On retient les ensembles qui contiennent un nombre maximum de noeuds têtes d'intervalles, soit :

$$\{\{c, e\}, \{k, e\}\}$$

- Les boucles correspondant généralement à un retour arrière dans le texte du programme, on peut songer à retenir l'ensemble des noeuds "les plus près du noeud d'entrée". Pour cela on affecte un poids à chaque noeud, égal à sa distance minimale (mesurée en nombre d'arcs) au noeud d'entrée. On choisit l'ensemble ayant un poids minimal, ce poids étant la somme des poids de ses éléments :

$$\{\{c, e\}, \{k, e\}\}, \text{ on choisit } N_{j_b} = \{c, e\}$$

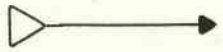
$$\begin{array}{cccc} \downarrow & \downarrow & \downarrow & \downarrow \\ 2 & 4 & 7 & 4 \\ \underbrace{} & & \underbrace{} & \\ 6 & & 11 & \end{array}$$

- On peut imaginer d'autres critères, mais la théorie de l'algorithme n'en dépend pas.

2.4.2 - Propriétés des noeuds du graphe -

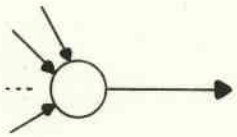
Nous représenterons graphiquement des noeuds au graphe comme suit :

a) - noeud d'entrée, n_e ;

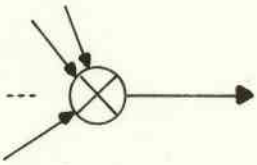


b) - noeuds de sortie N_s ;

c) - noeuds de jonction simple N_{j_s} ;
ces noeuds permettent de réunir
les alternatives d'une instruction si ou cas, ...



d) - noeuds de jonction de boucle N_{j_b}
ces noeuds permettent de fermer
les boucles, ...



e) - noeuds affectation N_a.

Hypothèse - H 212

$\{\forall n \in N_a \mid (n = \boxed{v := \bar{f}(v_1, \ldots, v_m)}) \wedge$

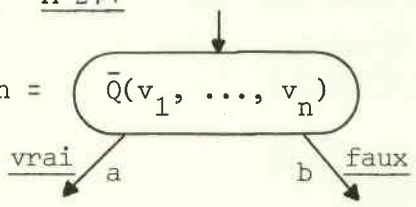
$(v_1, \ldots, v_n, v) \in I^{n+1}$, $\bar{f} \in E$, ensemble des dénotations d'expressions du langage$\} \Rightarrow$

1) $\{(V_c - \gamma(\square))^m \xrightarrow{f} V_c$, où f est la fonction dénotée par $\bar{f}$

2) {l'exécution du noeud n, ne modifie que la variable v (pas d'effets de bord)}

f) - noeuds test ou décisions N_t

Hypothèse - H 214

{$\forall n \in N_t \mid n =$ 

, où $(a, b) \in A^2$

$(v_1, \ldots, v_m) \in I^m$, $\bar{Q} \in B$, ensemble des dénotations d'expressions booléennes du langage} $\Rightarrow$

1) $\{(V_c - (\square))^m \xrightarrow{Q} \{\underline{\text{vrai}}, \underline{\text{faux}}\}$, où Q est l'expression dénotée par $\bar{Q}\}$

2) {l'exécution du noeud n, ne modifie aucune variable du programme, (pas d'effets de bord)}

3) {$\exists$ arc sortie vrai, arc sortie faux |

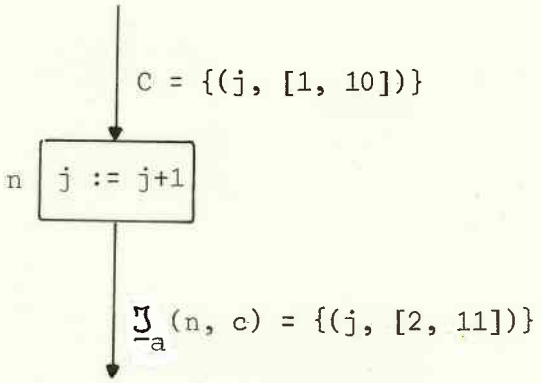
$N_t \xrightarrow{\text{arc sortie vrai}} A$; $N_t \xrightarrow{\text{arc sortie faux}} A$; avec

arc sortie vrai(n) = a et arc sortie faux (n) = b}

2.5 - Interprétation abstraite élémentaire -

2.5.1 - Interprétation élémentaire des affectations -

L'interprétation abstraite élémentaire d'un noeud n affectation, permet de calculer le contexte de l'arc de sortie du noeud n, égal à $\mathfrak{J}_a(n, c)$ en terme du contexte C de l'arc d'entrée du noeud n. Dans notre exemple introductif, on a vu que :



La fonction $\underline{\mathfrak{J}}_a$ est définie comme suit :

Hypothèse - H 300

1) - $\{N_a \times \mathcal{C} \xrightarrow{\underline{\mathfrak{J}}_a} \mathcal{C}\}$

$\{\forall n \in N_a \mid n =$ — [$v := \bar{f}(v_1, \ldots, v_m)$] → $\}$,

2) - $\{\forall C \in \mathcal{C}, \forall i \in I, i \neq v, \underline{\mathfrak{J}}_a(n, C)(i) = C(i)\}$

3) - $\{\forall C \in \mathcal{C},$

$$\underline{\mathfrak{J}}_a(n, C)(v) = @\left[\begin{array}{l}\{f(\nu_1, \ldots, \nu_m) \mid \\ \quad (\nu_1, \ldots, \nu_m) \in \gamma(C(v_1)) \times \ldots \times (C(v_m))\end{array}\right]\}\}$$

Définition - DEF 304

$\{\underline{\mathfrak{J}}_a$ est croissante sur $\mathcal{C}$ pour $\bar{\leq}\} \overset{\text{def}}{\Longleftrightarrow}$

$\{(\forall (x, y) \in \mathcal{C}^2) \wedge (\forall n \in N_a)$

$\{x \bar{\leq} y\} \Rightarrow \{\underline{\mathfrak{J}}_a(n, x) \bar{\leq} \underline{\mathfrak{J}}_a(n, y)\}\}$

Lemme - L 305

$\{\underline{\mathcal{J}}_a$ est croissante sur $\mathcal{C}$ pour $\bar{\leq}\}$

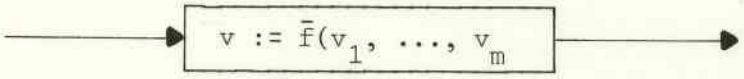
$$\{x \bar{\leq} y\} \Rightarrow \{\underline{\mathcal{J}}_a(n, x) \bar{\leq} \underline{\mathcal{J}}_a(n, y)\}$$

L111

$\Leftrightarrow$ $\quad \{\forall i \in I, x(i) \bar{\leq} y(i)\}$

$\Rightarrow$ $\quad \{\forall j \in I, \underline{\mathcal{J}}_a(n, x)(j) \bar{\leq} \underline{\mathcal{J}}_a(n, y)(j)\}$

Le noeud n est de la forme :



cas 1 - $j \neq v$,

d'après (H 300 - 2) $\underline{\mathcal{J}}_a(n, x)(j) = x(j)$ et $\underline{\mathcal{J}}_a(n, y)(j) = y(j)$

donc $x(j) \bar{\leq} y(j) \Rightarrow \underline{\mathcal{J}}_a(n, x)(j) \bar{\leq} \underline{\mathcal{J}}_a(n, y)(j)$.

cas 2 - $j = v$, (H 300, 3) $\Rightarrow$

$$\underline{\mathcal{J}}_a(n, x)(v) = @\left[\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_n) \in \gamma(x(v_1)) \times \ldots \times \gamma(x(v_m))\}\right]$$

et

$$\underline{\mathcal{J}}_a(n, y)(v) = @\left[\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_n) \in \gamma(y(v_1)) \times \ldots \times \gamma(y(v_m))\}\right]$$

mais $\forall k \in [1, m]\ x(v_k) \bar{\leq} y(v_k)$

L16

$\Rightarrow$ $\quad \gamma(x(v_k)) \sqsubseteq \gamma(y(v_k))$, $\forall k \in [1, m]$

$\Rightarrow$ $\quad \{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_m) \in \gamma(x(v_1)) \times \ldots \times \gamma(x(v_m))\}$

$\subset \{f(\tau_1, \ldots, \tau_m) \mid (\tau_1, \ldots, \tau_m) \in \gamma(y(v_1)) \times \ldots \times \gamma(y(v_m))\}$

$$\Rightarrow @[\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_m) \in \gamma(x(v_1)) \times \ldots \times \gamma(x(v_m))\}]$$

$$\bar{\leq} @ [\{f(\tau_1, \ldots, \tau_m) \mid (\tau_1, \ldots, \tau_m) \in \gamma(y(v_1)) \times \ldots \times \gamma(y(v_m))\}]$$

$$(\text{Car } V_1 \subset V_2 \Rightarrow V_2 = V_1 \cup V_2$$

$$\overset{H3}{\Rightarrow} \qquad @(V_2) = @(V_1 \cup V_2) = @(V_1) \bar{\cup} @(V_2)$$

$$\overset{DEF}{\Rightarrow} \qquad @(V_1) \bar{\leq} @(V_2))$$

$$\Rightarrow \qquad \underline{\mathfrak{I}}_a(n, x)(v) \bar{\leq} \underline{\mathfrak{I}}_a(n, y)(v)$$

Lemme - L 306

$$\forall n \in N_a \mid n = (v := \bar{f}(v_1, \ldots, v_m)) \wedge (v, v_1, \ldots, v_m) \in I^{m+1} \wedge \bar{f} \in E,$$

$$\{m = 0\} \Rightarrow \{\underline{\mathfrak{I}}_a(n,\Phi) = \{(v, @(\{f()\}))\}\}$$

$$\{m \geq 1\} \Rightarrow \{\underline{\mathfrak{I}}_a(n,\Phi) = \Phi\}$$

Pour démontrer le lemme L 306, on utilise le lemme L 116 :

$$\underline{\mathfrak{I}}_a(n,\Phi) = C \Leftrightarrow \{\forall i \in I, \ {}_a(n,\Phi)(i) = C(i)\}$$

1°) - i n'est pas membre gauche de l'affectation, alors d'après (H 300-2)

$$\underline{\mathfrak{I}}_a(n, \Phi)(i) = \Phi(i)$$

$$= \{(v, @(\{f()\}))\}(i) = \square$$

2°) - i est membre gauche de l'affectation, on a d'après (H 300-3)

$$\underline{\mathfrak{I}}_a(n, \Phi)(i) = @ [\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_m) \in \gamma(\Phi(v_1)) \times \ldots \times \gamma(\Phi(v_m))\}]$$

$$(\text{DEF102}) \qquad = @ [\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_m) \in \gamma(\square) \times \ldots \times \gamma(\square)\}]$$

2.1) $m \geq 1$

Mais $f(\nu_1, \ldots, \nu_m)$ n'est pas définie quand un de ses arguments ν_k est à valeurs dans $\gamma(\square)$ (voir (H 212-1)). Par conséquent, l'ensemble :

$\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_m) \in \gamma(\square) \times \ldots \times \gamma(\square)\} = \emptyset$, ensemble vide.

Donc :

$$\underline{\mathfrak{I}}_a(n, \Phi)(i) = @(\emptyset)$$
$$\text{(L8)} \qquad = \square$$
$$\text{(DEF 102)} \qquad = \Phi(i)$$

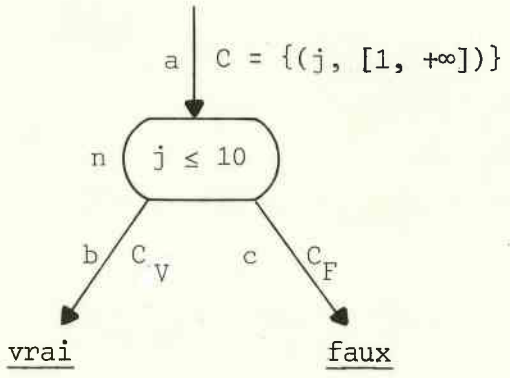
2.2) Si au contraire f, ne dépend d'aucun argument, (m = 0), f est constante, toujours définie, donc :

$$\underline{\mathfrak{I}}_a(n, \Phi)(i) = @[\{f()\}]$$

2.5.2 - Interprétation élémentaire des tests -

L'interprétation abstraite élémentaire d'un noeud n de test dans un contexte C, produit un contexte C_V pour l'arc de sortie vrai, et un contexte C_F pour l'arc de sortie faux. On note $(C_V, C_F) = \underline{\mathfrak{I}}_t(n, c)$.

Dans l'exemple introductif, on a vu le cas :



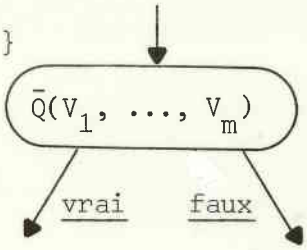
on a $\underline{\mathfrak{I}}_t(n, C) = (C_V, C_F) = (\{(j, [1, 10])\}, \{(j, [11, +\infty])\}$

En effet, j étant compris entre 1 et $+\infty$ sur l'arc a, il est compris entre 1 et 10 sur la branche b = arc sortie vrai(n), et est supérieure à 11 sur la branche C = arc sortie faux (n).

On peut formaliser les propriétés de $\underline{\mathfrak{I}}_t$ comme suit :

Hypothèse - H 307

1) - $\{N_t \times \mathcal{C} \xrightarrow{\underline{\mathcal{J}}_t} \mathcal{C} \times \mathcal{C}\}$

$\{(\forall n \in N_t \mid n =$ 

, où

$(v_1, \ldots, v_m) \in I^m, \bar{Q} \in B) \wedge$

$(\forall (C, C_V, C_F) \in \mathcal{C}^3 \mid \underline{\mathcal{J}}_t(n, C) = (C_V, C_F))\} \Rightarrow$

2) $(\forall i \in I)$,

$C_V(i) = @(\{\tau \mid (\tau \in \gamma(C(i)))$
$(\exists (\nu_1, \ldots, \nu_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m)) \mid$
$Q(\nu_1, \ldots, \nu_m))\})$

$C_F(i) = @(\{\tau \mid (\tau \in \gamma(C(i)))$
$(\exists (\nu_1, \ldots, \nu_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m)) \mid$
$\neg Q(\nu_1, \ldots, \nu_m))\})$

Comme précédemment, $\underline{\mathcal{J}}_t$ a certaines propriétés remarquables, sur lesquelles repose la correction de l'algorithme d'interprétation abstraite.

Définition - DEF 311

$\{\underline{\mathcal{J}}_t$ est croissante sur $\mathcal{C}$ pour $\bar{\leq}\} \overset{\text{def}}{\Longleftrightarrow}$

$\{\forall (x, y) \in \mathcal{C}^2 \wedge (\forall n \in N_t)$,

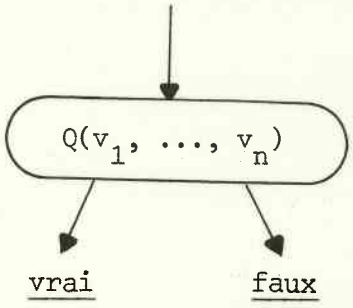
$\{x \bar{\leq} y\} \Rightarrow \{(\underline{\mathcal{J}}_t(n, x)[1] \bar{\leq} \underline{\mathcal{J}}_t(n, y)[1]) \wedge$

$(\underline{\mathcal{J}}_t(n, x)[2] \bar{\leq} \underline{\mathcal{J}}_t(n, y)[2])\}$

Lemme - L 312

$\{\underline{\mathcal{J}}_t$ est croissante sur $\mathcal{C}$ pour $\bar{\leq}\}$

Supposons que le noeud n soit de la forme



On fait la démonstration dans le cas de la branche de sort e vrai (même démonstration pour la branche de sortie faux).

$$\{x \tilde{\leq} y\} \Rightarrow \{(\underline{\mathcal{J}}_t(n, x)[1] \tilde{\leq} \underline{\mathcal{J}}_t(n, y)[1]\}$$

L 111

$\Leftrightarrow$ $\{\forall i \in I, x(i) \bar{\leq} y(i)\} \Rightarrow$

$\{\forall j \in I, \underline{\mathcal{J}}_t(n, x)[1](j) \bar{\leq} \underline{\mathcal{J}}_t(n, y)[1](j)\}$

$\forall i \in I, x(i) \underline{\leq} y(i)$

L16 $\Rightarrow \gamma(x(i)) \subset \gamma(y(i))$

$\Rightarrow \{\tau \mid (\tau \in \gamma(x(i))) \wedge (\exists (\nu_1, \ldots, \nu_m) \in \gamma(x(v_1)) \times \ldots \times \gamma(x(v_m)) \mid Q(\nu_1, \ldots, \nu_m))\}$

$\subset \{\tau \mid (\tau \in \gamma(y(i))) \wedge (\exists (l_1, \ldots, l_m) \in \gamma(y(v_1)) \times \ldots \times \gamma(y(v_m)) \mid Q(l_1, \ldots, l_m))\}$

H3 $\Rightarrow$ $@\{\tau \mid (\tau \in \gamma(x(i))) \wedge (\exists (\nu_1, \ldots, \nu_m) \in \gamma(x(v_1)) \times \ldots \times \gamma(x(v_m)) \mid Q(\nu_1, \ldots, \nu_m))\}$

$\underline{\leq}$ $@\{\tau \mid (\tau \in \gamma(x(i))) \wedge (\exists (l_1, \ldots, l_m) \in \gamma(y(v_1)) \times \ldots \times \gamma(y(v_m)) \mid Q(l_1, \ldots, l_m))\}$

H303-2 $\Rightarrow$ $\underline{\mathcal{J}}_t(n,x)[1](i) \underline{\leq} \underline{\mathcal{J}}_t(n,y)[1](i)$

Lemme - L 313

$\forall n \in N_t$, $\underline{\mathcal{J}}_t(n, \Phi) = (\Phi, \Phi)$

$$\underline{\mathcal{J}}_t(n, \Phi) = (\Phi, \Phi) \overset{L116}{\Longleftrightarrow} \{\forall i \in I, (\underline{\mathcal{J}}_t(n, \Phi)\ [1]\ (i) = \Phi(i)) \wedge (\underline{\mathcal{J}}_t(n, \Phi)\ [2]\ (i) = \Phi(i))$$

On a d'après (H307-2),

$\forall i \in I$,

$$\underline{\mathcal{J}}_t(n, \Phi)\ [1]\ (i) = @[\{\tau \mid (\tau \in \gamma(\Phi(i))) \wedge (\exists(\nu_1, \ldots, \nu_m) \in \gamma(\Phi(v_1) \times \ldots \times \Phi(v_m)) \mid Q(\nu_1, \ldots, \nu_m)\}]$$

$$= @[\{\tau \mid (\tau \in \gamma(\Box)) \wedge (\exists(\nu_1, \ldots, \nu_m) \in \gamma(\Box) \times \ldots \times \gamma(\Box) \mid Q(\nu_1, \ldots, \nu_m))\}]$$

mais Q n'est pas définie pour des valeurs de ses paramètres dans $\gamma(\Box)$, donc la condition :

$(\exists(\nu_1, \ldots, \nu_m) \in \gamma(\Box) \times \ldots \times \gamma(\Box) \mid Q(\nu_1, \ldots, \nu_m))$

n'est jamais vérifiée, soit

$$\underline{\mathcal{J}}_t(n, \Phi)\ [1]\ (i) = @[\{\tau \mid (\tau \in \gamma(\Box)) \wedge \underline{\text{faux}}\}]$$

$$= @[\emptyset]$$

$$= \Box \qquad \text{(L8)}$$

$$= \Phi(i) \qquad \text{(DEF 102)}$$

Même raisonnement pour $\underline{\mathcal{J}}_t(n, \Phi)\ [2]\ (i)$.

2.6 - Algorithme d'interprétation abstraite -

L'idée de base de l'algorithme d'interprétation abstraite, est de commencer avec un contexte initial sur l'arc d'entrée du programme, et le contexte vide sur les autres arcs. Pour chacun des différents types de noeuds, on décrit une transformation qui spécifie le contexte sur l'arc (les arcs) de sortie du noeud, en termes des contextes associés aux arcs d'entrée, et quand c'est nécessaire du contenu de ce noeud. Toutes les transformations ont la propriété que si un identificateur a une valeur quelconque choisie dans un contexte d'entrée, alors sa valeur de sortie, consécutive à n'importe quelle exécution du programme est dans le contexte de sortie. L'algorithme applique ces transformations, jusqu'à ce que les tous les contextes associés aux arcs du graphe se soient stabilisés, c'est-à-dire qu'une transformation appliquée à n'importe quel noeud, n'apporte aucune modification au contexte de sortie de ce noeud. La preuve de terminaison, repose sur l'élargissement des contextes aux noeuds de jonction de boucle et sur (L 117) qui interdit un nombre infini d'élargissements.

La compréhension de l'algorithme peut-être facilitée par la re-lecture du chapitre 1, elle nécessite l'étude des preuves de terminaison et de correction.

```
[1] procédure interprétation abstraite (graphe = (A x (N_a, N_t, N_js, N_jb, N_s, {n_e})))
[2] début
[3]    pour chaque arc de A faire contexte local (arc) := Φ refaire ;
[4]    chemins à exécuter := {arc initial (graphe)} ; jonctions := Ø ;
[5]    tantque (chemins à exécuter ≠ Ø) faire
[6]       tantque (chemins à exécuter ≠ Ø) faire  $sélectionner un arc à
                                                    traverser$
[7]          arc := choix (chemins à exécuter) ;
[8]          chemins à exécuter := chemins à exécuter -{arc} ;
[9]          noeud traité := extrémité-finale (arc) ;
[10]         cas
[11]         ┤ noeud traité ∈ N_a →
[12]            calcul contexte sortie (arc sortie(noeud traité),
                J_a(noeud traité, contexte local(arc))) ;
[13]         ┤ noeud traité ∈ N_t →
[14]            (V, F) := J_t(noeud traité, contexte local(arc)) ;
[15]            calcul contexte sortie (arc sortie vrai (noeud traité), V) ;
[16]            calcul contexte sortie (arc sortie faux (noeud traité), F) ;
[17]         ┤ noeud traité ∈ (N_js ∪ N_jb) → jonctions ∪ := {noeud traité} ;
[18]         ┤ noeud traité ∈ N_s → ;
[19]         fincas ;
[20]      refaire ;
[21]         pour chaque noeud de jonctions faire

[22]          contexte sortie := ⊎ contexte local (prédécesseur)
                                 prédécesseur  arcs d'entrée (noeud)

[23]          si ¬(contexte sortie ⊑ contexte local (arc sortie(noeud)) alors
[24]             cas
[25]             ┤ noeud ∈ N_js →
[26]                calcul contexte sortie (arc sortie (noeud),
[27]                                        contexte sortie) ;
[28]             ┤ noeud ∈ N_jb →
[29]                calcul contexte sortie (arc sortie(noeud),
[30]                   contexte local (arc sortie(noeud)) ⊽ contexte sortie) ;
[31]             fincas ;
[32]          finsi ;
[33]         refaire ;
[34]         jonction := Ø ;
[35]   refaire ;
[36]   procédure calcul contexte sortie (arc, contexte) ;
[37]   début
[38]      si ¬ (contexte ⊑ contexte local (arc)) alors
[39]         contexte local (arc) := contexte ;
[40]         chemins à exécuter ∪ := {arc} ;
[41]      finsi ;
[42]   fin ;
[43] fin
```

Remarques :

1) en [25], on a d'après [23] :

$$\neg(\text{contexte sortie} \stackrel{=}{\leq} \text{contexte local (arc sortie (noeud))}$$

d'après le passage de paramètres, en [26], [27] le test qui suit en [38] est vrai.

2) en [28], on a d'après [23] :

$$\neg(\text{contexte sortie} \stackrel{=}{\leq} \text{contexte local (arc sortie (noeud))}$$

$$(P13) \Longrightarrow \neg((\text{contexte local (arc sortie (neoud))} \stackrel{=}{\nabla} \text{contexte sortie}) \stackrel{=}{\leq} \text{contexte local (arc sortie (noeud))}$$

d'après le passage de paramètres, en [29]-[30], le test qui suit en [38] est également toujours vrai.

Le test de [38] est utile pour les appels de [12], [15], [16].

2.7 - Preuve de terminaison de l'algorithme d'interprétation abstraite -

Lemme - L400

Soit $(C_0, C_1, C_2, \ldots, C_m)$ la suite des "contextes locaux" associés successivement à un arc α, sortie de noeud de jonction de boucle, par la procédure "interprétation abstraite", à la ligne [29] ou [26]. Alors :

$\{ m \geq 0, \Phi = C_0 < C_1 < C_2 < \ldots < C_m \}$

La démonstration se fait par récurrence sur m.

- $m = 0$, $C_0 = \Phi([3])$ pour tout arc du graphe ;
- $m > 0$, supposons le théorème établi jusqu'à $m = m_0$.

On le démontre pour $m = m_0 + 1$;

L'affectation de C_{m_0+1} à contexte local (α), se fait en [29]. En ce point, on a d'après [23] :

$$\neg(\text{contexte sortie} \stackrel{=}{\leq} C_{m_0}) \qquad (1)$$

puis le test de la ligne [38] étant vrai, on a en [39]:

$$C_{m_0+1} = (C_{m_0} \stackrel{=}{\nabla} \text{contexte sortie}) \qquad (2)$$

(1) $\wedge(P6) \Longrightarrow C_{m_0} \bar{\leq} (C_{m_0} \bar{\nabla}$ contexte sortie)

(2) $\Longrightarrow C_{m_0} \bar{\leq} C_{m_0} + 1$

Par récurrence sur m, on obtient le résultat L400 sur lequel repose la terminaison de l'algorithme.

Théorème T401

La procédure "interprétation abstraite" termine pour tout graphe de programme.

Pour démontrer le théorème, en supposant que les opérations élémentaires $\underline{\nabla}_a$, $\underline{\nabla}_t$, $\bar{\cup}$, $\bar{\nabla}$ et $\bar{\leq}$ prennent un temps de calcul fini, il nous suffit de montrer que toutes les boucles dans l'algorithme sont exécutées un nombre fini de fois. Pour les cas non triviaux, ceci est démontré par les lemmes suivants :

Lemme L402

La boucle [21] à [33]est exécutée un nombre fini de fois.

Le nombre d'éléments de "jonctions" est fini d'après (H200) avant d'entrer dans la boucle, et on ne rajoute pas de noeuds à "jonctions" dans le corps [22] - [32] de cette boucle.

Lemme L403

La boucle [6] à [20] est exécutée un nombre fini de fois.

La démonstration se fait par l'absurde :

L'ensemble "chemins à exécuter" est fini, car N est fini (H200). Etant donné que la boucle ne termine pas, on enlève un nombre infini d'arcs à "chemins à exécuter" en [8]. Comme ce processus ne conduit jamais à un ensemble "chemins à exécuter" égal à vide (en [6]), c'est qu'ailleurs (en [12], [15], [16]), on a ajouté un nombre infini d'arcs à "chemins à exécuter". Comme le nombre d'arcs différents que l'on peut ainsi rajouter

est fini (H202 et H200) c'est que l'on a rajouté deux fois le même arc. c'est-à-dire que l'on a suivi un cycle = a = a_0, a_1, ..., a_m = a.
Le graphe de programme étant bien formé, ce cycle contient un arc a_i, dont l'extrémité finale est un noeud traité [9] de type jonction de boucle (L210). Dans ces conditions, la ligne [17] est exécutée, sans que a_i soit ajouté à "chemins à exécuter". Ceci est en contradiction avec le fait qu'on ait suivi le cycle a_0, a_1, ..., a_i, ..., a_m, donc le lemme est démontré par l'absurde.

Lemme L404

La boucle [5] à [35]est exécutée en nombre fini de fois.

Supposons, à contrario, que cette boucle soit infinie.
D'après les lemmes (L402) et (L403), on passe un nombre infini de fois en [34], avec {"chemins à exécuter" ≠ ∅}.Comme à la suite de [6] - [20] (qui termine) on a {"chemins à exécuter" = ∅}, c'est qu'à la suite de [21] - [33] on a rajouté au moins un arc à "chemins à exécuter".
Ceci étant vrai, pour chaque itération dans la boucle [5] - [35], c'est qu'on a exécuté un nombre infini de fois l'instruction [23] -[32].

a) Si l'instruction [29] est exécutée un nombre infini de fois, il existe un arc a égal à arc sortie (noeud), un nombre infini de fois dans l'instruction [29], car le nombre de ces arcs a est fini (H200).
On peut donc trouver une suite infinie de "contextes de sortie" CS_0, CS_1,... tels que la séquence CL_0, CL_1,... des contextes locaux de a soit de la forme $CL_0 = \Phi$, $CL_1 = CL_0 \bar{\nabla} CS_1$, ..., $CL_n = CL_{n-1} \bar{\nabla} CS_n$, ... et soit infinie.
La suite de ces contextes étant strictement croissante, d'après le lemme (L400),on a une contradiction avec le lemme (L117).L'instruction [29] est donc exécutée un nombre fini de fois.

b) Remarquons, que si l'instruction [23] -[32] a été exécutée un nombre infini de fois, alors que l'instruction [29] est exécutée un nombre de fois fini, c'est que l'instruction [26] est exécutée un nombre infini de fois (jonction" $\subseteq (N_{j_s} \cup N_{j_b})$). Il existe un arc a égal à arc sortie (noeud)

un nombre infini de fois dans l'instruction [26], car le nombre de ces arcs a est fini (H200). C'est dire que l'on a parcouru un nombre infini de fois, un certain cycle $a = a_0, a_1, \ldots, a_n = a$ du graphe. D'après (L210) ce cycle contient au moins un noeud n_{jb} de jonction de boucle, pour lequel on exécuterait à chaque fois, donc un nombre infini de fois, l'instruction [29]. Ce dernier cas, étant encore impossible, le lemme a été démontré par l'absurde.

2.8 - Preuve de correction de l'algorithme d'interprétation abstraite -

Lemme L500

Soit $(C_{\alpha 0}, C_{\alpha 1}, \ldots, C_{\alpha m})$ la suite des "contexte locaux" affectés à un arc α quelconque du graphe, par la procédure d'"interprétation abstraite", alors $(\forall \alpha, \forall m \geq 0)$, $\Phi = C_{\alpha 0} \overset{=}{<} \ldots \overset{=}{<} C_{\alpha m}$.

- Le lemme est trivialement vrai, pour l'arc d'entrée du graphe, pour lequel m = 0.
- Tout arc α, autre que arc initial (graphe) étant arc de sortie d'un noeud n, il suffit de démontrer le lemme pour des arcs de sortie de noeud affectation, test ou jonction simple, puisque (L400) a traité le cas des arcs sortie de noeud de jonction de boucle.

Pour démontrer le lemme, nous allons raisonner par récurrence sur le nombre d'itérations τ dans la boucle [5] ∿ [35].

On peut regrouper les raisonnements sur les arcs α de sortie des noeuds n d'affectation ou de test, en définissant :

$\underline{\mathcal{J}}(n, C) =$ cas

- $n \in N_a \rightarrow \underline{\mathcal{J}}_a (n, C)$;
- $(n \in N_t) \wedge (\alpha =$ arc sortie vrai $(n)) \rightarrow \underline{\mathcal{J}}_t (n, C)$ [1] ;
- $(n \in N_t) \wedge (\alpha =$ arc sortie faux $(n)) \rightarrow \underline{\mathcal{J}}_t (n, C)$ [2] ;

fincas ;

Cas 1 : $\tau = 1$

Pour exécuter la boucle [5] - [35], on commence par la boucle [6] ~ [20], dans laquelle on ne fait que des affectations aux contextes locaux des arcs de sortie des noeuds affectation ou test [12], [15], [16].
On a $C_{\alpha 0} = \Phi$ d'après l'initialisation à la ligne [3].
Si $C_{\alpha 1}$ est affecté au contexte local de α en [39], c'est que d'après [38], on a $\neg(C_{\alpha 1} \bar{\leq} C_{\alpha 0})$ soit $\neg(C_{\alpha 1} \bar{\leq} \Phi)$,

(P6) $\Rightarrow \Phi < \Phi \bar{\cup} C_{\alpha 1}$

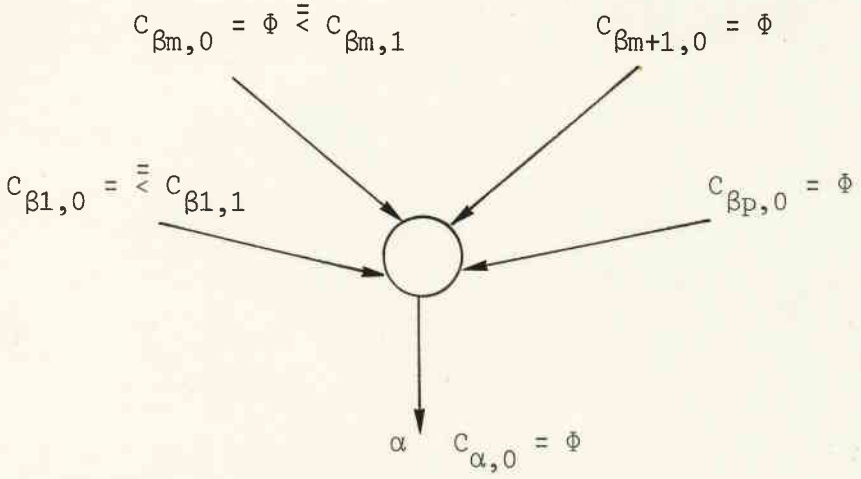
(DEF 110) $\Rightarrow \Phi \neq \Phi \bar{\cup} C_{\alpha 1}$

(L108) $\Rightarrow \Phi \neq C_{\alpha 1}$

(p7 - 6) $\Rightarrow \Phi < C_{\alpha 1}$

$\Rightarrow C_{\alpha 0} < C_{\alpha 1}$

Etant sorti de la boucle [5] ~ [20], on exécute la boucle [21] - [31], et seul le cas des noeuds de jonction simple est à étudier [25]; la configuration des noeuds de jonction est la suivante :



Les contextes locaux des arcs $\beta_1, \ldots, \beta_m$ sont non nuls, tandis que les contextes locaux des arcs $C_{\beta m+1}$ à $C_{\beta p}$ sont nuls.

D'après [3], $C_{\alpha,0} = \Phi$. Quand on fait une affectation au contexte local de α, c'est que :

$$\neg((\overset{p}{\underset{i=1}{U}} C_{\beta i,1}) \overset{=}{\leq} C_{\alpha,0} = \Phi) \text{ (d'après [22], [23])}$$

$$\text{(P6)} \implies \Phi \overset{=}{\leq} \Phi \overset{=}{U} (\overset{p}{\underset{i=1}{U}} C_{\beta i,1})$$

$$\text{(D110, (L108,4))} \implies \Phi \neq \overset{p}{\underset{i=1}{U}} C_{\beta i,1}$$

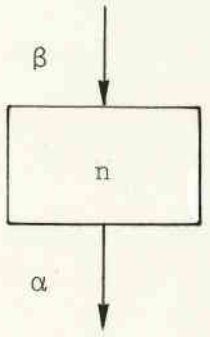
$$\text{(P7, b) et [26], [39]} \implies C_{\alpha,0} \overset{=}{\leq} \overset{p}{\underset{i=1}{U}} C_{\beta i,1} = C_{\alpha,1}.$$

Pour tous les arcs dont le contexte local n'est pas modifié par cette première itération dans la boucle [5] - [35], le lemme est également vrai, avec m = 0, à la suite de cette première itération.

Cas 2 : $\tau \geq 1$

On suppose par récurrence que le lemme est vrai en [5] après τ_0 itérations dans la boucle [5] - [35], on montre qu'il est également vrai à la $(\tau_0 + 1)^{\text{ième}}$ itération.

2.1) On commence par faire la preuve pour les arcs dont le contexte local est affecté dans la boucle [6] à [20]. Par hypothèse de récurrence, on a la configuration suivante :



$$C_{\beta,0} = \Phi \overset{=}{\leq} C_{\beta,1} \overset{=}{\leq} \ldots \overset{=}{\leq} C_{\beta,m} \overset{=}{\leq} C_{\beta,m+1}$$

$$C_{\alpha,0} = \Phi \overset{=}{\leq} C_{\alpha,1} \overset{=}{\leq} \ldots \overset{=}{\leq} C_{\alpha,m}$$

D'après [12], [15], ou [16] et [39] :

$$C_{\alpha,m} = \mathfrak{J}(n, C_{\beta,m}) \text{ et } C_{\alpha,m+1} = \mathfrak{J}(n, C_{\beta,m+1})$$

mais d'après (L305) ou (L312) :

$$C_{\beta,m} \overset{=}{<} C_{\beta,m+1} \Longrightarrow C_{\alpha,m} \overset{=}{\leq} C_{\alpha,m+1}$$

En outre d'après la ligne [38], on a :

$$\neg(C_{\alpha,m+1} \overset{=}{\leq} C_{\alpha,m})$$

soit :

$$(C_{\alpha,m} \overset{=}{\leq} C_{\alpha,m+1}) \wedge \neg (C_{\alpha,m+1}) \overset{=}{\leq} C_{\alpha,m})$$

(DEF 109) $\Longrightarrow ((C_{\alpha,m} \overset{=}{\cup} C_{\alpha,m+1}) = C_{\alpha,m+1}) \wedge ((C_{\alpha,m+1} \overset{=}{\cup} C_{\alpha,m}) = C_{\alpha,m})$

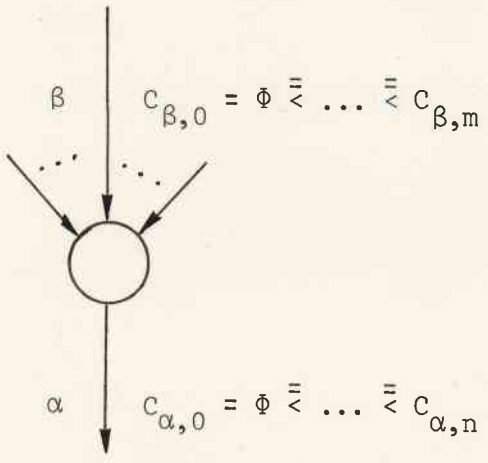
$\Longrightarrow \neg(C_{\alpha,m+1} = C_{\alpha,m}) \Longrightarrow C_{\alpha,m+1} \neq C_{\alpha,m}$

comme $(C_{\alpha,m} \overset{=}{\leq} C_{\alpha,m+1}) \wedge (C_{\alpha,m} \neq C_{\alpha,m+1})$

(DEF110) $\Longrightarrow C_{\alpha,m} \overset{=}{<} C.$

En conséquence, à la $(\tau_0+1)^{\text{ième}}$ itération dans la boucle [5] à [35], le lemme est vrai à la sortie de la boucle [6] à [20].

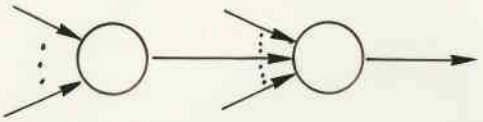
2.2) Ensuite, on traite les noeuds de jonctions dans la boucle [21] à [33]; Seul le cas des noeuds de jonction simple est à étudier ; on a la configuration suivante :



Sur chacun des arcs d'entrée du noeud de jonction simple, la séquence des contextes locaux est strictement croissante. En effet deux espèces d'arcs sont à considérer :

- les arcs d'entrée dont le contexte local n'a pas été modifié dans la $(\tau_0+1)^{ième}$ itération. Dans ce cas, d'après notre hypothèse de récurrence, la séquence des contextes affectés à ces arcs est strictement croissante.

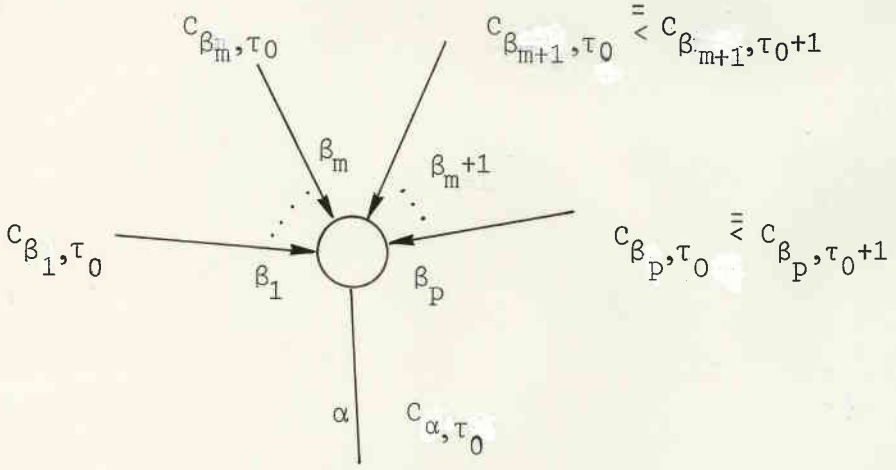
- Les arcs d'entrée dont le contexte local a été modifié par la $(\tau_0 + 1)^{ième}$ itération. Notons qu'il s'agit forcément d'arcs de sortie de noeuds affectation ou test, puisque deux noeuds tests successeurs :



ne peuvent pas être traités dans la même itération $(\tau_0 + 1)$ d'après [17].

En conséquence, pour ces arcs sorties de noeuds affectation ou test, la séquence des contextes locaux est d'après 2.1 strictement croissante ;

On a donc :



On avait :

$$C_{\alpha,\tau_0} = \bar{\bar{\bigcup}}_{i=1}^{p} C_{\beta_i,\tau_0}$$

En [22], on calcule :

$$C_{\alpha,\tau_0+1} = \bar{\bar{\bigcup}}_{i=1}^{p} C_{\beta_i,\tau_0+1} = \left(\bar{\bar{\bigcup}}_{i=1}^{p} C_{\beta_i,\tau_0}\right) \bar{\bar{\cup}} \left(\bar{\bar{\bigcup}}_{i=1}^{p} C_{\beta_i,\tau_0+1}\right)$$

et d'après [23], on a :

$$\neg\left(C_{\alpha,\tau_0+1} \bar{\bar{\leq}} C_{\alpha,\tau_0}\right)$$

Montrons également que :

$$C_{\alpha,\tau_0} \bar{\bar{\leq}} C_{\alpha,\tau_0+1}$$

on a :

$$C_{\alpha,\tau_0} \bar{\bar{\cup}} C_{\alpha,\tau_0+1} = \left(\bar{\bar{\bigcup}}_{i=1}^{p} C_{\beta_i,\tau_0}\right) \bar{\bar{\cup}} \left(\left(\bar{\bar{\bigcup}}_{i=1}^{p} C_{\beta_i,\tau_0}\right) \bar{\bar{\cup}} \left(\bar{\bar{\bigcup}}_{i=m+1}^{p} C_{\beta_i,\tau_0+1}\right)\right)$$

(L108) $\Longrightarrow$

$$= \left(\bar{\bar{\bigcup}}_{i=1}^{m} C_{\beta_i,\tau_0}\right) \bar{\bar{\cup}} \left(\bar{\bar{\bigcup}}_{i=m+1}^{p} \left(C_{\beta_i,\tau_0} \bar{\bar{\cup}} C_{\beta_i,\tau_0+1}\right)\right)$$

Mais :

$$\left(C_{\beta_i,\tau_0}\right) \bar{\bar{\leq}} \left(C_{\beta_i,\tau_0+1}\right), \quad \forall i \in [m+1,\ p]$$

donc (DEF109) $\Longrightarrow C_{\beta_i,\tau_0} \bar{\bar{\cup}} C_{\beta_i,\tau_0+1} = C_{\beta_i,\tau_0+1}$, soit :

$$C_{\alpha,\tau_0} \bar{\bar{\cup}} C_{\alpha,\tau_0+1} = \left(\bar{\bar{\bigcup}}_{i=1}^{m} C_{\beta_i,\tau_0}\right) \bar{\bar{\cup}} \left(\bar{\bar{\bigcup}}_{i=1}^{p} C_{\beta_i,\tau_0+1}\right)$$

$$= C_{\alpha,\tau_0+1}$$

En conséquence,

$$(C_{\alpha,\tau_0} \stackrel{=}{\leq} C_{\alpha,\tau_0+1}) \text{ et } \neg(C_{\alpha,\tau_0+1} \stackrel{=}{\leq} C_{\alpha,\tau_0})$$

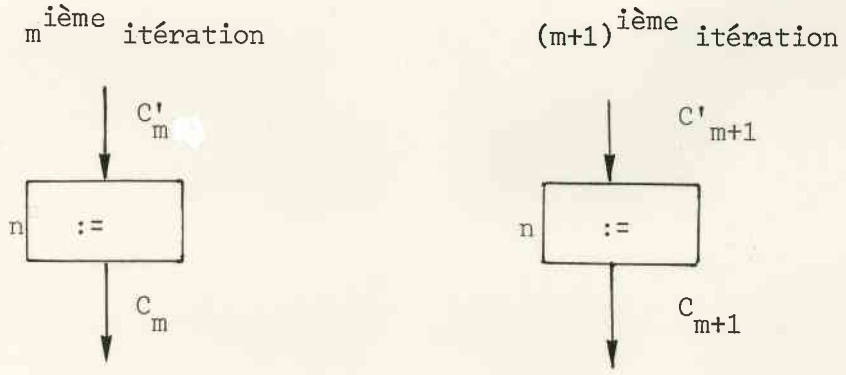
$$\Longrightarrow C_{\alpha,\tau_0} \stackrel{=}{<} C_{\alpha,\tau_0+1}$$

2.3) Pour les arcs, dont le contexte local n'est pas modifié, en l'itération (τ_0+1), le lemme a été établi lors d'une des itérations $1, 2, \ldots, \tau_0$.

3) Finalement, le lemme est démontré par récurrence sur τ : la séquence des contextes locaux affectés successivement à un arc est strictement croissante.

Remarque : Ce lemme justifie l'idée intuitive que les contextes associés successivement aux arcs du graphe, sont de plus en plus "larges", c'est-à-dire que l'on détermine des domaines de valeurs de plus en plus grands pour les variables de ces contextes.

Ce lemme éclaircit également le phénomène suivant : sur un arc de sortie d'un noeud affectation ou test, on ne fait pas l'union des contextes obtenus à chaque itération :



Les valeurs des identificateurs dans le contexte C_{m+1}, sont celles obtenues à la $m^{ème}$ itération, donc celles de C_m, unies à celles obtenues à la $(m+1)^{ème}$ itération, soit $\underline{J}(n, C'_{m+1})$ - Il faut donc poser :

$$C_{m+1} = C_m \stackrel{=}{\cup} \underline{J}(n, C'_{m+1})$$

Mais :

$$(L305) \text{ ou } (L311) \Longrightarrow C'_m < C'_{m+1} \Longrightarrow \underline{\mathcal{J}}(n, C'_m) \leq \underline{\mathcal{J}}(n, C'_{m+1})$$

$$\Longrightarrow C_m \leq \underline{\mathcal{J}}(n, C'_{m+1})$$

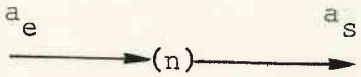
$$\Longrightarrow (C_m \bar{\cup} \underline{\mathcal{J}}(n, C_{m+1})) = \underline{\mathcal{J}}(n, C_{m+1})$$

Donc $C_{m+1} = C_m \bar{\cup} \underline{\mathcal{J}}(n, C'_{m+1}) = \underline{\mathcal{J}}(n, C'_{m+1})$.

La justification intuitive est que C'_{m+1} inclut C'_m, donc l'interprétation de n dans le contexte C'_{m+1} introduit dans C_{m+1} toutes les valeurs se trouvant dans C_m.

Lemme - L501

Pour tout noeud n du graphe de programme :



où C_{a_e} et C_{a_s} sont les derniers contextes locaux associés aux arcs a_e et a_s, par la procédure d'interprétation abstraite, on a :

$\forall n \in N$:

cas

- $n \in N_a \longmapsto \underline{\mathcal{J}}_a(n, C_{a_e}) = C_{a_s}$;
- $(n \in N_t) \wedge (a_s = \underline{\text{arc sortie vrai}}\ (n)) \longmapsto \underline{\mathcal{J}}_t(n, C_{a_e})\ [1] = C_{a_s}$;
- $(n \in N_t) \wedge (a_s = \underline{\text{arc sortie faux}}\ (n)) \longmapsto \underline{\mathcal{J}}_t(n, C_{a_e})\ [2] = C_{a_s}$;
- $n \in (N_{j_s} \cup N_{j_b}) \longmapsto C_{a_e} \bar{\leq} C_{a_s}$;

fincas ;

Intuitivement ce lemme exprime que lorsque l'interprétation abstraite se termine, une transformation élémentaire appliquée à n'importe quel noeud du graphe, n'approte aucune modification au contexte local assoié à l'arc de sortie de ce noeud.

1°) n est un noeud test ou affectation - $n \in (N_a \cup N_t)$.

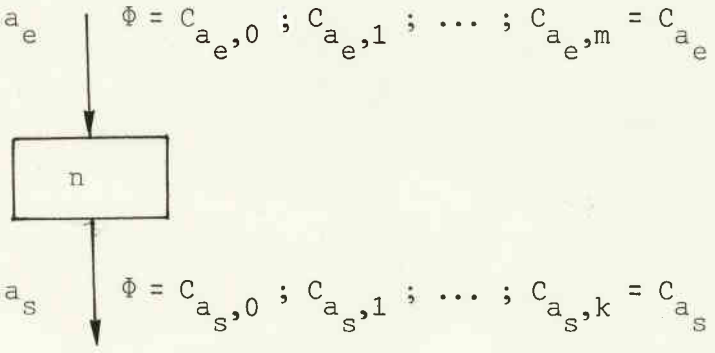
Définissons :

$$\mathcal{J}(n, C_{a_e}) = \underline{\text{cas}}$$
$$\quad n \in N_a \rightarrow \mathcal{J}_a(n, C_{a_e}) ;$$
$$\quad (n \in N_t) \wedge (a_s = \underline{\text{arc sortie vrai}}\ (n)) \rightarrow \mathcal{J}_t(n, C_{a_e})\ [1] ;$$
$$\quad (n \in N_t) \wedge (a_s = \underline{\text{arc sortie faux}}\ (n)) \rightarrow \mathcal{J}_t(n, C_{a_e})\ [2] ;$$
$$\underline{\text{fincas}}$$

Notons $C_{a_e,0}$, $C_{a_e,1}$, ..., $C_{a_e,m} = C_{a_e}$ et $C_{a_s,1}$, $C_{a_s,2}$, ..., $C_{a_s,k} = C_{a_s}$, les suites de contextes locaux associés par l'interprétation abstraite aux arcs d'entrée a_e et de sortie a_s du noeud n. On doit montrer que :

$$\mathcal{J}(n, C_{a_e}) = C_{a_s}$$

On a la configuration suivante :



1.1) Le cas $m < k$ est impossible, car pour traverser l'arc a_s (et lui affecter un contexte local), on est obligé de traverser a_e (et de lui affecter un contexte local).

1.2) Le cas $m = k$, correspond au fait que les arcs a_e et a_s ont été trversés un nombre égal de fois . Par récurrence sur m :

1.2.1) $n = k = 1$

D'après la ligne [38], on a : $\neg(\underline{\mathcal{J}}(n, C_{a_e,1}) \bar{\bar{\leq}} C_{a_s,0}) \Rightarrow$

$\neg(\underline{\mathcal{J}}(n, C_{a_e,1}) \bar{\bar{\leq}} \Phi) \overset{P6}{\Longrightarrow} \Phi < \Phi \bar{\bar{\cup}} \underline{\mathcal{J}}(n, C_{a_e,1})$

(L108-4) $\Rightarrow \Phi \bar{\bar{<}} \underline{\mathcal{J}}(n, C_{a_e,1}) \Rightarrow \Phi \bar{\bar{\leq}} \underline{\mathcal{J}}(n, C_{a_e})$

$(\Phi = C_{a_s,0}) \Rightarrow C_{a_s,0} \bar{\bar{\leq}} \underline{\mathcal{J}}(n, C_{a_e})$, donc le test de la ligne [38] est vrai car :

(DEF 110) $\Rightarrow (\underline{\mathcal{J}}(n, C_{a_e}) = C_{a_s,0} \bar{\bar{\cup}} \underline{\mathcal{J}}(n, C_{a_e})) \wedge (C_{a_s,0} \neq \underline{\mathcal{J}}(n, C_{a_e}))$

$\Rightarrow (C_{a_s,0} \neq C_{a_s,0} \bar{\bar{\cup}} \underline{\mathcal{J}}(n, C_{a_e}))$

(DEF 109) $\Rightarrow \neg(\underline{\mathcal{J}}(n, C_{a_e}) \bar{\bar{\leq}} (C_{a_s,0})$

D'après [39], on a : $C_{a_s,1} = C_{a_s} = \underline{\mathcal{J}}(n, C_{a_e})$.

1.2.2) $(m \geq 2) \wedge (m = k)$

Supposons le théorème vrai pour $m = 1, \ldots, m_0-1$, on le démontre pour $m = m_0$. On a par hypothèse de récurrence :

$$\underline{\mathcal{J}}(n, C_{a_e,m_0-1}) = C_{a_s,m_0-1} \qquad (1)$$

(L400) $\Rightarrow C_{a_e,m_0-1} \bar{\bar{<}} C_{a_e,m_0}$

(L305), (L312) $\Rightarrow \underline{\mathcal{J}}(n, C_{a_e,m_0-1}) \bar{\bar{\leq}} \underline{\mathcal{J}}(n, C_{a_e,m_0})$

(1) $\Rightarrow C_{a_s,m_0-1} \bar{\bar{\leq}} \underline{\mathcal{J}}(n, C_{a_e,m_0})$ donc le test de la ligne [38] est vrai

pour les mêmes raisons que précédemment et d'après [39], on a

$$C_{a_s,m_0} = C_{a_s} = \underline{J}(n, C_{a_e,m_0}) = \underline{J}(n, C_{a_e})$$

Par récurrence sur m, le théorème est démontré dans le cas 1.2.

1.3) Le cas m > k. On traverse l'arc a_e, au moins une fois de plus que l'arc a_s. Donc, pour :

$$C_{a_e,k+1},\ C_{a_e,k+2},\ \cdots,\ C_{a_e,i},\ \cdots,\ C_{a_e,m}$$

le test de la ligne [38] est faux, c'est-à-dire :

$$\neg(\neg(\underline{J}(n, C_{a_e,i}) \overline{\leq} C_{a_s,k})),\ \forall i \in [k+1,m]$$

$$\Longrightarrow \underline{J}(n, C_{a_e,i}) \overline{\leq} C_{a_s,k} = C_{a_s},\ \forall i \in [k+1,m] \qquad (2)$$

D'après le cas 1.1, on a :

$$C_{a_s} = C_{a_s,k} = \underline{J}(n, C_{a_e,k}) \qquad (3)$$

(L400) $\Longrightarrow C_{a_e,k} < C_{a_e,i},\ \forall i \in [k+1, m]$

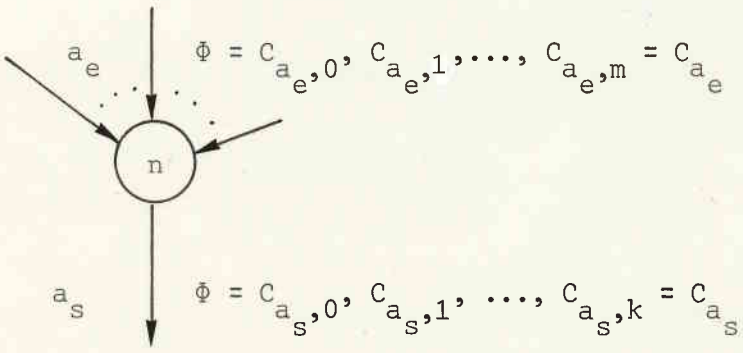
(L305) ou (L312) $\Longrightarrow \underline{J}(n, C_{a_e,k}) \leq \underline{J}(n, C_{a_e,i})$

(3) $\Longrightarrow C_{a_s} \leq \underline{J}(n, C_{a_e,i}),\ \forall i \in [k+1, m]$ (4)

(2)(4) $\Longrightarrow C_{a_s} = \underline{J}(n, C_{a_e,i}),\ \forall i \in [k+1, m]$

2°) Le noeud n est un noeud de jonction. $n \in N_{j_s} \cup N_{j_b}$.

On a la configuration :



2.1) Le cas m = k, correspond au fait que les arcs a_e et a_s ont été traversés un nombre égal de fois.
Raisonnons par récurrence sur m :

2.1.1.) m = k = 1

Posons CS' = $\bar{\bar{\cup}}$ contexte local (prédécesseur)

prédécesseur $\in$ {arc d'entrée (n) - a_e}

Posons CS = $C_{a_e,1} \bar{\bar{\cup}}$ CS'

Comme on traverse l'arc a_s après avoir traversé a_e, c'est que le test de la ligne [23] est vrai. Soit :

(5) $\neg(CS \bar{\bar{\leq}} C_{a_s,0})$, ce qui entraine d'après les remarques qui suivent le texte de la procédure d'interprétation abstraite que le test de la ligne [38] est vrai.

2.1.1.1) $n \in N_{j_s}$, noeud de jonction simple, et d'après les lignes [38] et [27]:

$$C_{a_s,1} = CS = C_{a_e,1} \bar{\bar{\cup}} CS' \qquad (6)$$

Donc :

$$C_{a_e,1} \bar{\bar{\cup}} C_{a_s,1} = C_{a_e,1} \bar{\bar{\cup}} (C_{a_e,1} \bar{\bar{\cup}} CS')$$

(L108,1) $\Longrightarrow$ $= (C_{a_e,1} \bar{\bar{\cup}} C_{a_e,1}) \bar{\bar{\cup}} CS'$

(L108,4) $\Longrightarrow$ $= C_{a_e,1} \bar{\bar{\cup}} CS'$

(6) $\Longrightarrow$ $= C_{a_s,1}$

(DEF 103) $\Longrightarrow C_{a_e,1} \bar{\bar{\leq}} C_{a_s,1}$

2.1.1.2) $n \in N_{j_b}$, noeud de jonction de boucle, et d'après les lignes [39] et [30] :

$$C_{a_s,1} = (C_{a_s,0} \bar{\bar{\nabla}} CS) \qquad (7)$$

Mais :

(L115) $\Longrightarrow (C_{a_s,0} \bar{\bar{\cup}} CS) \bar{\bar{\leq}} (C_{a_s,0} \bar{\bar{\nabla}} CS)$

donc $(C_{a_s,0} \bar{\bar{\cup}} CS) \bar{\bar{\leq}} C_{a_s,1}$

$(C_{a_s,0} = \Phi), (CS = C_{a_e,1} \bar{\bar{\cup}} CS') \Longrightarrow C_{a_e,1} \bar{\bar{\cup}} CS' \bar{\bar{\leq}} C_{a_s,1}$ (8)

(P2) $\Longrightarrow C_{a_e,1} \bar{\bar{\leq}} C_{a_e,1} \bar{\bar{\cup}} CS'$ (9)

(8), (9), (P1) $\Longrightarrow C_{a_e,1} \bar{\bar{\leq}} C_{a_s,1}$

2.1.2) $(m \geq 2) \wedge (m = k)$:

Supposons le théorème vrai pour $m = 1, \ldots, m_0 - 1$; on le démontre pour $m = m_0$. On a par hypothèse de récurrence :

$$C_{a_e,m_0-1} \bar{\bar{\leq}} C_{a_s,m_0-1}$$

Posons CS' = $\bar{\bar{\cup}}$ contexte local (prédecesseur)

prédecesseur $\in$ {arc d'entrée (n) - a_e}

$CS = C_{a_e,m_0} \bar{\bar{\cup}} CS'$

Comme on traverse l'arc a_s,m_0 après avoir traversé a_e,m_0, c'est que le test de la ligne [23] est vrai.

Soit $\neg(CS \bar{\bar{\leq}} C_{a_s,m_0-1})$ ce qui entraîne d'après les remarques qui suivent le test de la procédure d'interprétation abstraite que le test en [38] est vrai

2.1.2.1.) $n \in N_{j_s}$, noeud de jonction simple et d'après les lignes [39] et [27], on a :

$C_{a_s,m_0} = CS = C_{a_e,m_0} \bar{\bar{\cup}} CS'$ (10)

donc :

$C_{a_e,m_0} \bar{\bar{\cup}} C_{a_s,m_0} = C_{a_e,m_0} \bar{\bar{\cup}} (C_{a_e,m_0} \bar{\bar{\cup}} CS')$

(L108,1) $\Longrightarrow = (C_{a_e,m_0} \bar{\bar{\cup}} C_{a_e,m_0}) \bar{\bar{\cup}} CS'$

(L108,4) $\Longrightarrow$ $= C_{a_e,m_0} \bar{\bar{\cup}} CS'$

(10) $\Longrightarrow$ $= C_{a_s,m_0}$

$\Longrightarrow C_{a_e,m_0} \bar{\bar{\leq}} C_{a_s,m_0}$

2.1.2.2.) $n \in N_{j_b}$, noeud de jonction de boucle et d'après les lignes [39] et [30] :

$$C_{a_s,m_0} = (C_{a_s,m_0} \bar{\bar{\nabla}} CS)$$

(L115) $\Longrightarrow (C_{a_s,m_0-1} \bar{\bar{\cup}} CS) \bar{\bar{\leq}} (C_{a_s,m_0-1} \bar{\bar{\nabla}} CS)$

donc $C_{a_s,m_0-1} \bar{\bar{\cup}} CS \bar{\bar{\leq}} C_{a_s,m_0}$

$C_{a_s,m_0-1} \bar{\bar{\cup}} C_{a_e,m_0} \bar{\bar{\cup}} CS' \bar{\bar{\leq}} C_{a_s,m_0}$

$C_{a_s,m_0} \bar{\bar{\cup}} (C_{a_s,m_0-1} \bar{\bar{\cup}} CS') \bar{\bar{\leq}} C_{a_s,m_0}$ (12)

P2 $\Longrightarrow C_{a_e,m_0} \bar{\bar{\leq}} C_{a_e,m_0} \bar{\bar{\cup}} (C_{a_s,m_0-1} \bar{\bar{\cup}} CS')$

(12), (P2) et (P1) $\Longrightarrow C_{a_e,m_0} \bar{\bar{\leq}} C_{a_s,m_0}$

Par récurrence sur m, le théorème est démontré dans le cas 2.1

2.2 $n < k$.

On a traversé l'arc a_s au moins une fois de plus que l'arc a_e. Donc pour

$$C_{a_s,m+1} ; C_{a_s,m+2} ; \cdots ; C_{a_s,i} ; \cdots ; C_{a_s,k}$$

on a :

(L400) $C_{a_s,m} < C_{a_s,i}$, $\forall i \in [m+1, k]$ (13)

et d'après le cas 2.1.1., on a :

$$C_{a_e,m} \leq C_{a_s,m} \quad (14)$$

(13), (14), (P1) $\Longrightarrow C_{a_e} = C_{a_e,m} \leq C_{a_s,i}$, $\forall i \in [m+1, k]$

$\Longrightarrow C_{a_e} \leq C_{a_s}$ pour i = k ci dessus.

2.3) $n > k$.

On a traversé l'arc a_e au moins une fois de plus que l'arc a_s. Donc pour :

$$C_{a_e,k+1} \; ; \; C_{a_e,k+2} \; ; \; \cdots \; ; \; C_{a_e,i} \; ; \; \cdots \; ; \; C_{a_e,m} \; ; \text{ on a}$$

en appelant $CS = C_{a_e,i} \bar{\cup} CS'$ le contexte de sortie défini à la ligne [22], on a d'après la ligne [23] :

$$\neg(\neg(CS \bar{\leq} C_{a_s,k}))$$

$$\Longrightarrow \quad (C_{a_e,i} \bar{\cup} CS') \bar{\leq} C_{a_s,k} = C_{a_s} \qquad (15)$$

$$(P2) \Longrightarrow C_{a_e,i} \bar{\leq} (C_{a_e,i} \bar{\cup} CS') \qquad (16)$$

$$(15), (16), (P1) \Longrightarrow C_{a_e,i} \bar{\leq} C_{a_s}, \; \forall i \in [k+1, m]$$

$$\Longrightarrow C_{a_e} \bar{\leq} C_{a_s} \quad \text{pour } i = m$$

Le lemme est démontré dans tous les cas.

Nous allons établir la correction de l'algorithme d'interprétation abstraite. Quel que soit P un programme concret, notons :

$$P_{(a_1,\ldots,a_m)}(i)$$

la valeur de la variable i sur l'arc a_m, à la suite d'une exécution quelconque du programme P, le long du chemin $(a_1, \ldots, a_m)$, à partir de l'arc initial a_1 de P.

Notons C_{a_m}, le contexte local associé à l'arc a_m du graphe de P, par la procédure d'interprétation abstraite de P. La correction de la procédure d'interprétation découle du théorème suivant :

Théorème - T502

$\forall (a_1, \ldots, a_m), \; \forall i \in I :$

$$P_{(a_1, \ldots, a_m)}(i) \in \gamma(C_{a_m}(i))$$

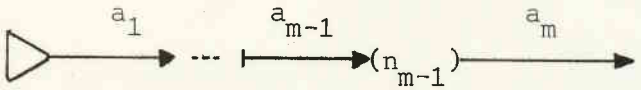
1°) m = 1

C_{a_1} = Φ, donc $C_{a_1}(i)$ = □, ∀i ∈ I

$P_{(a_1)}(i) \in \gamma(\square)$ car i n'est pas initialisé.

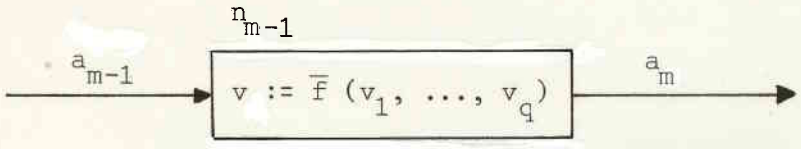
2°) m > 1

Supposons le théorème établi jusqu'à m-1 ce qui permet de l'établir pour m :



La démonstration dépend de la nature du noeud n_{m-1} et de la nature de l'arc a_m.

2.1) Affectation, $n_{m-1} \in N_a$



Par hypothèse de récurrence, on a :

$$\{\forall i \in I,\ P_{(a_1, \ldots, a_{m-1})}(i) \in \gamma(C_{a_{m-1}}(i))\} \qquad (1)$$

2.1.1.) Si i ≠ v, alors :

$P_{(a_1, \ldots, a_{m-1})}(i) = P_{(a_1, \ldots, a_m)}(i)$, d'après l'hypothèse (H212-2)

(L501) $\Rightarrow$ $\mathcal{J}_a(n_{m-1}, C_{a_{m-1}}) = C_{a_m}$ (2)

(L111) $\Rightarrow$ $\mathcal{J}_a(n_{m-1}, C_{a_{m-1}})(i) = C_{a_m}(i)$

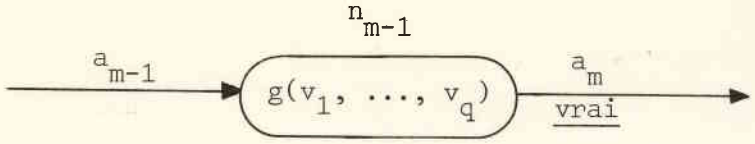
(H300-2) $\Rightarrow$ $C_{a_{m-1}}(i) = C_{a_m}(i)$ (3)

(1), (3) $\Rightarrow$ $P_{(a_1, \ldots, a_{m-1})}(i) \in \gamma(C_{a_m}(i))$ (4)

(2), (4) $\Rightarrow$ $P_{(a_1, \ldots, a_m)}(i) \in \gamma(C_{a_m}(i))$

2.2) Test, $n_{m-1} \in N_t$

(les deux cas, a_n arc de sortie vrai et a_n arc de sortie faux, se traitent de la même manière). On donne une démonstration pour le premier cas.



On a par hypothèse de récurrence :

$$\{\forall i \in I,\ P'_{(a_1, \ldots, a_{m-1})}(i) \in \gamma(C_{a_{m-1}}(i))\}$$

et on veut démontrer :

$$\{P_{(a_1, \ldots, a_m)}(i) \in \gamma(C_{a_m}(i))\}$$

2.2.1.) $C_{a_{m-1}}(i) \neq \square$

Posons $(C_{a_m}, C_F) = \underline{\mathfrak{I}}_t(n_{m-1}, C_{a_{m-1}})$. D'après la définition (H307-2) de l'interprétation abstraite, on a :

$\forall i \in I$, $C_{a_m}(i) = @(E)$, en posant

$$E = \{\tau \mid (\tau \in \gamma(C_{a_{m-1}}(i))) \wedge (\exists (\nu_1, \ldots, \nu_q) \in \gamma(C_{a_{m-1}}(v_1)) \times \ldots \times \gamma(C_{a_{m-1}}(v_q)) \mid g(\nu_1, \ldots, \nu_q))\}$$

2.2.1.1.) $\exists v_j,\ j \in [1,q] \mid C_{a_{m-1}}(v_j) = \square$.

On a par hypothèse de récurrence, $P_{(a_1, \ldots, a_{m-1})}(v_j) \in \gamma(\square)$. L'élaboration du test g conduit à un résultat indéfini, le programme est incorrect, ce qui se traduit par $P'_{(a_1, \ldots, a_m)}(i) \in \gamma(\square)$. D'autre part, ν_j est toujours indéfini, car $\nu_j \in \gamma(\square)$, en conséquence $g(\nu_1, \ldots, \nu_j, \ldots, \nu_q)$ est indéfini et l'ensemble E est vide, donc $C_{a_m}(i) = \square$.

2.2.1.2.) $\forall v_j,\ j \in [1,q]$, $C_{a_{m-1}}(v_j) \neq \square$. Comme on a suivi l'arc a_m, branche vrai du test, on a

$$g(P_{(a_1, \ldots, a_m)}(v_1), \ldots, P_{(a_1, \ldots, a_m)}(v_q)) = \text{vrai} \qquad (1)$$

D'autre part, d'après l'hypothèse de récurrence

$\forall i \in I,\ P_{(a_1, \ldots, a_{m-1})}(i) \in \gamma(C_{a_{m-1}}(i))$ (2)

On a (1) $\wedge$ (2) $\Longrightarrow P_{(a_1, \ldots, a_{m-1})}(i) \in E$ (3)

D'après (H214-2), g n'a pas d'effets de bord, donc

$P_{(a_1, \ldots, a_m)}(i) = P_{(a_1, \ldots, a_{m-1})}(i)$

(3) $\Longrightarrow\ P_{(a_1, \ldots, a_m)}(i) \in E$

(H5) $\Longrightarrow P_{(a_1, \ldots, a_m)}(i) \in \gamma(@(E))$

$\Longrightarrow P_{(a_1, \ldots, a_m)}(i) \in \gamma(C_{a_m}(i))$

2.2.2.) $C_{a_{m-1}}(i) = \square$

i n'est pas défini dans le contexte d'entrée. Comme l'évaluation de l'expression booléenne $g(v_1, \ldots, v_q)$ n'a pas d'effets de bord (H214-2), elle n'affecte pas de valeur à i, qui n'est donc pas défini dans le contexte de sortie. Par conséquent, comme $P_{(a_1, \ldots, a_m)}(i) \in \gamma(\square)$, on est conduit à montrer que $C_{a_m}(i) = \square$.

(H307-2) $\Longrightarrow C_{a_m}(i) = \underline{\mathfrak{J}}_t(n_{m-1}, C_{a_{m-1}})\ [1]\ (i) = @(E)$

en posant

$E = \{\tau \mid (\tau \in \gamma(\square)) \wedge (\exists\ (v_1, \ldots, v_q) \in \gamma(C_{a_{m-1}}(v_1)) \times \ldots \times \gamma(C_{a_{m-1}}(v_q)) \mid g(v_1, \ldots, v_q))\}$

$\subseteq \{\tau \mid (\tau \in \gamma(\square))\}$

$\subseteq \gamma(\square)$.

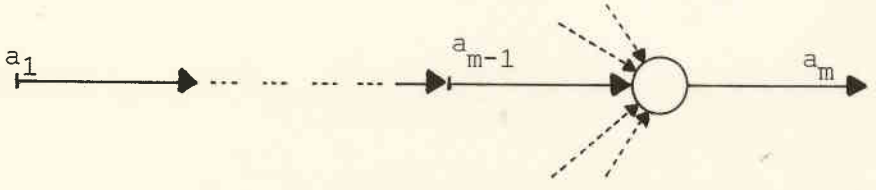
D'après (H 3), on a $@(E) \underline{\leq} @(\gamma(\square))$

(H 6) $\Longrightarrow\ @(E) \underline{\leq} \square$

D'autre part, (P7) $\Longrightarrow \square \underline{\leq} @(E)$

soit $@(E) = \square$ et $C_{a_m}(i) = \square$.

3°) Noeud de jonction, $n_{m-1} \in N_{j_s} \cup N_{j_b}$



Par hypothèse de récurrence, on a :

$$\forall i \in I,\ P_{(a_1, \ldots, a_{m-1})}(i) \in \gamma(C_{a_{m-1}}(i)) \qquad (11)$$

Etant donné que la passage par un noeud de jonction ne modifie aucune variable du programme, lors d'une exécution réelle, on a :

$$P_{(a_1,\ldots, a_{m-1})}(i) = P_{(a_1,\ldots, a_m)}(i) \qquad (12)$$

3.1) $C_{a_{m-1}}(i) = \square$ (13)

(11), (13), (12) $\Rightarrow P_{(a_1, \ldots, a_m)}(i) \in \gamma(\square)$

(L15) $\Rightarrow P_{(a_1, \ldots, a_m)}(i) \in \gamma(\square \bar{\cup} C_{a_m}(i))$

(L8) $\Rightarrow P_{(a_1,\ldots, a_m)}(i) \in \gamma(C_{a_m}(i))$

3.2) $C_{a_{m-1}}(i) \neq \square$

(L501) & (L116) $\Rightarrow C_{a_{m-1}}(i) \bar{\leq} C_{a_m}(i)$

(1), (L16) $\Rightarrow P_{(a_1, ——, a_{m-1})}(i) \in \gamma(C_{a_m}(i))$

(12) $\Rightarrow P_{(a_1, \ldots, a_m)}(i) \in \gamma(C_{a_m}(i)$

Finalement, on a montré que :

$P_{(a_1, \ldots, a_{m-1})}(i) \in \gamma(C_{a_{m-1}}(i))$ entraîne que

$P_{(a_1, \ldots, a_m)}(i) \in \gamma(C_{a_{m-1}}(i))$

Par récurrence sur m, le résultat est vrai pour toute valeur de m.

4°) - EXEMPLES DE CORRESPONDANCES ENTRE VALEURS CONCRETES ET VALEURS ABSTRAITES, POUR LES DONNEES COURANTES DES LANGAGES DE PROGRAMMATION -

Les données et opérations élémentaires de l'algorithme d'interprétation abstraite sont définies en fonction des propriétés des programmes que l'on veut étudier. Nous présentons dans ce chapitre quelques exemples d'études de programmes, concernant les intervalles de variation de leurs valeurs entières, la parité des valeurs entières, la bonne définition d'accès à des objets par l'intermédiaire de pointeurs, etc ....

4.1 - Intervalle de variation des valeurs entières d'un programme -

Dans les langages de programmation, les "entiers" sont en fait un sous-ensemble $\overline{\mathbb{N}}$ des entiers de $\mathbb{N}$ compris entre deux bornes, que nous noterons symboliquement $+\infty$ et $-\infty$.

4.1.1) - L'ensemble V_c des valeurs concrètes est donc l'ensemble des entiers compris entre $-\infty$ et $+\infty$: $V_c = \overline{\mathbb{N}}$

4.1.2) - L'ensemble V_a des valeurs abstraites est l'ensemble des intervalles fermés de $\mathbb{N}$, c'est-à-dire

$$V_a = \{[n,m] \mid (n \in V_c) \wedge (m \in V_c) \wedge (n \leq m)\} \cup \{\square\}$$

4.1.3) - L'opération @ d'abstraction d'ensembles de valeurs abstraites fait correspondre à tout ensemble d'entiers un intervalle fermé contenant tous ces entiers :

$$\{\forall X \subset V_c,$$
$$@(X) = \underline{\text{cas}}$$
$$\quad \vdash X = \emptyset \longmapsto \square \ ;$$
$$\quad \vdash X \neq \emptyset \longmapsto \left[\underline{\underline{MIN}}_{x \in X}(x) \ , \ \underline{\underline{MAX}}_{x \in X}(x)\right] \ ;$$
$$\quad \underline{\text{fincas}} \ ; \qquad \}$$

Il est évident que l'opération @ satisfait la définition (DEF1) d'après 4.1.2 et $\underline{\underline{MIN}}_{x \in X}(x) \leq \underline{\underline{MAX}}_{x \in X}(x)$

4.1.4) - L'union $\bar{\cup}$ de valeurs abstraites est définie comme suit :

$\forall (X,Y) \in V_a^2$:

$X \bar{\cup} Y =$ cas X, Y dans

- $\square, \square \rightarrow \square$;
- $\square, [n,m] \rightarrow [n,m]$;
- $[n,m], \square \rightarrow [n,m]$;
- $[n_1,m_1],[n_2,m_2] \rightarrow [\min(n_1,n_2),\max(m_1,m_2)]$

fincas ; }

Cette opération satisfait la définition DEF2, car $(n_1 \leq m_1) \wedge (n_2 \leq m_2) \Rightarrow \underline{\min}(n_1,n_2) \leq \underline{\max}(m_1,m_2)$, et elle est définie pour toutes les valeurs abstraites.

4.1.5) - Montrons que @ est un homomorphisme de $(2^{V_c},\cup)$ dans $(V_a,\bar{\cup})$:

$\{\forall (V_1,V_2) \mid V_1 \subseteq V_c \; ; \; V_2 \subseteq V_c$:

$@ (V_1 \cup V_2) =$

cas V_1, V_2 dans

- $\emptyset, \emptyset \Rightarrow @ (\emptyset \cup \emptyset) = @ (\emptyset) = \square = \square \bar{\cup} \square$
  $= @ (\emptyset) \bar{\cup} @ (\emptyset) = @ (V_1) \bar{\cup} @ (V_2)$;
- $\emptyset, \neq \emptyset \Rightarrow @ (\emptyset \cup V_2) = @ (V_2) = \square \bar{\cup} @ (V_2)$
  $= @ (\emptyset) \bar{\cup} @ (V_2) = @ (V_1) \bar{\cup} @ (V_2)$;
- $\neq \emptyset, \emptyset \Rightarrow @ (V_1 \cup \emptyset) = @ (V_1) \bar{\cup} @ (V_2)$;

d'après (4.1.3) : $\neq \emptyset, \neq \emptyset \Rightarrow @(V_1 \cup V_2) = \left[\underset{x \in V_1 \cup V_2}{\underline{MIN}(x)} , \underset{y \in V_1 \cup V_2}{\underline{MAX}(y)} \right]$

$$= \left[\underline{\min} \left(\underset{x \in V_1}{\underline{MIN}(x)} , \underset{y \in V_2}{\underline{MIN}(y)} \right) , \underline{\max} \left(\underset{x \in V_1}{\underline{MAX}(x)} , \underset{y \in V_2}{\underline{MAX}(y)} \right) \right]$$

(4.1.4) $\Rightarrow$

$$= \left[\underset{x \in V_1}{\underline{MIN}(x)} , \underset{y \in V_1}{\underline{MAX}(y)} \right] \bar{\cup} \left[\underset{x \in V_2}{\underline{MIN}(x)} , \underset{y \in V_2}{\underline{MAX}(y)} \right]$$

(4.1.3) $\Rightarrow$

$= @ (V_1) \bar{\cup} @ (V_2)$;

fincas ;

4.1.6) - L'opération γ de concrétisation de valeurs abstraites fait correspondre à un intervalle, l'ensemble des entiers compris dans cet intervalle :

$\{\forall x \in V_a$;

$\gamma(x) =$ cas x dans
| $\square \Longrightarrow \emptyset$;
| $[n,m] \Longrightarrow (e \mid n \leq e \leq m)$;
fincas ; $\}$

La définition de γ satisfait (DEF.4).

4.1.7) - Il est trivial de montrer que d'après leurs définitions, γ et @ satisfont les hypothèses (H5) et (H6)

(H5)
Cas $v \subseteq V_c$ dans
| $\emptyset \Rightarrow v \subseteq \gamma(@(v)) = \emptyset$;
| $\nexists \emptyset \Rightarrow v \subseteq \gamma(@(v)) = \{e \mid \underset{x \in v}{\underline{MIN}}(x) \leq e \leq \underset{y \in v}{\underline{MAX}}(y)\}$;
fincas ;

(H6)
Cas $v \in V_a$ dans
| $\square \Rightarrow \square = @(\gamma(\square))$;
| $[n,m] \Rightarrow [n,m] = @(\gamma([n,m]))$;
fincas ;

4.1.8) - La comparaison de valeurs abstraites, selon (DEF12) est :

$\forall (x,y) \in V_a^2$

$\{x \bar{\leq} y\} \Leftrightarrow \{x \bar{\cup} y = y\}$

$\{x \bar{\leq} y\} \Leftrightarrow$ cas x, y dans
| $\square, \square \Rightarrow$ true ;
| $\square, \nexists \square \Rightarrow$ true ;
| $\nexists \square, \square \Rightarrow$ false ;
| $[n_1,m_1],[n_2,m_2] \Rightarrow [\underline{min}(n_1,n_2), \underline{max}(m_1,m_2)] = [n_2,m_2]$
$\Leftrightarrow (\underline{min}(n_1,n_2) = n_2) \wedge (\underline{max}(m_1,m_2) = m_2)$
$\Leftrightarrow (n_2 \leq n_1) \wedge (m_2 \geq m_1)$;
fincas ;

$\{x \bar{\leq} y\} \Longleftrightarrow$ cas $x \in V_a$, $y \in V_a$ dans
- $\square$, ? $\Rightarrow$ true ;
- $\nmid\square$, $\square$ $\Rightarrow$ false ;
- $[n_1,m_1],[n_2,m_2] \Rightarrow (n_2 \leq n_1) \wedge (m_2 \geq m_1)$;

fincas ;

4.1.9) - Comme $\{x \bar{<} y\} \Longleftrightarrow \{x \bar{\leq} y\} \wedge \{x \neq y\}$, on peut également démontrer :

$\{x \bar{<} y\} \Longleftrightarrow$ cas $x \in V_a$, $y \in V_a$ dans
- ?, $\square$ $\Rightarrow$ false ;
- $\square$, $\nmid\square$ $\Rightarrow$ true ;
- $[n_1,m_1],[n_2,m_2] \Rightarrow$
  $\{(n_2 < n_1) \wedge (m_2 \geq m_1)\} \vee \{(n_2 \leq n_1) \wedge (m_2 > m_1)\}$;

fincas ;

4.1.10) - L'élargissement $\bar{\nabla}$ de valeurs abstraites est défini par :

$(x \bar{\nabla} y) =$ cas $x \in V_a$, $y \in V_a$ dans
- $\square$, ? $\Rightarrow$ y ;
- ?, $\square$ $\Rightarrow$ x ;
- $[n_1,m_1],[n_2,m_2] \Rightarrow$
  [si $n_2 < n_1$ alors $-\infty$ sinon n_1 fsi ;
  si $m_2 > m_1$ alors $+\infty$ sinon m_1 fsi] ;

fincas ;

La définition de $\bar{\nabla}$ satisfait évidemment (DEF 14).

4.1.11) - Montrons que les définitions de $\bar{\cup}$ et $\bar{\nabla}$ satisfont l'hypothèse (H14) :
$(x \bar{\cup} y) \bar{\leq} (x \bar{\nabla} y)$

cas $x \in V_a$, $y \in V_a$ dans
- $\square$, $\square$ $\Longrightarrow$ $\square \bar{\cup} \square \bar{\leq} \square \bar{\nabla} \square$ car $\square \bar{\leq} \square$
- $\square$, [n,m] $\Longrightarrow$ $\square \bar{\cup}$ [n,m] = [n,m]
  $\bar{\leq} \square \bar{\nabla}$ [n,m] = [n,m]
- [n,m], $\square$ $\Longrightarrow$ [n,m] $\bar{\cup} \square$ = [n,m] $\bar{\leq}$ [n,m] $\bar{\nabla} \square$ = [n,m]
- $[n_1,m_1]$, $[n_2,m_2]$ $\Longrightarrow$

$x \bar{\cup} y = [\underline{\min}(n_1,n_2), \underline{\max}(m_1,m_2)]$;

$x \bar{\nabla} y = [\underline{si}\ n_2 < n_1\ \underline{alors}\ -\infty\ \underline{sinon}\ n_1\ \underline{fsi}$;

$\underline{si}\ m_2 > m_1\ \underline{alors}\ +\infty\ \underline{sinon}\ m_1\ \underline{fsi}]$;

d'après (4.1.8) : $\{x \bar{\cup} y \bar{\leq} x \bar{\nabla} y\} \Leftrightarrow$

$(\underline{si}\ n_2 < n_1\ \underline{alors}\ -\infty\ \underline{sinon}\ n_1\ \underline{finsi} \leq \underline{\min}\ (n_1,n_2))$

$\wedge\ (\underline{si}\ m_2 > m_1\ \underline{alors}\ +\infty\ \underline{sinon}\ m_1\ \underline{finsi} \geq \underline{\max}\ (m_1,m_2))$;

$\Leftrightarrow (\underline{si}\ n_2 < n_1\ \underline{alors}\ -\infty \leq n_2\ \underline{sinon}\ n_1 \leq n_1\ \underline{finsi})$

$\wedge\ (\underline{si}\ m_2 > m_1\ \underline{alors}\ +\infty \geq m_2\ \underline{sinon}\ m_1 \geq m_1\ \underline{finsi})$;

$\underline{fincas}$;

4.1.12) - Nous devons montrer que toute suite de valeurs abstraites s_o, ..., s_n, ..., construite à partir de la suite infinie v_1, ..., v_m, ..., par la relation de récurrence $s_o = \square$; $s_n = s_{n-1} \bar{\nabla} v_n$, ..., n'est pas infinie et strictement croissante pour la relation $\bar{<}$:

$s_o \bar{<} s_1 = s_o \bar{\nabla} v_1 \quad \Rightarrow$ (4.1.12)

$\square < \square \bar{\nabla} v_1 \quad \Rightarrow$ (4.1.10)

$\square < v_1 \quad \Rightarrow$ (4.1.9)

$v_1 = [n_1,m_1]$

$s_1 \bar{<} s_2 \quad \Rightarrow$ (4.1.12)

$v_1 \bar{<} v_1 \bar{\nabla} v_2 \quad \Rightarrow$ (P4), (PDEF2)

$[n_1,m_1] \nmid [n_1,m_1] \bar{\nabla} [n_2,m_2] \Rightarrow$ (4.1.10)

$(n_1 \nmid \underline{si}\ n_2 < n_1\ \underline{alors}\ -\infty\ \underline{sinon}\ n_1\ \underline{fsi}) \vee (m_1 \nmid \underline{si}\ m_2 > m_1\ \underline{alors}\ +\infty\ \underline{sinon}$

$\Rightarrow \quad (n_2 < n_1) \vee (m_2 > m_1) \qquad m_1\ \underline{fsi})$

$\underline{cas\ 1}$: $(n_2 < n_1) \wedge (m_2 > m_1)$

On a $s_2 = [-\infty,+\infty]$, donc quel que soit v_3

On a $s_3 = s_2 \bar{\nabla} v_3 = [-\infty,+\infty] = s_2$, donc la suite est finie

$\underline{cas\ 2}$: $(n_2 < n_1) \wedge (m_2 \leq m_1)$

On a $s_2 = [-\infty,m_1]$

$s_2 < s_3 \quad \Rightarrow$ (4.1.12)

$[-\infty,m_1] < [-\infty,m_1] \bar{\nabla} v_3 \Rightarrow$ (4.1.9)

$[-\infty,m_1] < [-\infty,m_1] \bar{\nabla} [n_3,m_3]$ ⟹ (4.1.10)

$< [-\infty,$ si $m_3 > m_1$ alors $+\infty$ sinon m_1 fsi$]$

⟹ (PDEF2) $m_1 \nmid$ (si $m_3 > m_1$ alors $+\infty$ sinon m_1)

⟹ $m_3 > m_1$

⟹ $s_3 = [-\infty,+\infty]$ et s_4 ne peut être qu'égal à s_3

cas 3 : $(n_1 \leq n_2) \wedge (m_2 > m_1)$

On a $s_2 = [n_1,+\infty]$

$s_2 < s_3$ ⟹ (4.1.12)

$[n_1,+\infty] < [n_1,+\infty] \bar{\nabla} [n_3,m_3]$ ⟹ (4.1.10), (PDEF2)

$n_1 \nmid$ si $n_3 < n_1$ alors $-\infty$ sinon n_1 fsi

⟹ $n_3 < n_1$

⟹ $s_3 = [-\infty,+\infty]$ et dans ce cas s_4 est égal à s_3 quel que soit v_4.

En conclusion, la série s_0, s_1, ... a au plus quatre termes strictement croissants.

4.1.12) - Arithmétique sur les intervalles -

4.1.12.1) Considérons l'interprétation abstraite élémentaire d'un noeud affectation de la forme :

$$v := \bar{f}(v_1, \ldots, v_m).$$

On a vu que $\forall i \in I, \forall C \in \mathcal{C}$, $i \nmid v$ ⟹ (H300-2)

$$\mathfrak{I}_a(n,C)(i) = C(i)$$

et pour $i = v$, on a : (H300-3) :

$$\mathfrak{I}_a(n,C)(v) = @[\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m))\}]$$

D'après (H212-1), on a vu que si un des identificateurs $(v_1,\ldots,v_m)$ a la valeur $\Box$ dans le contexte C ($m > 1$), alors $\mathfrak{I}_a(n,C)(v) = \Box$. (Voir L306).
Au contraire, dans le cas général, on a d'après (4.1.3) :

(1) $\mathfrak{I}_a(n,c)(v) = [MIN(\{f(\nu_1,\ldots,\nu_m) \mid (\nu_1,\ldots,\nu_m) \in \gamma(C(v_1))\times\ldots\times\gamma(C(v_m))\}),$

MAX$(\{f(\nu_1,\ldots,\nu_m) \mid (\nu_1,\ldots,\nu_m) \in \gamma(C(v_1))\times\ldots\times\gamma(C(v_m))\})]$

Dans un langage comme PASCAL, $\bar{f}(v_1,\ldots,v_n)$ dénote une expression arithmétique, obtenue par composition de fonctions élémentaires (+, -, *, ...) appliquées aux valeurs des variables (v_1, ..., v_m) et de constantes. Afin de calculer simplement la formule du membre droit de (1) pour toute expression $f(v_1,\ldots,v_m)$, nous allons développer une arithmétique sur les intervalles.

4.1.12.2) A toute expression arithmétique $f(v_1,\ldots,v_n)$ définie dans un contexte abstrait C, on associe une expression arithmétique sur les intervalles, notée $\overline{f(v_1,\ldots,v_m),C}$ comme suit :

a) - si $f(v_1,\ldots,v_m)$ = cte (expression réduite à une constante) :

$$\overline{f(v_1,\ldots,v_m),C} = [cte,cte]$$

b) - si $f(v_1,\ldots,v_m) = v_k$ (expression réduite à une variable) :

$$\overline{f(v_1,\ldots,v_m),C} = C(v_k)$$

c) - si $f(v_1,\ldots,v_m) = \underline{op}(expr_1(v_1,\ldots,v_m), expr_2(v_1,\ldots,v_m))$
(expression arithmétique obtenue par application d'un opérateur élémentaire $\underline{op}$ unaire ou binaire ($\underline{op}$ = +, -, *, ...) à une ou deux expressions arithmétiques $expr_1$ et $expr_2$) :

$$\overline{f(v_1,\ldots,v_n),C} = \overline{\underline{op}}\left(\overline{expr_1(v_1,\ldots,v_m),C}, \overline{expr_2(v_1,\ldots,v_m),C}\right)$$

où :

$$\overline{\underline{op}}([i_1,s_1],[i_2,s_2]) = [i,s]$$

$$= \begin{bmatrix} \underline{MIN}\ \{\underline{op}(x,y)\} & , & \underline{MAX}\ \{\underline{op}(x,y)\} \\ x\epsilon\gamma([i_1,s_1]) & & x\epsilon\gamma([i_1,s_1]) \\ y\epsilon\gamma([i_2,s_2]) & & x\epsilon\gamma([i_2,s_2]) \end{bmatrix}$$

4.1.12.3) Montrons maintenant que :

$$\overline{f(v_1,\ldots,v_m),C} = [\underline{MIN}(\{f(\nu_1,\ldots,\nu_m) \mid (\nu_1,\ldots,\nu_m) \in \gamma(C(v_1))\times\ldots\times\gamma(C(v_m))\}),$$
$$\underline{MAX}(\{f(\nu_1,\ldots,\nu_m) \mid (\nu_1,\ldots,\nu_m) \in \gamma(C(v_1))\times\ldots\times\gamma(C(v_m))\})]$$

- Le lemme est évident pour les constantes (4.1.12.2.a) et pour les variables simples également car (4.1.12.2.b) :

$$\begin{bmatrix} \underline{\text{MIN}}\ (v_k) & , & \underline{\text{MAX}}\ (v_k) \\ v_k \epsilon \gamma(C(v_k)) & & v_k \epsilon \gamma(C(v_k)) \end{bmatrix} = C(v_k)$$

- Supposons par hypothèse d'induction structurale que le lemme est démontré pour $\overline{expr_1(v_1,\ldots,v_m),C}$ et $\overline{expr_2(v_1,\ldots,v_m),C}$, on le démontre pour :

$$f(v_1,\ldots,v_m),C = \overline{\underline{op}}\left(\overline{expr_1(v_1,\ldots,v_m),C},\ \overline{expr_2(v_1,\ldots,v_m),C}\right)$$

4.1.12.2.c

$\Rightarrow$

$$= \begin{bmatrix} \underline{\text{MIN}}\ \{\underline{op}(x,y)\} & , & \underline{\text{MAX}}\ \{\underline{op}(x,y)\} \\ x \epsilon \gamma(\overline{expr_1(\ldots),C}) & & x \epsilon \gamma(\overline{expr_1(\ldots),C}) \\ y \epsilon \gamma(\overline{expr_2(\ldots),C}) & & y \epsilon \gamma(\overline{expr_2(\ldots),C}) \end{bmatrix}$$

Hypothèse d'induction :

$\Rightarrow$

$$= \begin{bmatrix} \underline{\text{MIN}}\ \{\underline{op}(x,y)\} & , & \underline{\text{MAX}}\ \{\underline{op}(x,y)\} \\ x \epsilon \gamma([\underline{\text{MIN}}\{expr_1(v_1,\ldots,v_m) \mid & & , \underline{\text{MAX}}\{expr_1(v_1,\ldots,v_m) \mid \ ]) \\ \quad (v_1,\ldots,v_m) \epsilon \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m))\} & & \quad (\ ,\ldots,\ ) \epsilon \ldots \times \ldots \times \ldots\} \\ y \epsilon \gamma([\underline{\text{MIN}}\{expr_2(\ldots\ \ ) \mid & & , \underline{\text{MAX}}\{\ldots \mid \quad\quad]) \\ \quad (\ldots\ \ ) \epsilon \ldots \times \ldots \times \ldots\} & & \quad (\ldots) \epsilon \ldots \times \ldots \times \ldots\} \end{bmatrix}$$

En utilisant l'identité :

$$\underline{\text{MIN}}\ (\{f(v_1,\ldots,g(v_1,\ldots,v_m),\ \ldots\ v_m) \mid (v_1,\ldots,v_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m))\})$$

$$= \underline{MIN}\ (\{f(v_1, \ldots, \tau, \ldots, v_m) \mid (v_1, \ldots, \tau, \ldots, v_m) \in \gamma(C(v_1)) \times \ldots$$
$$\ldots \times \gamma([\underline{MIN}(\{g(\mu_1,\ldots,\mu_m) \mid (\mu_1,\ldots,\mu_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m))\}),$$
$$\underline{MAX}(\{g(\mu_1,\ldots,\mu_m) \mid (\mu_1,\ldots,\mu_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m))\})]) \times \ldots$$
$$\ldots \times \gamma(C(v_m))\})$$

et une identité identique pour l'opérateur $\underline{MAX}$, on obtient :

$$= \left[\begin{array}{l} \underline{MIN}\ \{\underline{op}(expr_1(v_1,\ldots,v_m),\ expr_2(v_1,\ldots,v_m)) \\ (v_1,\ldots,v_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m)) \end{array} \right. \quad , \underline{MAX}\ \{\ldots\} \left.\begin{array}{l} \\ \ldots \end{array}\right]$$

Par induction structurale sur les expressions arithmétiques, le lemme 4.1.12.3 est démontré.

4.1.12.4) Il reste maintenant à introduire pour chaque opérateur $\underline{op}$ du langage étudié, à opérandes entiers, l'opérateur $\overline{\underline{op}}$ correspondant sur les intervalles arithmétiques satisfaisant la condition énoncée en 4.1.12.2.c.

a) Additions : $i \leq s$, $j \leq t$:

$$[i,s] \bar{+} [j,t] = \left[\begin{array}{ll} \underline{MIN}(x+y) & \\ x \in \gamma([i,s]) & \\ y \in \gamma([j,t]) & \end{array}\right. , \left.\begin{array}{l} \underline{MAX}(x+y) \\ x \in \gamma([i,s]) \\ y \in \gamma([j,t]) \end{array}\right]$$

$$= \left[\begin{array}{lll} \underline{MIN}(x) & + \underline{MIN}(y) & \\ x \in \gamma[i,s] & y \in \gamma[j,t] & \end{array}\right. , \left.\begin{array}{ll} \underline{MAX}(x) & + \underline{MAX}(y) \\ x \in \gamma[i,s] & y \in \gamma[j,t] \end{array}\right]$$

$$= [i+s\ ,\ j+t]$$

b) Soustractions :

$$[i,s] - [j,t] = \left[\begin{array}{ll} \underline{MIN}(x-y) & \\ x \in \gamma([i,s]) & \\ y \in \gamma([j,t]) & \end{array}\right. , \left.\begin{array}{l} \underline{MAX}(x-y) \\ x \in \gamma([i,s]) \\ y \in \gamma([j,t]) \end{array}\right]$$

$$= \left[\begin{array}{lll} \underline{MIN}(x) & - \underline{MAX}(y) & \\ x \in \gamma([i,s]) & y \in \gamma([j,t]) & \end{array}\right. , \left.\begin{array}{ll} \underline{MAX}(x) & - \underline{MIN}(y) \\ x \in \gamma([i,s]) & y \in \gamma([j,t]) \end{array}\right]$$

$$= [i-t\ ,\ s-j]$$

Pour les opérateurs suivants, on pourra vérifier qu'ils satisfont l'hypothèse 4.1.12.2.c, sans que nous reproduisions la démonstrations.

c) Multiplication :

$[i,s] \bar{*} [j,t] = [\underline{MIN}(i*j, i*t, s*j, s*t), \underline{MAX}(i*j , i*t, s*j, s*t)]$

d) Signe : le signe d'un entier est défini par :

```
integer procedure SGN (integer value A) ;
      if A < 0 then -1 else +1 ;
```

Sur les intervalles, on définira :

```
SGN ([i,s]) = si i ≥ 0 alors
                   [1,1]
              sinonsi s < 0 alors
                   [-1,-1]
              sinon
                   [-1,1]
              finsi ;
```

e) Valeur absolue :

```
abs ([i,s]) = si i ≥ 0 alors
                  [i,s]
              sinonsi j ≤ 0 alors
                  [-s,-i]
              sinon
                  [0, max(abs(i),abs(s))]
              finsi ;
```

f) Division : Par définition, on a :

A div B = SGN (A×B) × D(abs(A), abs(B))

avec

```
integer procedure D (integer value A, B) ;
    if A < B then 0 else D(A-B,B) + 1 ;
```

On peut donc définir :

```
[i,s] div [j,t] =

si [j,t] = [0,0] alors

     □

sinonsi SGN([i,s]) ⊼ SGN([j,t]) = [1,1] alors
       [MIN(abs(i), abs(s)) div max(abs(j), abs(t)),
           max(abs(i), abs(s)) div min(abs(j), abs(t))]
     sinonsi SGN([i,s]) ⊼ SGN([j,t]) = [-1,-1] alors
         [-max(abs(i), abs(s)) div min(abs(j), abs(t)),
            -min(abs(i), abs(s)) div max(abs(j), abs(t))]
     sinon
         [-max(abs(i), abs(s)) div min(abs(j), abs(t)),
            + max(abs(i), abs(s)) div min(abs(j), abs(t))]
finsi ;
```

g) Modulo : Par définition, on a :

A modulo B = A - (A div B) * B
donc SGN (A modulo B) = SGN (A)
Ce qui permet de définir :

```
  [i,s] modulo [j,t] =

     si [j,t] = [0,0] alors
           □
     sinonsi SGN ([i,s]) = [1,1] alors
            [0, max(abs(j), abs(t))]
     sinonsi SGN ([i,s]) = [-1,-1] alors
            [-max(abs(j), abs(t)), 0]
     sinon
          [-max(abs(j), abs(t)), max(abs(j), abs(t)]
     finsi ;
```

4.1.12.5) Remarque :
On peut décrire l'opération modulo de façon plus précise, en tenant compte du fait que
[0, 5] modulo [7, 10] = [0, 5], par exemple.

Avec la définition que l'on a donnée, on trouverait :

$[0, 5]$ $\overline{\text{modulo}}$ $[7, 10] = [0, 10]$.

Ceci contredit donc 4.1.12.2.c, où l'on a posé que :

$[0, 5]$ $\overline{\text{modulo}}$ $[7, 10]$ est égal à :

$$\left[\begin{array}{ll} \underline{\text{MIN}}\ \{x \ \underline{\text{modulo}}\ y\}, & \underline{\text{MAX}}\ \{x \ \underline{\text{modulo}}\ y\} \\ x \in \gamma([0, 5]) & x \in \gamma([0, 5]) \\ y \in \gamma([7, 10]) & y \in \gamma([7, 10]) \end{array}\right]$$

On contredit donc 4.1.12.1.(1) et finalement l'hypothèse H300.3. Pour un noeud affectation n, de la forme

$$v := x \ \underline{\text{modulo}}\ y$$

interprété dans un contexte $C = \{(x, [0, 5]), (y, [7, 10])\}$, on aura :

$$\mathcal{J}_a(n, C)(v) = [0, 10] \overline{\geq} @(\{(\gamma \ \underline{\text{modulo}}\ \mu) \mid (\nu \in \gamma(C(x)) \wedge (\mu \in \gamma(C(y))))\}) = [0,5]$$

Intuitivement, ce choix correspond à une perte d'informations sur le domaine de valeurs de v, et l'on pourrait montrer que l'algorithme d'interprétation abstraite serait correct en posant :

$$\mathcal{J}_a(n, C)(v) = \varepsilon(f, (v_1, \ldots, v_m), C)$$

la fonction ε d'interprétation abstraite de l'expression f dans le contexte C permettant une "perte d'informations", c'est-à-dire que :

$$\varepsilon(f, (v_1, \ldots, v_m), C) \overline{\geq} @[\{f(\nu_1, \ldots, \nu_m) \mid (\nu_1, \ldots, \nu_m) \in \gamma(C(v_1)) \times \ldots \times \gamma(C(v_m))\}]$$

Il est toutefois nécessaire que $\mathcal{J}_a$ soit croissante sur $\mathcal{C}$ pour $\overline{\leq}$, ce que l'on obtient par une hypothèse du genre :

$$\forall (C_1, C_2) \in \mathcal{C}^2,\ C_1 \overline{\leq} C_2 \Rightarrow \varepsilon(f,(v_1,\ldots,v_n),C_1) \overline{\leq} \varepsilon(f,(v_1,\ldots,v_n,C_2)$$

Nous avons évité d'introduire la fonction ε, pour simplifier l'exposé.

4.1.12.6) Remarque :

Il est bien connu des mathématiciens [8] que les règles de calcul pour l'arithmétique sur les entiers, ne sont pas applicables en arithmétique sur les intervalles ; par exemple :

$$([1, 2] \bar{-} [-3, 1]) \bar{*} [1, 5]$$
$$= [0, 5] \bar{*} [1, 5] = [0, 15]$$

tandis que :

$$([1, 2] \bar{*} [1, 5]) \bar{-} ([-3, 1] \bar{*} [1, 5])$$
$$= [1, 10] \bar{-} [-15, 5] = [-4, 25]$$

On a le résultat connu que

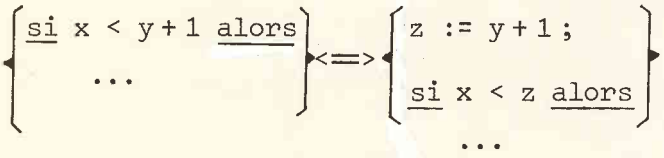
$$(x \bar{-} y) \bar{*} z \subseteq (x \bar{*} z) \bar{-} (y \bar{*} z).$$

Il est donc essentiel que les expressions arithmétiques soient évaluées dans l'interprétation abstraite élémentaire dans l'ordre utilisé dans le code généré par le compilateur du langage étudié. En particulier, si ce compilateur a une phase d'optimisation, on doit utiliser le graphe du programme optimisé comme entrée de l'algorithme d'interprétation abstraite.

4.1.13) - Prédicats sur les intervalles :

L'interprétation abstraite élémentaire des noeuds tests est définie par l'hypothèse (H307).

Nous allons nous restreindre au cas où les prédicats sont construits à partir de relations élémentaires ($<$, $\leq$, $=$, $\neq$, $\geq$, $>$, in) entre les variables et constantes composées par les opérateurs ($\wedge$, $\vee$, $\neg$). On peut toujours se ramener à ce cas en introduisant des variables supplémentaires :



Le traitement des prédicats, (dans la forme générale définie en PASCAL), qui fait intervenir des déductions du genre {$x \in [2, 10]$ et $y \in [1, 5]$ et $1 \leq x * y \leq 4$} ==> {$x \in [2, 4]$ et $y \in [1, 2]$} sera fourni ultérieurement.

Dans notre cas, soient des variables a et b définies dans des intervalles [i, s] et [j, t]. Le fait qu'une relation entre ces variables soit vraie ou fausse, implique que les valeurs de ces variables soient comprises dans des sous-ensembles de leurs intervalles de définition. Par exemple :

$\{a \in [2, 5] \wedge b \in [-5, 10] \wedge a \leqslant b\}$

$\Rightarrow \{a \in [2, 5] \wedge b \in [2, 10]\}$

De même, $\{a \in [2, 5] \wedge a \in [3, 7]\} \Rightarrow \{a \in [3, 5]\}$.

De manière plus générale, on peut écrire :

$\{a \in [i, s] \wedge b \in [j, t] \wedge a \leqslant b\} \Rightarrow$

cas
| $i > t \Rightarrow (a \in \gamma(\square) \wedge b \in \gamma(\square))$;
| autres $\Rightarrow (a \in [i, \min(s, t)] \wedge b \in [\max(i, j), t])$;
fincas ;

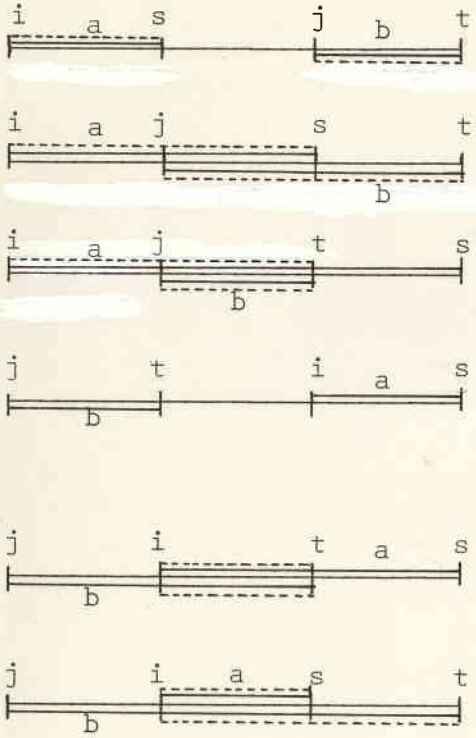
De même,

$\{a \in [i, s] \wedge a \in [j, t]\} \Rightarrow$

cas
| $(j > s) \vee (i > t) \Rightarrow (a \in \gamma(\square), b \in \gamma(\square))$;
| autres $\Rightarrow (a \in [\max(i, j), \min(s, t)])$;
fincas ;

Les démonstrations se font par analyse de cas, comme, par exemple :

$(a \in [i,s] \wedge b \in [j,t])$



a ——— Domaine de définition de la variable a

---------- Domaine de validité de la relation

$a \leq b$

$a \in [i,s],\ b \in [j,t]$

$a \in [i,s],\ b \in [j,t]$

$a \in [i,t],\ b \in [j,t]$

- cas impossible :
  $a \in \gamma(\square),\ b \in \gamma(\square)$

$a \in [i,t],\ b \in [i,t]$

$a \in [i,s],\ b \in [i,t]$

cas $i > t \Rightarrow (a \in \gamma(\square),\ b \in \gamma(\square))$
  autres $\Rightarrow (a \in [i, \min(s,t)]$
    $\wedge\ b \in [\max(j,i), t]$
fincas ;

$a < b$

$a \in [i,s],\ b \in [j,t]$

$a \in [i,s],\ b \in [j,t]$

$a \in [i,t-1],\ b \in [j,t]$

- cas impossible :
  $a \in \gamma(\square),\ b \in \gamma(\square)$

$a \in [i,t-1],\ b \in [i+1,t]$

$a \in [i,s],\ b \in [i+1,t]$

cas $i \geq t \Rightarrow (a \in \gamma(\square),\ b \in \gamma(\square))$
  autres $\Rightarrow (a \in [i, \min(s, t-1)]$
    $\wedge\ b \in [\max(i+1, j), t])$
fincas ;

Comme dans le cas des expressions arithmétiques, on peut associer à tout prédicat du langage, un prédicat sur les intervalles, qui une fois évalué , donne le résultat de l'interprétation élémentaire d'un noeud test. Par exemple, l'évaluation d'un noeud test de la forme $(a \leq b) \wedge (b \leq a)$, dans un contexte $C = \{(a, [i, s]), (b, [j, t])\}$, conduit sur la branche vraie à :

cas
- $i \geq t \Rightarrow \{(a, \square), (b, \square)\}$
- $j > s \Rightarrow \{(a, \square), (b, \square)\}$
- autres $\Rightarrow$

$\{(a, [i, s]) \leq (b, [j, t])\} \wedge \{(b, [j, t]) \leq (a, [i, s])\}$

$\Rightarrow \{(a, [i, \underline{\min}(s, t)]), (b, [\underline{\max}(j, i), t])\} \wedge \{(b, [j, \underline{\min}(s,t)]) \wedge (a, [\underline{\max}(i,j), s])\}$

$\Rightarrow \{\{(a, [i, \underline{\min}(s,t)]) \wedge (a, [\underline{\max}(i,j), s])\}, \{(b, [\underline{\max}(j,i), t]) \wedge (b, [j, \underline{\min}(s,t)])\}\}$

$\Rightarrow \{(a, [\underline{\max}(i, \underline{\max}(i,j)), \underline{\min}(\underline{\min}(s,t), s)]), (b, [\underline{\max}(\underline{\max}(i,j), j), \underline{\min}(t, \underline{\min}(s,t))])\}$

$\Rightarrow \{(a, [\underline{\max}(i,j), \underline{\min}(s,t)]), (b, [\underline{\max}(i,j), \underline{\min}(s,t)])\}$

fincas

Ce qui est évident quand on songe que $\{(a \leq b) \wedge (b \leq a)\} \Leftrightarrow \{a = b\}$.

La négation est plus difficile à traiter. En effet :

$\{a \in [i, s] \wedge b \in [j, t] \wedge aRb \Rightarrow a \in [i', s'] \wedge b \in [j', t']\}$
n'exclut pas que :

$\{\exists a \in [i', s'] \wedge \exists b \in [j', t'] \mid \neg\, aRb\}$

donc le fait que a doit appartenir à [i', s'] et b à [j', t'] pour que la relation aRb puisse être vraie, n'implique pas que a ne doit pas appartenir à [i', s'] et que b ne doit pas appartenir à [j', t'] pour que la négation de la relation aRb soit vraie.

Pour résoudre le problème, on est conduit à partionner le domaine de a, c'est-à-dire [i, s] et celui de b, [j, t] en trois domaines $[i_v, s_v]$, $[i_F, s_F]$, $[i_I, s_I]$ et $[j_v, t_v]$, $[j_F, t_F]$, $[j_I, t_I]$ respectivement, tels que si $a \in [i_v, s_v]$ alors la relation aRb est vraie pour toute valeur de b, si $a \in [i_F, s_F]$ alors $\neg(aRb)$ est vraie pour toute valeur de b, et enfin si $a \in [i_I, s_I]$ alors la relation aRb est incertaine, c'est-à-dire qu'il y a des valeurs de b pour lesquelles aRb est vraie et d'autres pour lesquelles aRb est fausse(même chose pour b).

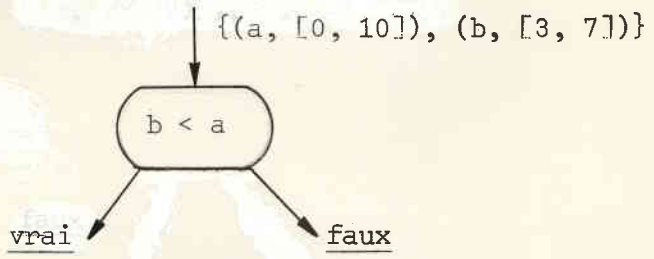
Exemple :

Si $a \in [0, 10]$ et $b \in [3, 7]$, alors :

- $a \in [8, 10] \Rightarrow b < a$
- $a \in [0, 3\ ] \Rightarrow \neg(b < a)$
- $a \in [4, 7\ ] \Rightarrow (b < a) \vee \neg(b < a)$

Pour appliquer ce résultat à l'interprétation élémentaire d'un noeud test, le domaine de a sur la branche vrai est $[i_V, s_V] \bar{\cup} [i_I, s_I]$ et sur la branche faux, c'est $[i_F, s_F] \bar{\cup} [i_I, s_I]$, la remarque 4.1.12.5. étant également applicable dans ce cas.

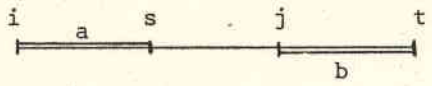
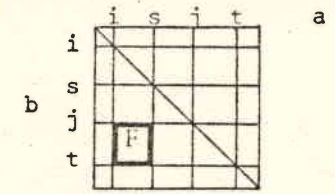
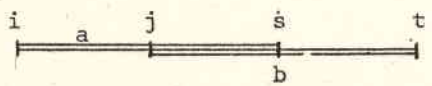
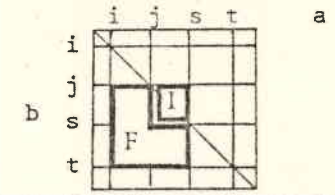
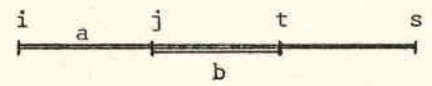
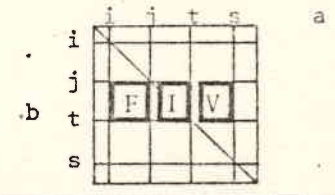
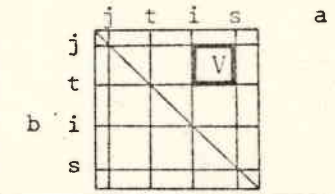
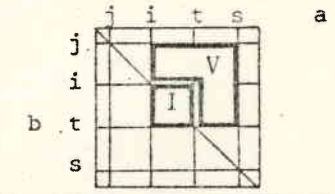
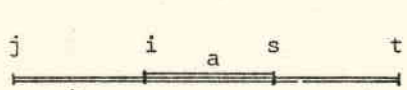
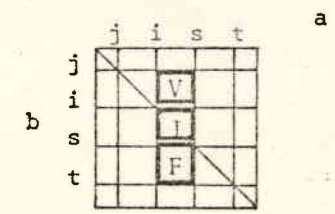
Sur notre exemple, on obtient :



$\{(a, [8, 10] \bar{\cup} [4, 7]), (b, [3, 7])\} = \{(a, [4, 10]), (b, [3, 7])\}$

$\{(a, [0, 3] \bar{\cup} [4, 7]), (b, [3, 7])\} = \{(a, [0, 7]), (b, [3, 7])\}$

Pour traiter la relation d'inégalité stricte, on obtient (les cas de dégénérescence sont omis) :

$a \in [i,s]$ $b \in [j,t]$	$b < a$ est vrai	$b < a$ est faux	$b < a$ est incertain	Partition
	$b \in \gamma(\square)$, $a \in \gamma(\square)$	$b \in [j,t]$, $a \in [i,s]$	$b \in \gamma(\square)$, $a \in \gamma(\square)$	
	$b \in \gamma(\square)$, $a \in \gamma(\square)$	$b \in [s,t]$ $a \in [i,j]$	$a \in [j+1,s]$ $b \in [j,s-1]$	
	$a \in [t+1,s]$	$a \in [i,j]$	$a \in [j+1,t]$ $b \in [j,t]$	
	$b \in [j,t]$, $a \in [i,s]$	$b \in \gamma(\square)$, $a \in \gamma(\square)$	$b \in \gamma(\square)$, $a \in \gamma(\square)$	
	$b \in [j,i-1]$ $a \in [t+1,s]$	$b \in \gamma(\square)$, $a \in \gamma(\square)$	$a \in [i,t]$ $b \in [i,t]$	
	$b \in [j,i-1]$	$b \in [s,t]$	$a \in [i,s]$ $b \in [i,s-1]$	

On remarque que le domaine d'incertitude est le domaine initial ([i, s] pour a) moins les domaines où la relation est toujours vraie ou toujours fausse.
Autrement dit, $[i_I, s_I]$ et $[j_I, t_I]$ sont inutiles puisqu'on peut les reconstituer à partir de [i, s], $[i_V, s_V]$, $[i_F, s_F]$ et [j, t], $[j_V, t_V]$, $[j_F, t_F]$ respectivement.

A toute relation R ($\leq$, $<$, $\neq$, $=$, $>$, $\geq$, in) du langage, on fait correspondre un opérateur $\underline{\underline{R}}$ défini par :

$$\underline{\underline{R}}\ (a, b, C) \Longrightarrow (C_V, C_F)\ ;$$

qui à deux identificateurs a et b et un contexte C fait correspondre deux contextes C_V et C_F dans lesquels la relation aRb est toujours vraie, ou fausse respectivement.

A l'opérateur de négation, $\neg$, on fait correspondre un opérateur $\underline{\underline{\neg}}$, défini par :

$$\underline{\underline{\neg}}\ (C_V, C_F) = (C_F, C_V)$$

A l'opérateur $\wedge$ de conjonction, on fait correspondre un opérateur $\underline{\underline{\wedge}}$, défini par :

$$(C_V, C_F)\ \underline{\underline{\wedge}}\ (C'_V, C'_F)$$
$$= (C_V\ \underline{\underline{\cap}}\ C'_V,\ C_F\ \underline{\underline{\cup}}\ C'_F)$$

Avec :

$$C_1\ \underline{\underline{\cap}}\ C_2 = \{(i, v)\ |\ (i \in I),\ (v \in V_a - \{\square\}) \wedge (v = C_1(i)\ \underline{\cap}\ C_2(i))\}$$

et

$$v_1\ \underline{\cap}\ v_2 = \underline{\text{cas}}\ \ v_1 \in V_a,\ v_2 \in V_a\ \underline{\text{dans}}$$
$$\quad \vdash (?, \square) \Longrightarrow v_1\ ;$$
$$\quad \vdash (\square, ?) \Longrightarrow v_2\ ;$$
$$\quad \vdash [i, s], [j, t] \Longrightarrow$$
$$\qquad \underline{\text{cas}}$$
$$\qquad \vdash (j > s),\ (i > t) \Longrightarrow \square\ ;$$
$$\qquad \vdash \underline{\text{autres}} \Longrightarrow [\underline{\max}(i, j),\ \underline{\min}(s, t)]\ ;$$
$$\qquad \underline{\text{fincas}}\ ;$$
$$\underline{\text{fincas}}\ ;$$

De même :

$$C_1 \bar{\underline{\cup}} C_2 = \{(i, v) \mid (i \in I), (v \in V_a - \{\square\}), (v = C_1(i) \bar{\underline{\cup}} C_2(i))\}$$

avec :

$$v_1 \bar{\underline{\cup}} v_2 = \underline{\text{cas}}\ \ v_1 \in V_a, v_2 \in V_a\ \underline{\text{dans}}$$

```
        | (?, □) ==> v1 ;
        | (□, ?) ==> v2 ;
        | [i, s], [j, t] ==>
                cas
                | (j > s), (i, t) ==> □;
                | autres ==>[min(i, j), max(s, t)] ;
                fincas ;
        fincas ;
```

Enfin, on fait correspondre à la disjonction v, l'opérateur $\bar{\bar{v}}$ défini par :

$$(C_V, C_F) \bar{\bar{v}} (C'_V, C'_F) = (C_V \bar{\bar{\underline{\cup}}} C'_V, C_F \bar{\bar{\underline{\cap}}} C'_F)$$

Exemple :

$\{a \in [-\infty, 10]$ et $(1 < a) \wedge (a < 12)\} \Longrightarrow$

$\bar{\bar{\leq}}$ (1, a, {(1, [1, 1]), (a, [$-\infty$, 10])}) $\bar{\bar{\underline{\wedge}}}$ $\bar{\bar{\leq}}$ (a, 12,{(12, [12, 12]), (a,[$-\infty$,10])})

$\Longrightarrow$ ({(a, [2, 10])}, {(a, [$-\infty$, 1])}) $\bar{\bar{\underline{\wedge}}}$ ({(a, [$-\infty$, 11])}, Φ)

$\Longrightarrow$ ({(a, [2,10] $\bar{\bar{\underline{\cap}}}$ [$-\infty$,11])},{(a,[$-\infty$,1] $\bar{\bar{\underline{\cup}}}$ □)})

$\Longrightarrow$ ({(a,[2,10])},{(a,[$-\infty$,1])}

Finalement, le résultat de l'interprétation abstraite élémentaire d'un noeud test n de la forme $Q(v_1,\ldots,v_m)$ dans un contexte C s'obtient par l'évaluation abstraite de $\bar{\bar{Q}}(v_1,\ldots,v_m)$, C comme nous l'avons décrit précédemment. Ceci produit deux contextes (C_v, C_F). Sur la branche de sortie vrai, on a

$$\underline{\mathcal{J}}_t(n,C)\ [1]\ (v_k) = C(v_k) \bar{\underline{=}} C_F(v_k)$$

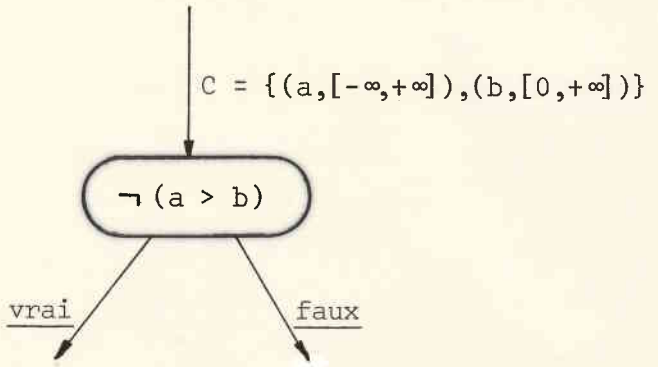
$$\underline{\mathcal{J}}_t(n,C)\ [2]\ (v_k) = C(v_k) \bar{\underline{=}} C_v(v_k)$$

où l'opérateur $\bar{\underline{=}}$ est défini comme suit :

```
v1 ≡ v2 = cas v1 ∈ Va, v2 ∈ Va dans
          | (□,?) ⟹ □ ;
          | (≠□,□) ⟹ v1 ;
          | ([i,s],[j,t]) ⟹
            cas
            | (j ≤ t < i ≤ s) ⟹ [i,s] ;
            | (j ≤ i ≤ t < s) ⟹ [t+1,s] ;
            | (i < j ≤ t < s) ⟹ [i,s] ;
            | (i < j ≤ s ≤ t) ⟹ [i,j-1] ;
            | (i ≤ s < j < t) ⟹ [i,s] ;
            | (j ≤ i ≤ s ≤ t) ⟹ □
            fincas ;
          fincas ;
```

Certains cas de dégénérescence où les intervalles sont réduits à un point sont omis.

Exemple :



on évalue

$\underline{\underline{\neg}}\,(\underline{\underline{\geq}}, a, b, c) =$

$\underline{\underline{\neg}}\,(\Phi, \{(a,[-\infty,0])\})$

$= \quad (\{(a, [-\infty,0])\}$

Sur la branche vrai :

- la valeur de a est :
  $[-\infty,+\infty] \underline{\underline{\cap}} \square = [-\infty,+\infty]$;
- la valeur de b est
  $[0,+\infty] \underline{\underline{\cap}} \square = [0,+\infty]$;

Sur la branche faux :

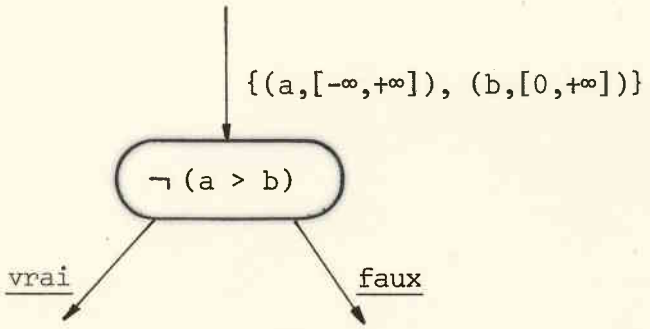
- la valeur de a est

$$[-\infty,+\infty] \stackrel{=}{\underline{}} [-\infty,0] = [1,+\infty]$$

- la valeur de b est

$$[0,+\infty] \stackrel{=}{\underline{}} \square = [0,+\infty] \ ;$$

Soit



{(a,[-∞,+∞]), (b,[0,+∞])} {(a,[1,+∞]), (b,[0,+∞]}

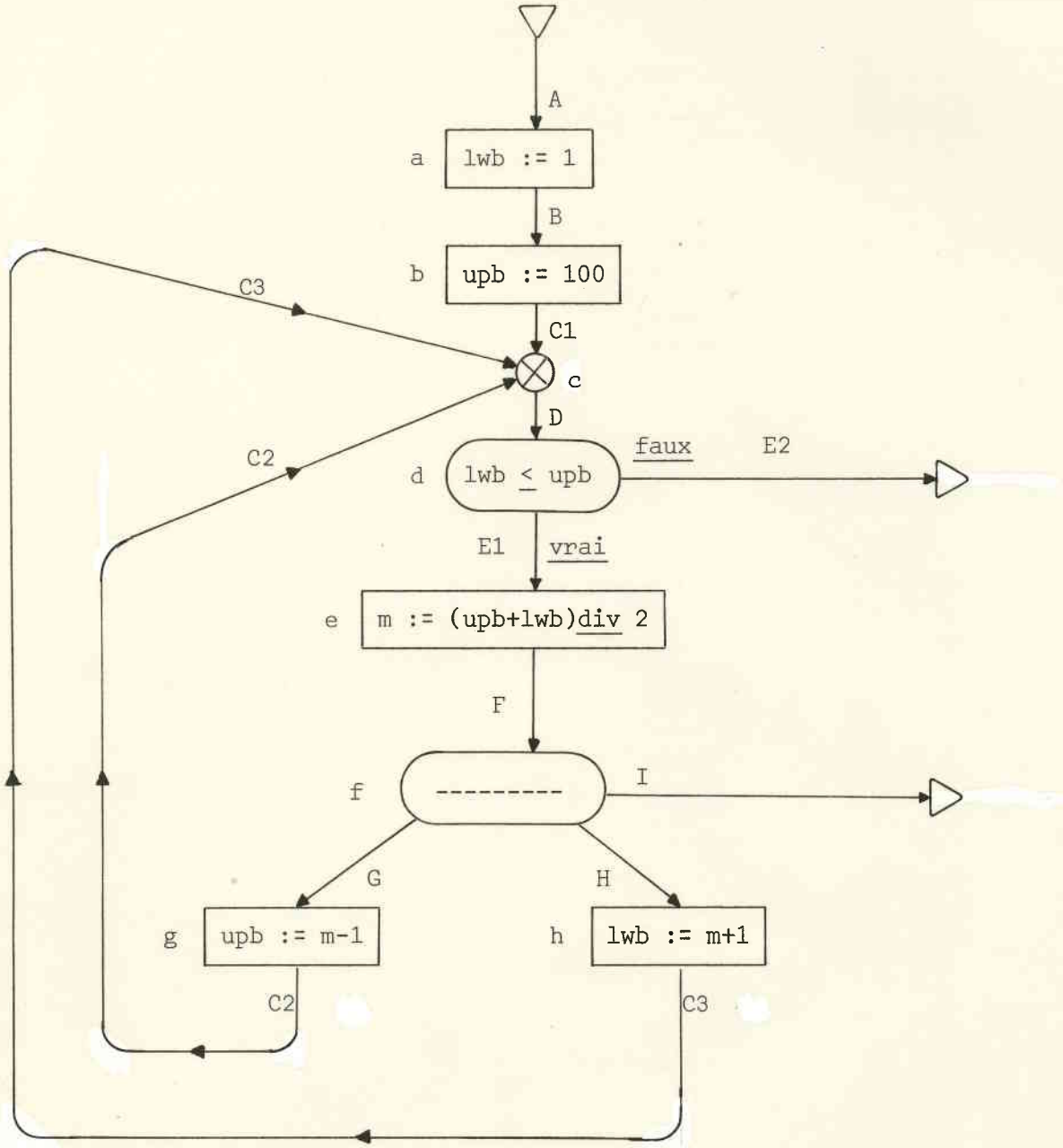
Au-delà de cette courte présentation intuitive, il faudrait présenter formellement cette interprétation élémentaire des noeuds tests, et qu'elle satisfasse l'hypothèse H307. Ceci sera fait ultérieurement, et sans restrictions sur les prédicats.

4.1.14 - Exemple d'application 1 : vérification statique de l'inclusion des valeurs des indices d'un tableau dans les bornes de ce tableau -

Considérons le programme suivant (en PASCAL), de recherche de l'indice m d'une valeur entière R dans un tableau d'entiers T rangé en ordre croissant, par dichotomie :

```
[1]  var T : array [1 .. 100] of integer ;
[2]      R, upb, lwb, m : integer ;
[3]  begin
[4]  |  {initialisation de T en R ---}
[5]  |  lwb := 1 ; upb := 100 ;
[6]  |  if (R < T(lwb)) v (R > T(upb)) then
[7]  |  |  go to 1 ;
[8]  |  while lwb ≤ upb do
[9]  |     begin
[10] |     |  m := (upb+lwb) div 2 ;
[11] |     |  if R < T(m) then
[12] |     |  |    upb := m-1
[13] |     |  else
[14] |     |  |  if R = T(m) then
[15] |     |  |  |  begin write (m) ; go to 1 end
[16] |     |  |  else
[17] |     |  |  |  lwb := m+1
[18] |     end
[19] 1 : end.
```

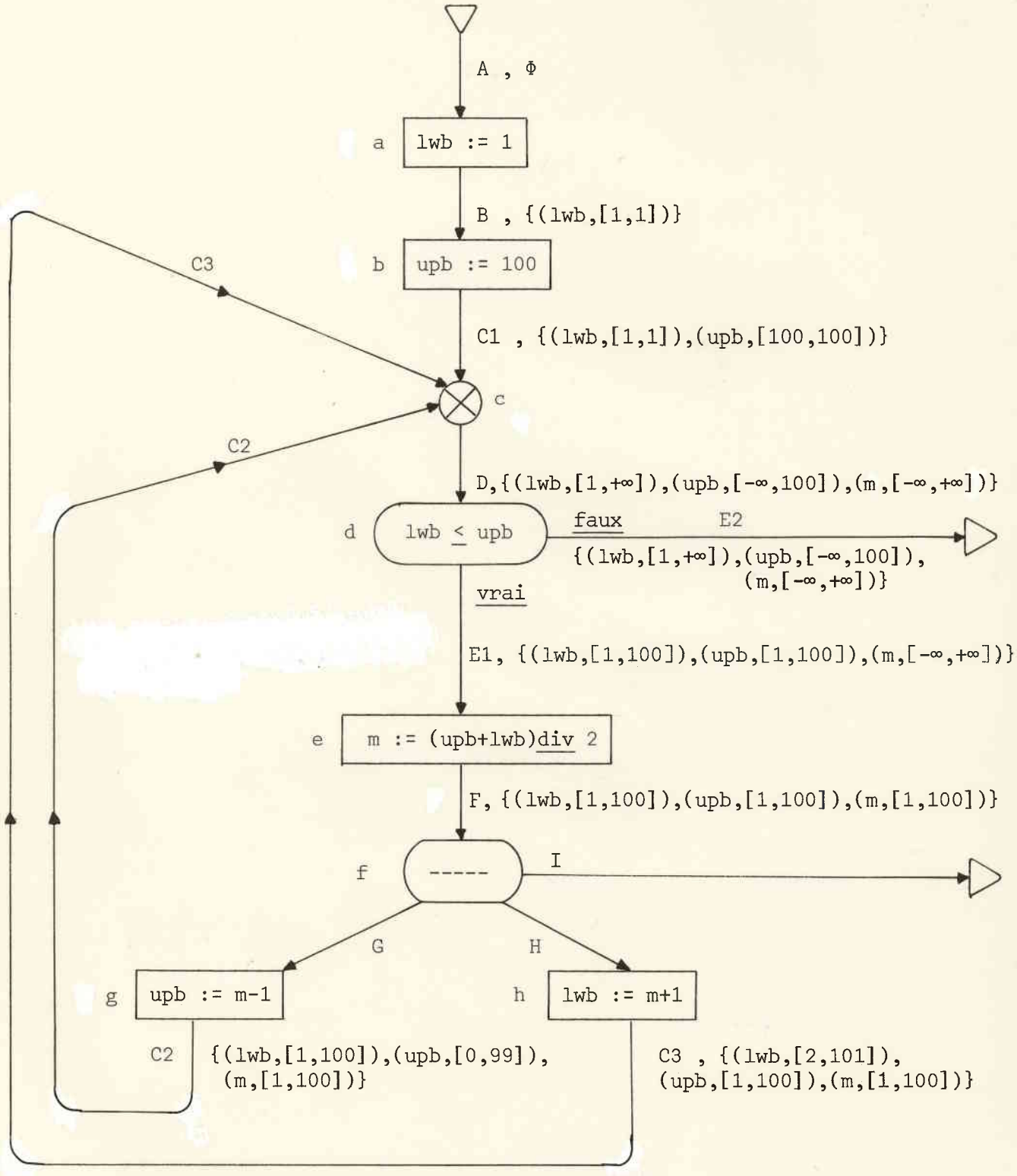
Un complicateur générant des tests dynamiques de programmes inclurait dans le code généré des tests pour vérifier que lwb, upb et m sont compris entre 1 et 100 aux lignes [6], [11], [12]. On peut toutefois montrer statiquement que ces tests sont inutiles. Pour faciliter la présentation, on peut ignorer R et T, ce qui donne le graphe de programme suivant :



La trace de l'algorithme est la suivante :

Pas	Arc	Contexte local (arc)	Noeud traité	Contexte de l'arc de sortie du noeud traité	Chemins à exécuter	Jonction
1					A	Ø
2	(A)	Φ	a affect.	lwb =[1,1]	B	Ø
3	(B)	lwb = [1,1]	b affect.	lwb =[1,1]), upb =[100,100]	C1	Ø
4	(C1)	lwb = [1,1], upb = [100,100]	c jonct.	lwb =[1,1] , upb =[100,100]	Ø	D
5	D	lwb = [1,1], upb = [100,100]	d test	lwb =[1,1]), upb =[100,100] | Ø	E1 | E2	Ø
6	E2	Ø	sortie		E1	Ø
7	E1	lwb = [1,1], upb = [100,100]	e affect.	lwb = [1,1], upb = [100,100], m = [50,50]	F	Ø
8	F	lwb = [1,1], upb = [100,100], m = [50,50]	f test	lwb = [1,1], upb = [100,100], m = [50,50]	G I H	Ø
9	I	lwb = [1,1], upb = [100,100], m = [50,50]	sortie		G	Ø
10	G	lwb = [1,1], upb = [100,100], m = [50,50]	g affect.	lwb = [1,1], upb = [49,49], m = [50,50]	H C2	Ø
11	C2	lwb = [1,1], upb = [49,49], m = [50,50]	c jonct.		H	D
12	H	lwb = [1,1], upb = [100,100], m = [50,50]	h affect.	lwb = [51,51], upb = [100,100], m = [50,50]	C3	
13	C3	lwb = [51,51], upb = [100,100],m = [50,50]	c jonct.	lwb = [1,51], upb = [49,100], m = [50,50]		D
14	D	lwb = [1,+∞], upb = [-∞,100], m = [50,50]	d test	lwb = [1,100] upb = [1,100] m = [50,50] | lwb = [1, +∞] upb = [-∞,100] m = [50,50]	E1 | E2	Ø
15	E2	lwb = [1,+∞], upb = [-∞,100], m = [50,50]	sortie		E1	
16	E1	lwb = [1,100], upb = [1,100], m = [50,50]	e affect.	lwb = [1,100], upb = [1,100], m = [1,100]	I	
17	(F)	lwb = [1,100], upb = [1,100], m = [1,100]	f test	lwb = [1,100], upb = [1,100], m = [1,100]	G I H	
18	(I)	lwb = [1,100], upb = [1,100], m = [1,100]	sortie		G	
19	(G)	lwb = [1,100], upb = [1,100], m = [1,100]	g affect.	lwb = [1,100], upb = [0,99], m = [1,100]	C2	
20	(C2)	lwb = [1,100], upb = [0,99], m = [1,100]	c fonct.		H	D
21	(H)	lwb = [1,100], upb = [1,100], m = [1,100]	h affect.	lwb = [2,101], upb = [1,100], m = [1,100]	C3	
22	(C3)	lwb = [2,101], upb = [1,100], m = [1,100]	c jonct.	lwb = [1,101], upb = [0,100], m = [1,100]		D
23	(D)	lwb = [1,+∞], upb = [-∞,100], m = [-∞,+∞]	d test	lwb = [1,100] upb = [1,100] m = [-∞,+∞] | lwb = [1,+∞] upb = [-∞,100] m = [-∞,+∞]	E1 F2	Ø
24	(E2)	lwb = [1,+∞], upb = [-∞,100], m = [-∞,+∞]	sortie		E1	
25	(E1)	lwb = [1,100], upb = [1,100], m = [-∞,+∞]	e affect.	lwb = [1,100], upb = [1,100], m = [1,100]	F	
26	F	arrêt				

Le résultat final montre que les vérifications dynamiques sont inutiles :



A la suite de cet exemple, on peut remarquer que l'on n'a pas traité le cas de R ni celui du tableau T :

- Notre programme initialise R à la ligne [4], par exemple, par une instruction de lecture. Dans l'interprétation abstraite, on affecte à R une valeur $[-\infty,+\infty]$, et cette valeur est propagée sur tous les arcs du graphe du programme.
- Dans l'interprétation abstraite, on peut considérer un tableau comme étant constitué d'un seul élément abstrait. Toute affectation à un élément quelconque du tableau est considérée comme une affectation à l'élément abstrait. Tout test sur un élément quelconque du tableau porte également sur l'élément abstrait unique. L'interprétation abstraite est donc conduite comme si le tableau était un entier. Par contre, le résultat final de l'interprétation abstraite est à interpréter différemment : si sur un arc la valeur abstraite du tableau T est [i,s], alors tout élément T[k] du tableau (où k est compris entre les bornes inférieures et supérieures de T) est compris entre i et s.

$$\{i \leq T[k] \leq s,\ \forall k \in [\underline{binf}(T), \underline{bsup}(T)]\}.$$

- Dans l'algorithme d'interprétation abstraite que nous avons présenté, on ne tient pas compte des déclarations des variables entières. En PASCAL et en LIS, par exemple, on déclarerait dans notre exemple :

```
[1]  var T : array[1 .. 100] of integer ;
[2]  R : integer ;
 -   upb : 0 .. 99 ;
 -   lwb : 1 .. 101 ;
 -   m : 1 .. 100 ;
```

On peut tenir compte de cette information, quand on élargit les contextes abstraits aux noeuds de jonction de boucles ; pour une variable v, déclarée entre les bornes v_i et v_s, alors on définira l'élargissement de la valeur abstraite x de v, par la valeur abstraite y de v comme suit :

```
(x ⊽ y) = cas x ∈ V_a, y ∈ V_a alors
        | □, ? => y ;
        | ?, □ => x ;
        | [n1,m1],[n2,m2] =>
            [si n2 < n1 alors
                si v_i ≤ n2 alors v_i sinon -∞ fsi
             sinon n1 fsi ;
            si m2 > m1 alors
              si m2 ≤ v_s alors v_s sinon +∞ fsi
            sinon m1 fsi ; ]
        fincas ;
```

Dans la trace de notre algorithme, on aurait

{(lwb,[1,1],(upb,[100,100])
  $\bar{\nabla}$ {(lwb,[1,51]),(upb,[49,100]),(m,[50,50])}
= {(lwb,[1,101]),(upb,[0,100]),(m,[50,50])}
au lieu de
  {(lwb,[1,+∞]),(upb,[-∞,100]),(m,[50,50])

Les déclarations permettent donc des résultats plus précis, mais ne sont pas indispensables. En tout cas, on a une vérification statique des types, et le compilateur n'a pas à générer de testspour vérifier les domaines de définition de m, lwb et upb aux lignes [5], [10], [12] et [17].

- Enfin, dans certains langages, comme LIS, les intervalles de définition des entiers sont symboliques.
On aura :

```
lire N ;
A, B, C : 1 .. N ; E, F : -N .. +N ;
```

En ALGOL 60, les tableaux dynamiques ont des bornes formelles. Dans ce cas, encore, le principe de notre algorithme est valable, mais son application plus délicate, à cause des manipulations formelles de formules. On rejoint les difficultés des techniques d'évaluation symbolique des programmes.

4.2 - Etude des valeurs des pointeurs d'un programme -

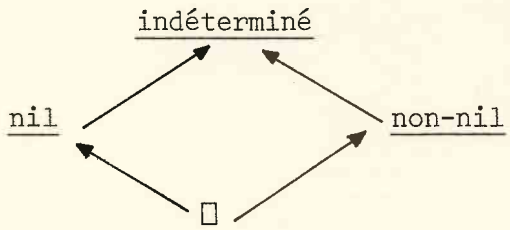
Soit à déterminer pour chaque variable de type pointeur ou référence d'un programme, si elle peut ou non prendre la valeur nil en un point quelconque du programme ;
on peut définir une interprétation abstraite sur les valeurs (□, nil, non-nil, indéterminé). Les valeurs concrètes sont donc des pointeurs éventuellement nuls, et la fonction d'abstraction d'ensembles de valeurs concrètes est :

$\forall X \subset V_c$,
  @(X) = cas
  ├ X = ∅ → □ ;
  ├ X = {nil} → nil ;
  ├ {∀x ∈ X, x ≠ nil} → non-nil ;
  ├ autres → indéterminé ;
  fincas ;

L'union des valeurs abstraites $\bar{U}$ est définie par :

x $\bar{U}$ y = cas x ∈ V_c, y ∈ V_c dans
  ├ □, □ → □ ;
  ├ □, ? → y ;
  ├ ?, □ → x ;
  ⋮ ...
  ├ nil, nil → nil ;
  ├ non-nil, non-nil → non-nil ;
  ├ autres → indéterminé ;
  fincas ;

Pour la comparaison de valeurs abstraites, on a le diagramme suivant :



Le nombre de valeurs abstraites étant fini, on peut définir l'élargissement de valeurs abstraites par :

$x \bar{\nabla} y = x \bar{U} y$

L'interprétation des affectations est triviale.

p := nil ;
affecte la valeur abstraite nil à p ;

p := x ;
affecte la valeur abstraite de x à p.

Il faut aussi en PASCAL,considérer des affectations à des variables de type pointeur sous la forme

alloc(p) ou alloc(p,t)

ce qui alloue à p la valeur indéterminé puisque le résultat de l'allocation peut être la valeur nil quand la classe d'allocation est pleine, ou une valeur différente de nil quand l'allocation est effective.

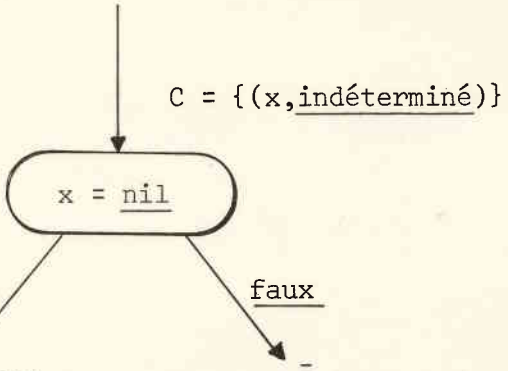
L'interprétation des tests est également simple puisqu'on ne peut que comparer des variables de type pointeur :

```
(x = y) = cas x, y dans
        | nil, nil → {(x,nil),(y,nil)}, Φ ;
        | nil, non-nil → Φ, {(x,nil),(y,non-nil)};
        | nil, indéterminé → {(x,nil),(y,nil)},{(x,nil),(y,non-nil)} ;
        | non-nil, nil → Φ, {(x,non-nil),(y,nil)} ;
        | non-nil, indéterminé → Φ, {(x,non-nil),(y,nil)}
        | indéterminé, non-nil → Φ, {(x,nil),(y,non-nil)} ;
        | indéterminé, nil → {(x,nil),(y,nil)},{(x,non-nil),(y,nil)} ;
        | autres → Φ, Φ;
        fincas ;
```

Comme en 4.1.13, on a, pour chaque cas, réparti le contexte d'entrée, en deux contextes de sortie, l'un où le test est toujours vrai, et l'autre où le test est toujours faux.

Pour les autres opérations ($\neg$, $\vee$, $\wedge$), le traitement des prédicats est le même qu'en 4.1.13. En particulier, l'opérateur $\neq$ est défini par
$\{x \neq y\} <=> \{\neg(x=y)\}$.

L'interprétation élémentaire d'un test comme



$C' = C \doteq \{(x,\text{non nil})\} = \{(x,\text{indéterminé} \doteq \text{non-nil})\} = \{(x,\text{nil})\}$

$C'' = C \doteq \{(x,\text{nil})\} = \{(x,\text{indéterminé} \doteq \text{nil})\} = \{(x,\text{non-nil})\}$

utilise la différence de valeurs abstraites définie comme suit :

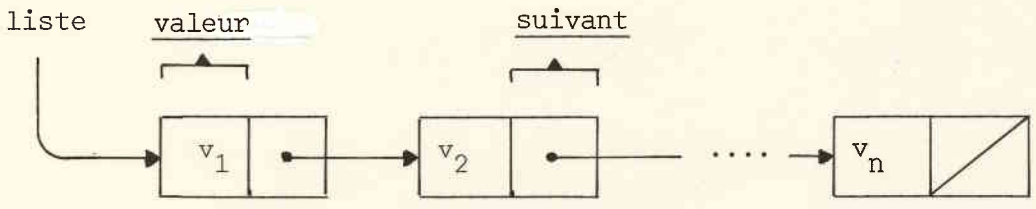
$v_1 \doteq v_2$ = cas $v_1 \in V_a$, $v_2 \in V_a$ alors

- indéterminé, nil → non-nil
- indéterminé, non-nil → nil
- indéterminé, indéterminé → □
- indéterminé, □ → indéterminé
- non-nil, nil → □
- etc ...

fincas ;

(On a $\gamma(v_1 \doteq v_2) = C_{\gamma(v_1)}[\gamma(v_1) \cap \gamma(v_2)]$.

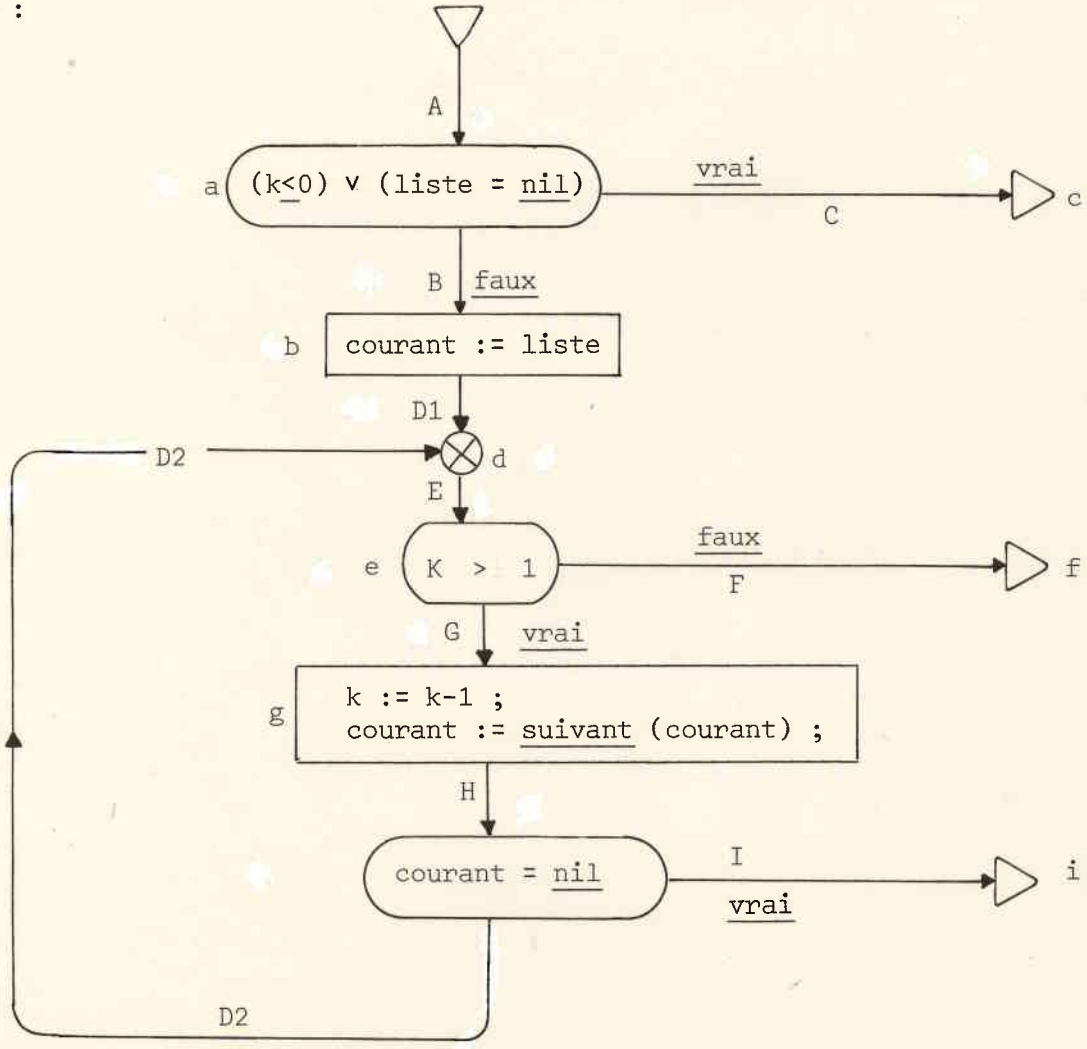
Considérons maintenant un exemple d'application, il s'agit de trouver la valeur du $K^{ème}$ enregistrement d'une liste linéaire.



```
[1]  si (k ≤ 0) ∨ (liste = nil) alors
[2]  |     erreur.
[3]  finsi ;
[4]  courant := liste ;
[5]  tantque  k > 1 faire
[6]  |       k := k-1 ;
[7]  |       courant := suivant (courant) ;
[8]  |       si courant = nil alors
[9]  |       |  erreur.
[10] |       finsi ;
[11] refaire ;
[12] v := valeur (courant) ;
```

Lors de l'interprétation abstraite, la fonction suivant, à résultat de type référence, produit une valeur abstraite "indéterminé", la fonction valeur, un entier entre [-∞,+∞] par exemple.

Pour faire la trace de l'interprétation abstraite, on utilise l'organigramme suivant :



La trace se fait ici avec un contexte initial,

{(K, [-∞,+∞]), (liste,indéterminé)}

puisque K est un entier, et liste une référence éventuellement nulle à un enregistrement de la liste linéaire.

Pas	Arc	Contexte local (arc)	Noeud traité	Contexte de l'arc de sortie du noeud traité	Chemin à exécuter	Jonction
1			noeud d'entrée	{(k, [-∞,+∞]), (liste, indéterminé)}	A	Ø
2	(A)	{(k, [-∞,+∞]), (liste, indéterminé)}	a test	{(k, [-∞,0]),(liste, nil)} {(k, [1,+∞]), (liste, non-nil)}	C B	Ø
3	(C)	{(k, [-∞,0]), (liste, nil)}	c sortie		B	Ø
4	(B)	{(k, [1,+∞]), (liste, non-nil)}	b affect.	{(k, [1,+∞]), (liste, non-nil), (courant, non-nil)}	D1	Ø
5	D1	{(k, [1,+∞]), ..., (courant, non-nil)}	d jonct.		Ø	{d}
6	D1 D2	{(k, [1,+∞]), ..., (courant, non-nil)} Φ	d jonct.	{(k, [1,+∞]), (courant, non-nil)}	E	Ø
7	(E)	{(k, [1,+∞]), (courant, non-nil)}	e test	{(k, [1,1]), (courant, non-nil)} {(k, [2,+∞]), (courant, non-nil)}	F G	Ø
8	(F)	{(k, [1,1]), (courant, non-nil)}	f sortie		G	Ø
9	(G)	{(k, [2,+∞]), (courant, non-nil)}	g affect.	{(k, [1,+∞]), (courant, indéterminé)}	H	Ø
10	(H)	{(k, [1,+∞]), (courant, indéterminé)}	h test	{(k, [1,+∞]), (courant, nil)} {(k, [1,+∞]), (courant, non-nil)}	I D2	Ø Ø
11	(I)	{(k, [1,+∞]), (courant, nil)}	i sortie		D2	Ø
12	D2	{(k, [1,+∞]), (courant, non-nil)}	d jonct.		Ø	{d}
13	(D1) (D2)	{(k, [1,+∞]), (courant, non-nil)} {(k, [1,+∞]), (courant, non-nil)}	d jonct.	arrêt	Ø	Ø

Finalement, l'interprétation abstraite nous apprend que le pointeur "courant" n'est pas nil en [7] et [12], donc les accès aux éléments de la liste sont corrects.

Remarques :

1 - Le coût de vérification dynamique de la correction des pointeurs, lors d'un accès à un enregistrement est habituellement peu cher, puisqu'on utilise le mécanisme de protection mémoire pour les programmes utilisateurs (en mode esclave). Par contre, pour les langages d'implémentation de système, cet artifice n'est plus valable puisque certains programmes en mode maître ont accès à toute la mémoire. Il est donc indispensable de faire la vérification statiquement pour éviter de truffer les programmes de tests de non-nullité inutiles.

2 - Comme dans le cas des tableaux, l'interprétation abstraite d'objets de type enregistrement, se fait comme s'il n'y avait qu'un seul enregistrement par classe, sur lequel pointent toutes les références à cette classe.

3 - Dans les deux exemples que l'on a donnés, l'interprétation abstraite vient en complément de la théorie des types classiques, puisqu'il s'agit de déterminer le sous-type d'une variable, compte tenu du flot de contrôle du programme. Ceci permet d'imaginer une théorie des types, qui soit plus fine que celles actuellement connues. Par exemple, on peut imaginer d'introduire des types comme pair, impair, ... et de faire statiquement le contrôle des types.

4.3 - Parité des variables entières d'un programme -

Soit à déterminer la parité des variables d'un programme. On peut définir une interprétation abstraite sur les valeurs ($\square$, pair, impair, entier). Avec $\forall X \subset V_c$

$@(X) =$ cas
- $X = \emptyset \rightarrow \square$;
- $\{\forall x \in X,$ pair$(x)\} \rightarrow$ pair
- $\{\forall x \in X,$ impair$(x)\} \rightarrow$ impair ;
- autres $\rightarrow$ entier ;

fincas ;

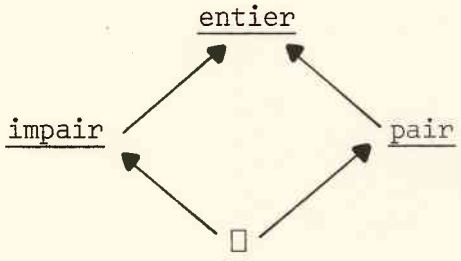
$\forall (X,Y) \in V_a^2$

$X \bar{\cup} Y$ = cas X, Y alors

| □, □ → □ ;
| □, ? → Y ;
| ?, □ → X ;
| pair, pair → pair ;
| impair, impair → impair ;
| autres → entier ;

fincas ;

Pour la comparaison de valeurs abstraites, on a le diagramme de HASSE suivant :



L'élargissement de valeurs abstraites étant défini par $x \bar{\nabla} y = x \bar{\cup} y$, la convergence de l'algorithme est assurée car le nombre de valeurs abstraites est fini.

L'interprétation des expressions arithmétiques est triviale :

Addition :

| □ $\bar{+}$ x → □ ;
| x $\bar{+}$ □ → □ ;
| pair $\bar{+}$ pair → pair ;
| pair $\bar{+}$ impair → impair ;
| impair $\bar{+}$ pair → impair ;
| impair $\bar{+}$ impair → pair ;
| x $\bar{+}$ entier → entier ;
| entier $\bar{+}$ x → entier ;

etc ....

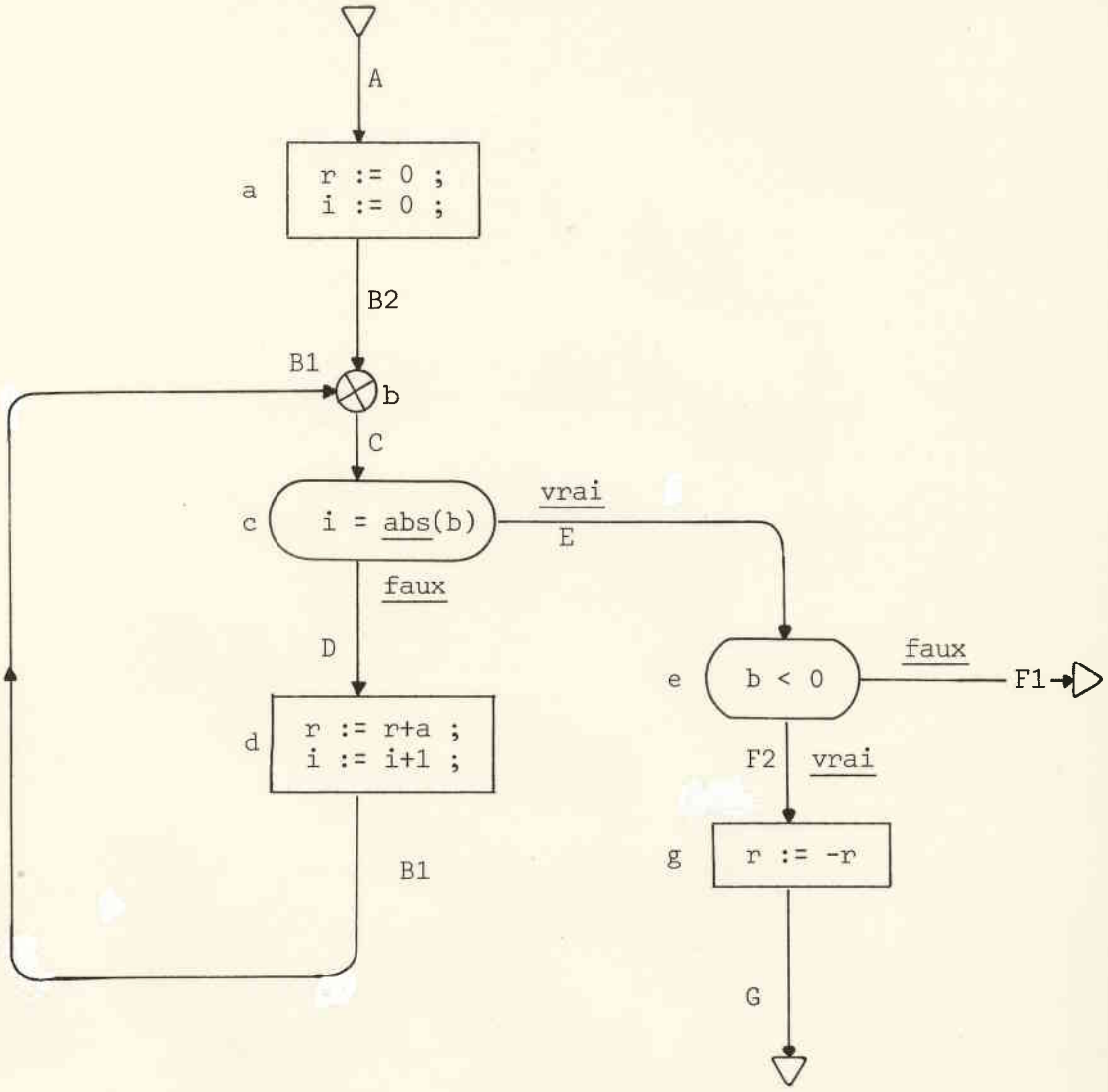
- L'interprétation des tests est également simple puisque $<$, $\leq$, $\geq$, $>$, $\neq$, in n'apportent aucun renseignement, tandis que l'égalité en apporte :

{x = y} ∧ {(x,pair)} ⇒ y pair

{x = y} ∧ {(x,pair),(y,impair)} est impossible,

etc ....

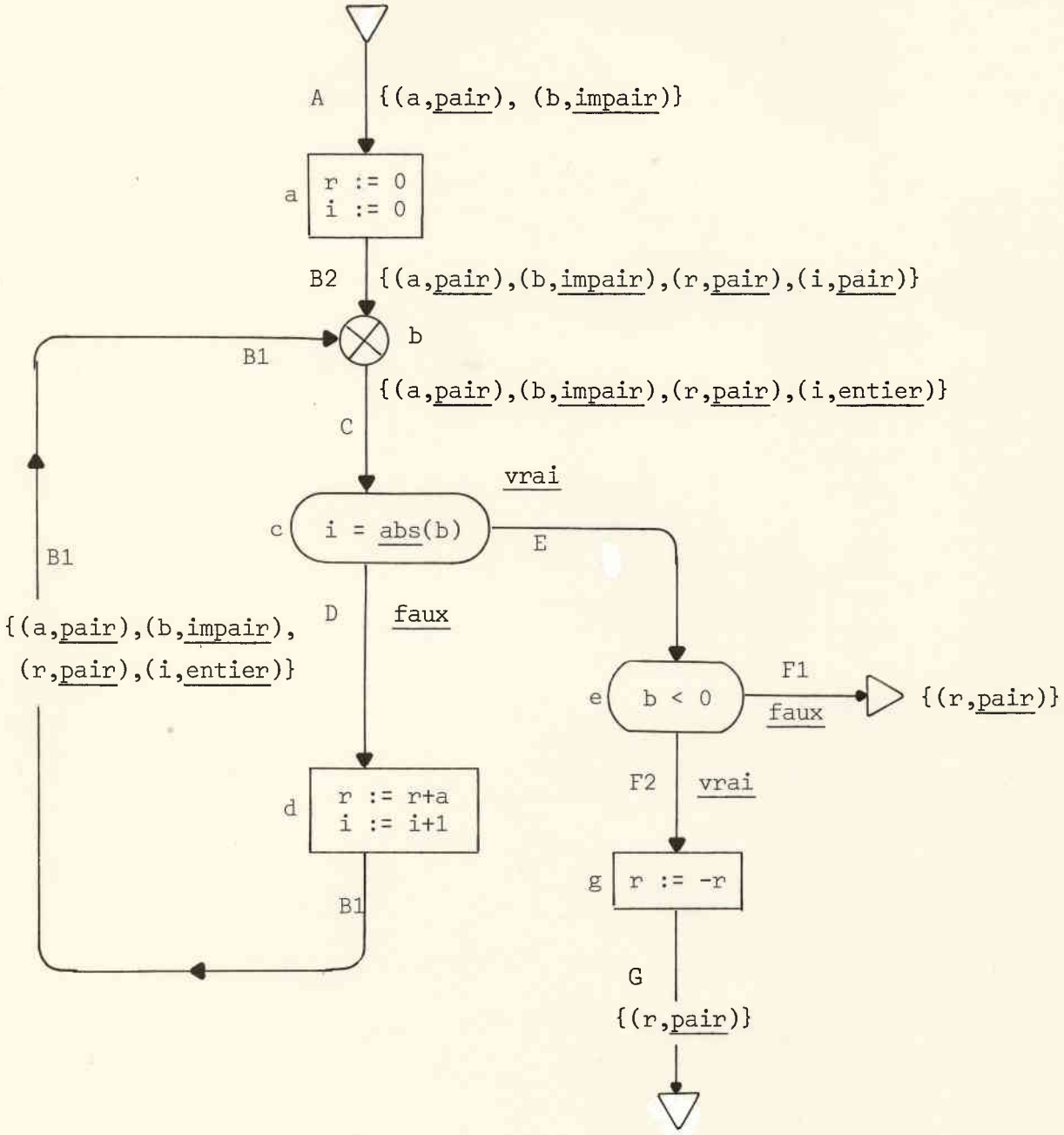
Prenons l'exemple du programme suivant, qui calcule a × b :



Avec un contexte initial, a est pair, b est impair, on obtient :

Pas	Arc	Contexte local (arc)	Noeud traité	Contexte de l'arc de sortie du noeud traité	Chemins à exécuter	Jonction
0					A	Ø
1	(A)	{(a, pair), (b, impair)}	{a} affect.	{(a, pair), (b, impair), (r, pair), (i, pair)}	B2	Ø
2	B2	{(a, pair), (b, impair), r, pair), (i, pair)}	{b} jonct.			{b}
	B1	Φ				Ø
	B2	{(a, pair), (b, impair), (r, pair), (i, pair)}	b jonct.	{(a, pair), (b, impair), (r, pair), (i, pair)}	C	Ø
3	C	{(a, pair), (b, impair), (r, pair), (i, pair)}	{c} test	Φ	E	Ø
				{(a, pair), (b, impair), (r, pair), (i, pair)}	D	Ø
4	D	{(a, pair), (b, impair), (r, pair), (i, pair)}	{d} affect.	{(a, pair), (b, impair), (r, pair), (i, impair)}	B1	Ø
5	B1	{(a, pair), (b, impair), (r, pair), (i, impair)}	{b} jonct.			{b}
	B1	{(a, pair), (b, impair), (r, pair), (i, impair)}				Ø
	B2	{(a, pair), (b, impair), (r, pair), (i, pair)}	{b} fonct.	{(a, pair), (b, impair), (r, pair), (i, entier)}	C	Ø
6	(C)	{(a, pair), (b, impair), (r, pair), (i, entier)}	{c} test	{(a, pair), (b, impair), (r, pair), (i, impair)}	E	Ø
				{(a, pair), (b, impair), (r, pair), (i, entier)}	D	Ø
7	(D)	{(a, pair), (b, impair), (r, pair), (i, entier)}	{d} affect.	{(a, pair), (b, impair), (r, pair), (i, entier)}	B1	Ø
8	B1	{(a, pair), (b, impair), (r, pair), (i, entier)}	{b} jonct.		Ø	{b}
9	(E)	{(a, pair), (b, impair), (r, pair), (i, impair)}	{e} test	{(a, pair), (b, impair), (r, pair), (i, impair)}	F1	Ø
				{(a, pair), (b, impair), (r, pair), (i, impair)}	F2	Ø
10	(F1)	{(a, pair), (b, impair), (r, pair), (i, impair)}	sortie			Ø
11	(F2)	{(a, pair), (b, impair), (r, pair), (i, impair)}	{g} affect.	(a, pair), (b, impair), (r, pair), (i, impair)	G	Ø
12	(G)	{(a, pair), (b, impair), (r, pair), (i, impair)}	sortie			Ø
	(B1)	{(a, pair), (b, impair), (r, pair), (i, entier)}				Ø
	(B2)	{(a, pair), (b, impair), (r, pair), (i, pair)}	{b} jonct.	(a, pair), (b, impair), (r, pair), (i, entier)	C	Ø
13	C	arrêt				Ø

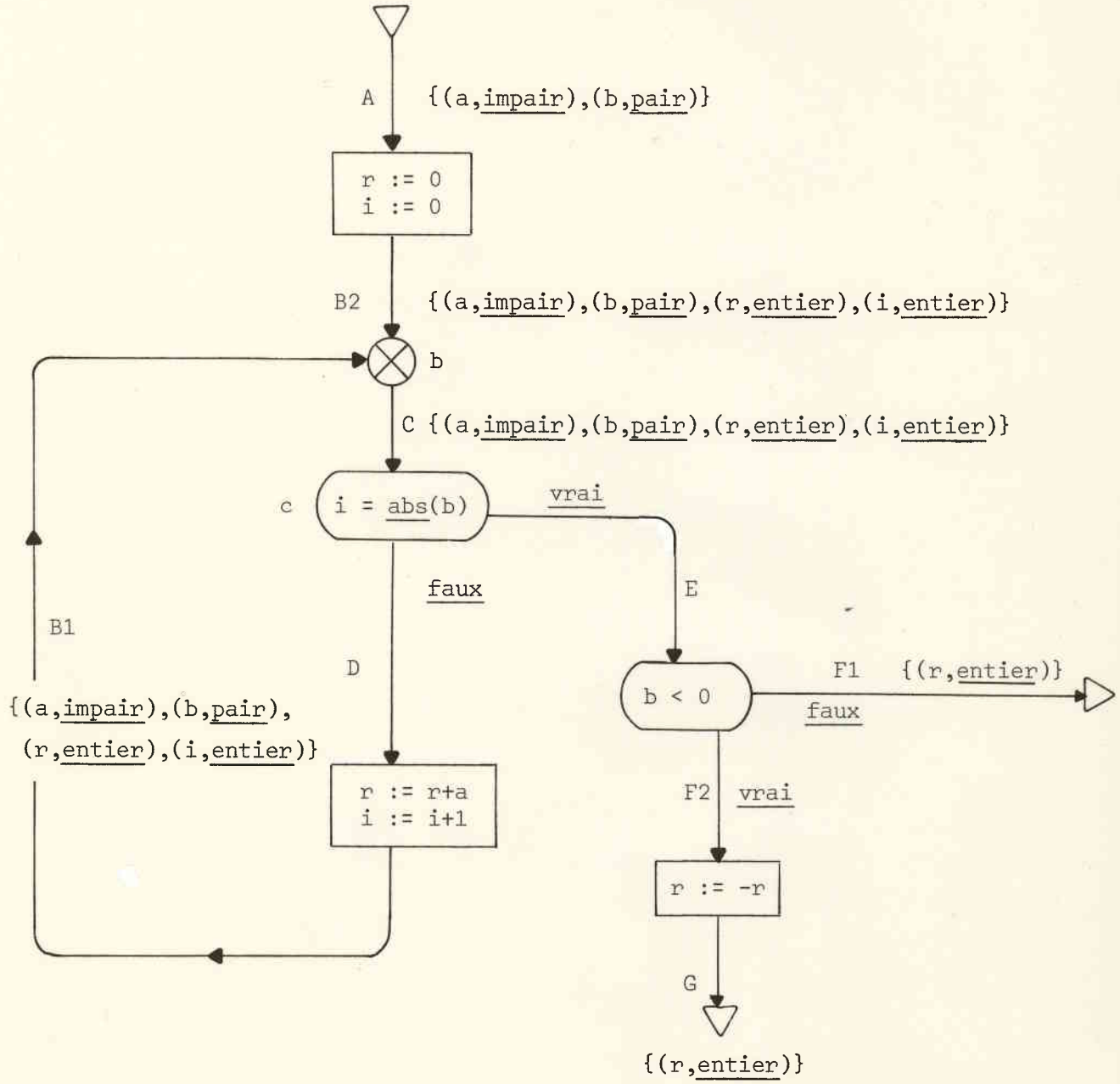
Le résultat exprime que le produit d'un nombre pair par un nombre impair est un nombre pair :



Avec un contexte initial, a est impair, b est pair, on obtient :

Pas	Arc	Contexte local (arc)	Noeud traité	Contexte de l'arc de sortie du noeud traité	Chemins à exécuter	Jonctio
0					A	ø
1	(A)	{(a, impair), (b, pair)}	{a} affect.	{(a, impair), (b, pair), (r, pair), (i, pair)}	B2	
2	B2	{(a, impair), (b, pair), (r, pair), (i, pair)}	{b} jonct.		ø	{b}
	B1	ø	b			
	B2	{(a, impair), (b, pair), (r, pair), (i, pair)}	jonct.	{(a, impair), (b, pair), (r, pair), (i, pair)}	C	ø
3	C	{(a, impair), (b, pair), (r, pair), (i, pair)}	{c} test	{(a, impair), (b, pair), (r, pair), (i, pair)}	E	ø
				{(a, impair), (b, pair), (r, pair), (i, pair)}	D	
4	E	{(a, impair), (b, pair), (r, pair), (i, pair)}	{e} test	{(a, impair), (b, pair), (r, pair), (i, pair)}	F1	ø
				{(a, impair), (b, pair), (r, pair), (i, pair)}	F2	
5	F1	{(a, impair), (b, pair), (r, pair), (i, pair)}	sortie		F2	ø
6	F2	{(a, impair), (b, pair), (r, pair), (i, pair)}	{g} affect.	{(a, impair), (b, pair), (r, pair), (i, pair)}	G	ø
7	G	{(a, impair), (b, pair), (r, pair), (i, pair)}	sortie		D	ø
8	D	{(a, impair), (b, pair), (r, pair), (i, pair)}	{d} affect.	{(a, impair), (b, pair), (r, impair), (i, impair)}	B1	
9	B1	{(a, impair), (b, pair), (r, impair), (i, impair)}	{b} jonct.			{b}
	B2	{(a, impair), (b, pair), (r, pair), (i, pair)}				
	B1	{(a, impair), (b, pair), (r, impair), (i, impair)}	{b}	{(a, impair), (b, pair), (r, entier), (i, entier)}	C	
10	(C)	{(a, impair), (b, pair), (r, entier), (i, entier)}	{c} test	{(a, impair), (b, pair), (r, entier), (i, pair)}	E	
				{(a, impair), (b, pair), (r, entier), (i, entier)}	D	
11	(E)	{(a, impair), (b, pair), (r, entier), (i, pair)}	{e} test	{(a, impair), (b, pair), (r, entier), (i, pair)}	F1	
				{(a, impair), (b, pair), (r, entier), (i, pair)}	F2	
12	(F1)	{(a, impair), (b, pair), (r, entier), (i, pair)}	sortie			
13	(F2)	{(a, impair), (b, pair), (r, entier), (i, pair)}	{g} affect.	{(a, impair), (b, pair), (r, entier), (i, pair)}	G	
14	(G)	{(a, impair), (b, pair), (r, entier), (i, pair)}	sortie			
15	(D)	{(a, impair), (b, pair), (r, entier), (i, entier)}	{d} affect.	{(a, impair), (b, pair), (r, entier), (i, entier)}	B1	
	B1	{(a, impair), (b, pair), (r, entier), (i, entier)}	{b} jonct.			{b}
	(B2)	{(a, impair), (b, pair), (r, pair), (i, pair)}				
	(B1)	{(a, impair), (b, pair), (r, entier), (i, entier)}	{b} jonct.	{(a, impair), (b, pair), (r, entier), (i, entier)}	C	
16	C	arrêt				

Cette fois-ci, le résultat est plus décevant, puisque le produit d'un nombre impair par un nombre pair est un entier :



Intuitivement, ce résultat s'explique par le fait que la description du produit a × b n'est pas symétrique en a et b. Si l'on étudie la trace de l'algorithme d'interprétation abstraite de plus près, on s'aperçoit que sur l'arc B1, r est pair quand i est pair, tandis que r est impair quand i est impair. On peut donc imaginer une description du contexte de l'arc C sous la forme :

{(a,impair), (b,pair), (r,pair), (i,pair)} $\bar{\cup}$ {(a,impair), (b,pair), (r,impair), (i,impair)}.

Dans ce cas, d'après le test du noeud c, l'arc E n'est suivi que lorsque i est égal à abs(b), b étant tout le temps pair, l'arc E est suivi uniquement quand i est pair, et dans ce cas, d'après le contexte local de C, r est pair en E, donc également en F1 et G.

On comprend que c'est le traitement de l'union et l'extension de contextes aux noeuds de jonction qui fait perdre la relation entre valeurs abstraites de r et i. (Pas 9 de la trace). On imagine également que si la structure des contextes permettait de distinguer entre plusieurs alternatives, on pourrait mémoriser ces relations entre variables, et finalement obtenir une analyse plus fine des programmes. Par exemple, SINTZOFF [3-b] propose une définition des contextes de la forme $(Q \wedge \bigwedge_i x_i = V_i)$, où Q est une assertion, chaque x_i est un identificateur du programme et V_i sa valeur abstraite. L'union de contextes est définie par

$$(Q \wedge \bigwedge_i x_i = V_i) \bar{\cup} (Q' \wedge \bigwedge_i x_i = V'_i)$$

$$\Rightarrow (Q \vee Q', \bigwedge_i (x_i = [Q \rightarrow V_i \mid Q' \rightarrow V'_i \mid \underline{\text{true}} \rightarrow \square]))$$

où [... → ... | ... → ... | ... → ...] est l'expression conditionnelle de Mac-Carthy. Cette approche permet donc d'exprimer des relations entre les domaines de valeurs des variables. Malheureusement, son implémentation n'est guère envisageable dès à présent, puisqu'elle implique des démonstrations automatiques. Toutefois, l'idée d'exprimer des relations entre variables, au travers d'alternatives entre contextes est applicable à notre algorithme, son étude est actuellement commencée, et fera l'objet d'un prochain document.

5°) - CONCLUSION -

□ Divers approfondissements de cette étude sont nécessaires avant d'envisager une implémentation :

a - Extension de l'algorithme d'interprétation abstraite, au cas des définitions et appels de fonctions récursives ou non.

b - Etude de l'interprétation élémentaire des noeuds tests, dans le cas général.

c - Etude d'une structuration des contextes abstraits permettant d'exprimer des relations entre valeurs abstraites des variables, sous la forme d'alternatives entre contextes abstraits, tels qu'ils sont définis actuellement.

d - Etude des constructions des langages ayant des "effets de bord".

□ En dehors des points a, b, c et d qui, à notre avis, doivent précéder une implémentation, pour qu'elle soit digne d'intérêt, nous envisageons également :

e - L'étude d'un algorithme de vérification de types évolués, dont l'exemple 4.3 donne un avant-goût.

f - L'extraction des programmes d'invariants liant les variables des programmes.

□ Tous les résultats que nous avons présentés quant au domaine des variables à l'exécution, est conditionnel à la bonne initialisation de cette variable. Ainsi, si sur un arc a, la variable x a la valeur abstraite v, toute exécution du programme conduit sur cet arc à des valeurs concrètes r de x telles que $r \in \gamma(v)$. Mais on a toujours $\square \underline{\leq} v$, donc $\gamma(\square) \subseteq \gamma(v)$, donc il se peut que x soit non initialisé. Les résultats sont à interpréter comme suit :

1 - Si $v = \square$, alors x n'est pas initialisé, ou l'arc a est une branche morte.

2 - Si $v \neq \square$, alors si x est initialisé, on a $r \in \gamma(v)-\gamma(\square)$.

En fait, le problème de l'initialisation des variables est complexe , et peut être résolu de deux façons :

- une meilleure définition des langages, ou une meilleure méthodologie de programmation ;

- une vérification statique de l'initialisation qui suppose dans notre approche d'avoir résolu les points a, b, c, d, e et f.

En conclusion, nous pensons qu'il est trop tôt pour aborder ce problème dans l'état actuel des connaissances.

- On a vu également comment l'interprétation abstraite permet d'éliminer les tests de cohérence dynamiques inutiles. Toutefois, nous n'avons pas abordé le problème de la répartition optimale des tests dans le programme. Là encore, nous n'envisageons pas d'étudier ce problème lié aux techniques d'optimisation et d'évaluation des performances des programmes.

- Enfin, d'autres tentent d'obtenir les mêmes résultats que nous par des méthodes d'interprétation symbolique [9] ou de preuves mécaniques de programmes [10]. Cette approche nous semble trop complexe pour être fructueuse dès maintenant.

REMERCIEMENTS :

Nous remercions P. JORRAND, S. SCHUMAN, D. BERT, B. LORHO et H. SAYA pour les discussions que nous avons eues sur ce travail.

BIBLIOGRAPHIE

[1] N. WIRTH
"The programming language PASCAL".
Acta Informatica, Vol. 1, pp. 35-63 (1971).
Springer - Verlag - 1971.

[2] HABERMAN A.N.
"Critical Comments on the Programming Language PASCAL".
Acta Informatica, Vol. 3, pp. 47-57 (1973).
Springer - Verlag - 1973.

[3] SINTZOFF M.
a) "Calculating properties of programs by valuations on specific models".
Proc. ACM conf. on proving assertions about programs.
SIGPLAN Notices,Vol.7, Nb 1, Jan. 1972, pp. 203-207.

b) "Vérification d'assertions pour des fonctions utilisables comme valeurs et affectant des variables extérieures".
Construction, amélioration et vérification de programmes. IRIA - Arc et Senans - 1-3 Juillet 1975, pp. 12-27.

[4] KILDALL G.A.
a) "A unified approach to global program optimization".
Conference Record of ACM symposium on principles of programming languages.
pp. 194-206.
Boston, Massachusetts, October 1-3 1973.

b) "Global expression optimization during compilation".
These Doctor of Philosophy, 1972, 163 p.
University of Washington, Computer Science Group, TR 72-06-02.

[5] WEGBREIT B.
"Property extraction in well-founded property sets".
Center for Research in Computing Technology Harvard University.
Cambridge, Massachusetts, Feb. 1973, 44 pp.

[6] KARR M.

a) "On affine Relationships Among Variables of a Program".
CA-7402-2811, Feb. 28, 1974.
Massachusetts Computer Associates, Inc.
Lakeside Office Park, Wakefield, Mass. 01880, U.S.A.

b) "Proving Inequalities".
CA-7406-1011, June 10, 1974, revised April 15, 1975.
Massachusetts Computer Associates, Inc.
26 Princess Street, Wakefield, Mass. 01880, U.S.A.

[7] ALLEN F.
"Control flow analysis".
Proceedings of a Symposium on Compiler Optimization.
University of Illinois at Urbana-Champaign, July, 1970.

[8] RIS F.N.
"Tools for the analysis of interval arithmetic".
RC 5305 - IBM Thomas J. Watson Research Center Yorktown Heights, New York, 10598, U.S.A. - March 11, 1975.

[9] KING J.C.
"Symbolic execution and program testing".
RC 5082 - IBM Thomas J. Watson Research Center,
Yorktown Heights, New York, 10598, U.S.A., October 18, 1974.

[10] Colloques IRIA
Construction, amélioration et vérification de programmes
Arc et Senans, 1-3 Juillet 1975.
IRIA, Rocquencourt, 78150 Le Chesnay.

[11] BERGE C.
"Graphes et hypergraphes".
Dunod Université - DUNOD - Paris 1973.

[12] MAGHOUT K.
"Sur la détermination des nombres de stabilité et du nombre chromatique d'un graphe".
C.R. Acad. Sc. Paris, 248, 1959, pp. 3522-3523.