

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 ($\alpha x + \beta y + \dots = \gamma$) entre des variables (x, y, ...) d'un programme [6].

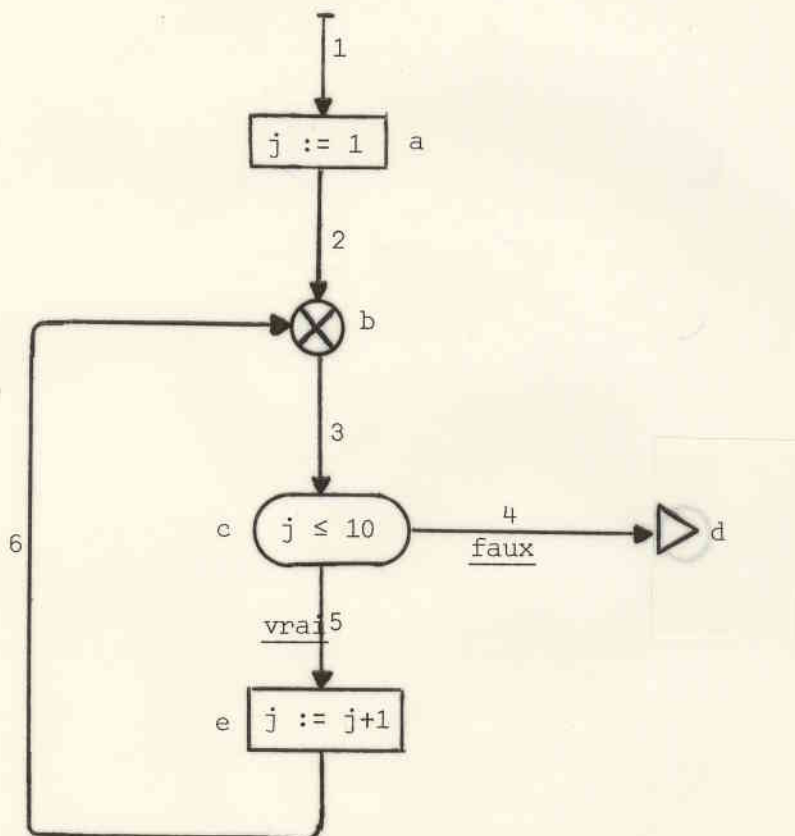
Dans notre cas, cette interprétation abstraite des programmes consiste à "exécuter" ces programmes, non pas avec des valeurs associées aux variables, mais avec des domaines de valeurs associées aux variables. On peut ainsi déterminer quel est le domaine des valeurs possible pour les variables en chaque 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), différent 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, $a, a+1, a+2, \dots, b$ ($b \geq a$).

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.

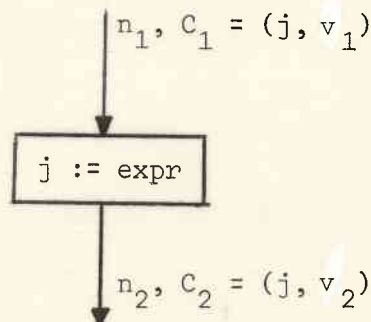
La construction des contextes abstraits associés aux arcs du programme se fait pas à pas, comme suit : on commence par l'arc d'entrée du programme, avec un certain contexte initial, et le contexte vide sur les autres arcs. Pour chacun des trois types de noeuds (affectation, décision, jonction), on décrit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

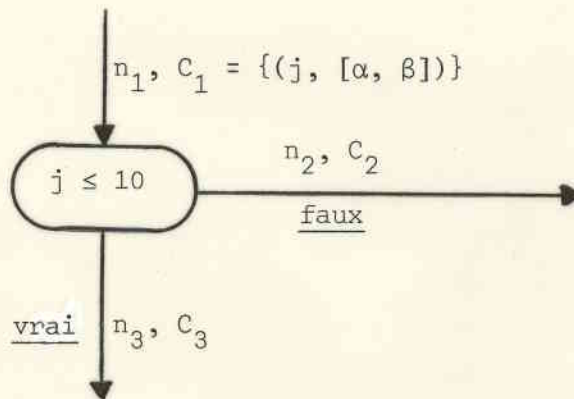
une valeur "v", qui est affectée à j, c'est-à-dire que la valeur "v₂" associée à j dans le contexte C₂ est remplacée par "v" (ou un couple (j, v) est introduit dans C₂ si j n'avait pas de valeur dans le contexte C₂).

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 [-∞, +∞]).

Une expression e somme de deux expressions e₁ + e₂ s'évalue, en évaluant e₁ ce qui donne [α, β], en évaluant collatéralement e₂ ce qui donne [γ, δ] puis en délivrant la valeur abstraite [α + γ, β + δ].

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 n₃, avec le contexte C₃ égal à C₁, puisqu'aucun élément du contexte C₁ n'a été modifié par la comparaison.

2ème cas : $\alpha > 10$, on continue l'interprétation par n₂, avec C₂ égal à C₁.

3ème cas : $\alpha \leq 10 < \beta$

Dans une interprétation réelle, l'arc n₃ sera suivi si $j \leq 10$, donc pour les valeurs de j comprises entre α et 10, tandis que l'arc n₂ 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

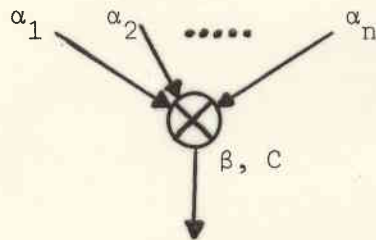
n₃, C₃ = {(j, [α, 10])} et n₂, C₂ = {(j, [11, β])}. Le choix du chemin à suivre en premier n'a aucune importance. On voit donc, qu'à cause des noeuds de

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



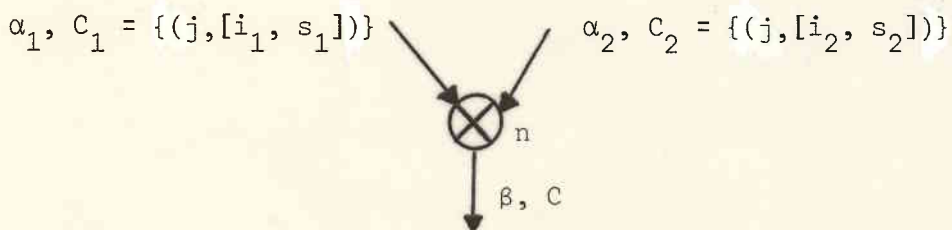
(auquel cas, on s'arrête et l'on choisit de poursuivre avec un autre chemin candidat à l'exécution) ;

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 $\alpha_1, \dots, \alpha_n$. 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 $C' = C_1 \bar{\cup} C_2$, 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 $C' \bar{\subset} C$, (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 $C \bar{\cup} C'$. (A la première itération avec $\Phi \bar{\cup} C' = C'$). Noter, qu'après m passages sur l'arc β , on utilise le contexte $(\dots ((\Phi \bar{\cup} C'_1) \bar{\cup} C'_2) \dots \bar{\cup} C'_m)$ sur cet arc. Pour que le processus termine, il faut que la suite $C_0 = \Phi, C_1 = (\Phi \bar{\cup} C'_1), C_2 = ((\Phi \bar{\cup} C'_1) \bar{\cup} C'_2), \dots, C_m = (\dots ((\Phi \bar{\cup} C'_1) \bar{\cup} C'_2) \dots \bar{\cup} C'_m)$ 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 $C_0 = \Phi, C_1 = C'_1 = \{(j, [1, 1])\}, \dots, C_i = C'_i \bar{\cup} C'_{i-1} = \{(j, [1, i])\}$, suite non convergente. Pour assurer cette convergence, on remplacera C par $C \bar{\cap} C'$, avec

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

On a maintenant :

$$C_0 = \emptyset$$

$$C_1 = \emptyset \bar{\vee} \{(j, [1, 1])\} = \{(j, [1, 1])\}$$

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

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

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

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

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

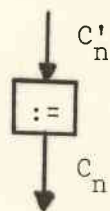
$$= \{(j, [1, +\infty] \bar{\vee} [1, 3])\} = \{(j, [1, +\infty])\}$$

$$= C_2$$

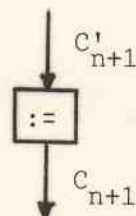
On a évidemment une perte d'information, mais les exemples que nous donnerons confirment que ce choix est judicieux, en particulier, nous démontrerons que l'interprétation abstraite de tous les programmes termine, même si ces programmes ne terminent pas dans les exécutions réelles.

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 :

$n^{\text{ème}}$ itération



$(n+1)^{\text{ème}}$ itération



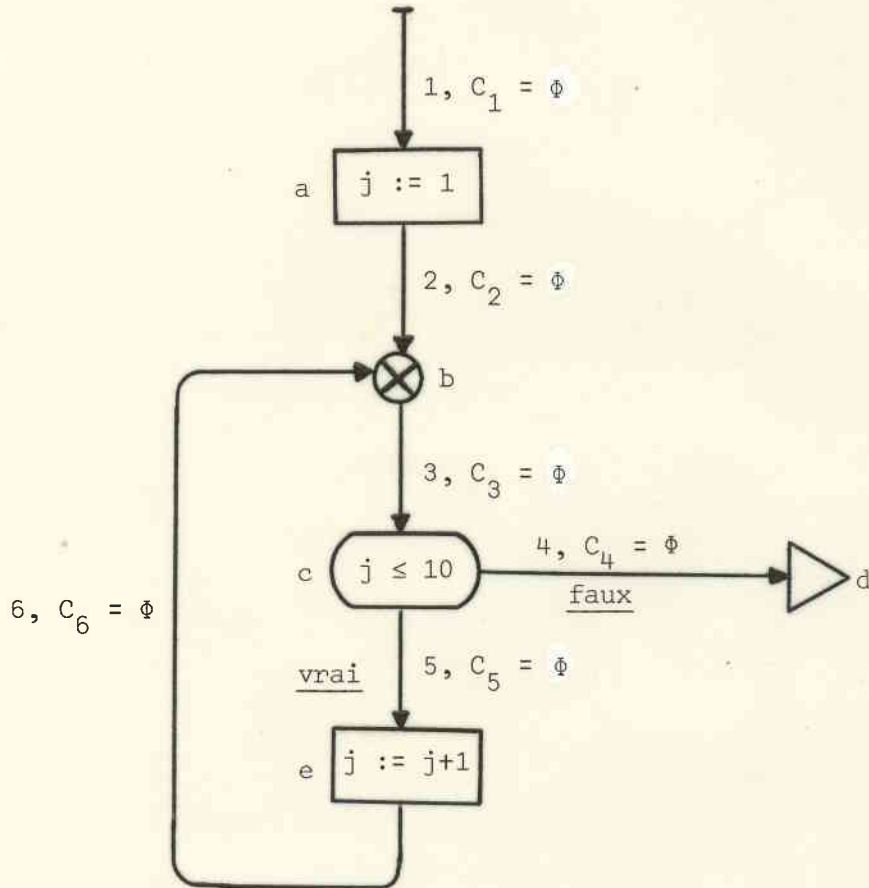
si une itération supplémentaire a été nécessaire c'est que C'_{n+1} n'est pas inclus dans C'_n .

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 $C'_n \bar{\vee} C'$. 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 :

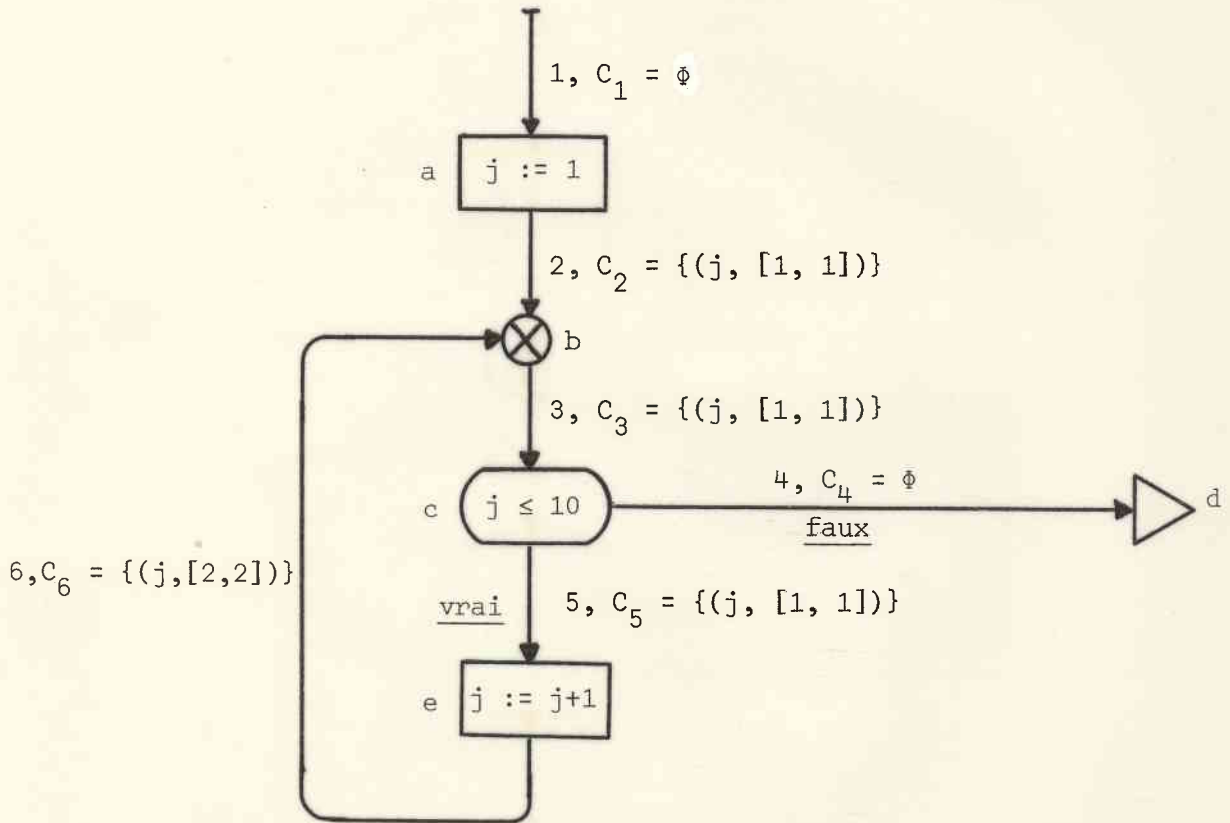
$$\begin{aligned}
 C' &= C_6 \bar{\cup} C_2 \\
 &= \Phi \bar{\cup} \{(j, [1, 1])\} = \{(j, [1, 1])\}
 \end{aligned}$$

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

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 :

$$\begin{aligned} C_6 &= \{(j, [1, 1] + [1, 1])\} \\ &= \{(j, [2, 2])\} \end{aligned}$$

On obtient, en attendant sur le noeud b :



Le contexte de sortie sur le noeud b, est

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

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

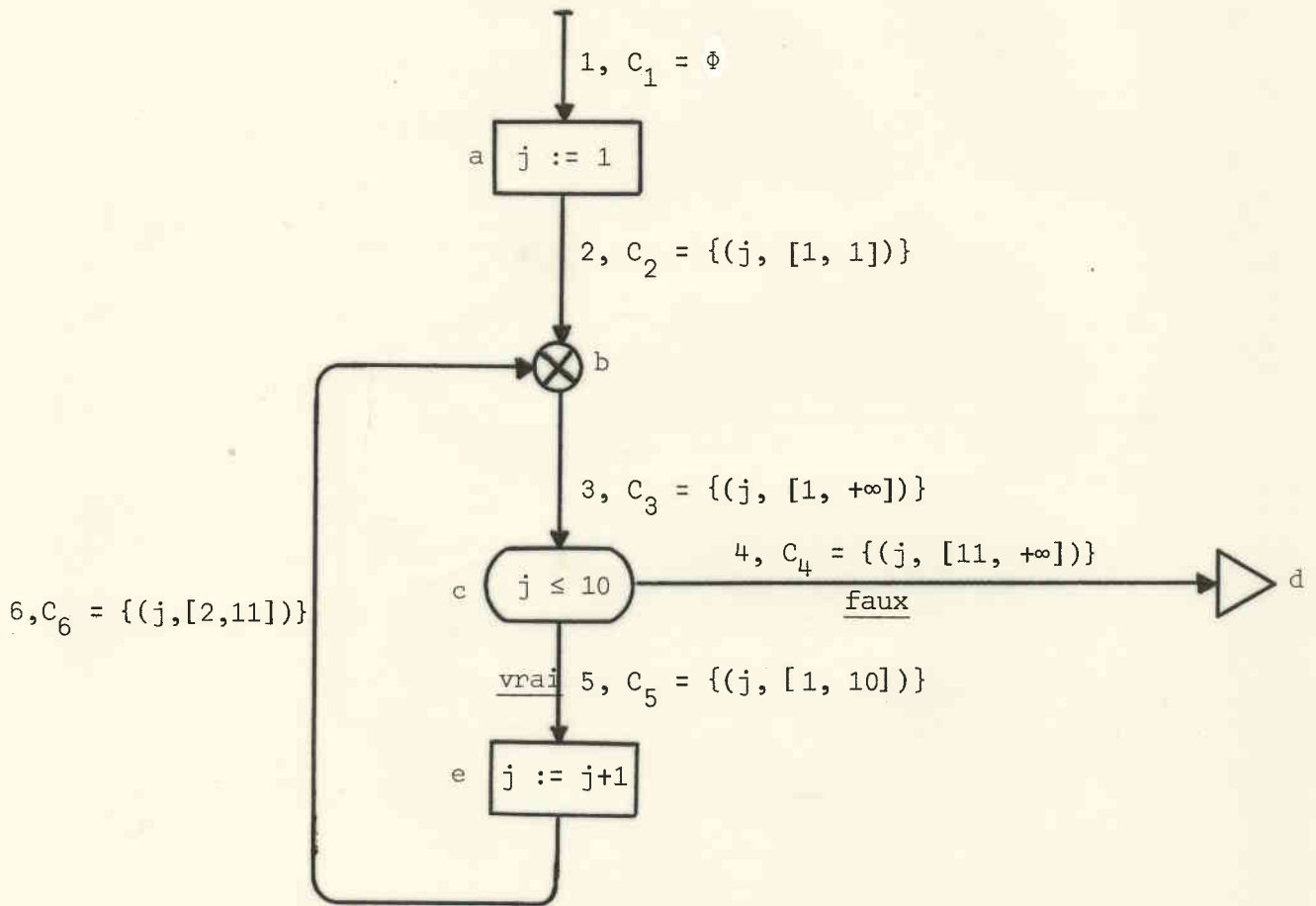
$$\begin{aligned} C_3 &= \{(j, [1, 1])\} \bar{\vee} \{(j, [1, 2])\} \\ &= \{(j, [1, 1] \bar{\vee} [1, 2])\} = \{(j, [1, +\infty])\} \end{aligned}$$

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 :

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

On obtient, en attente sur le noeud b :



Le contexte de sortie sur le noeud b, est :

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

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'interprétation 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'interprétation 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 :

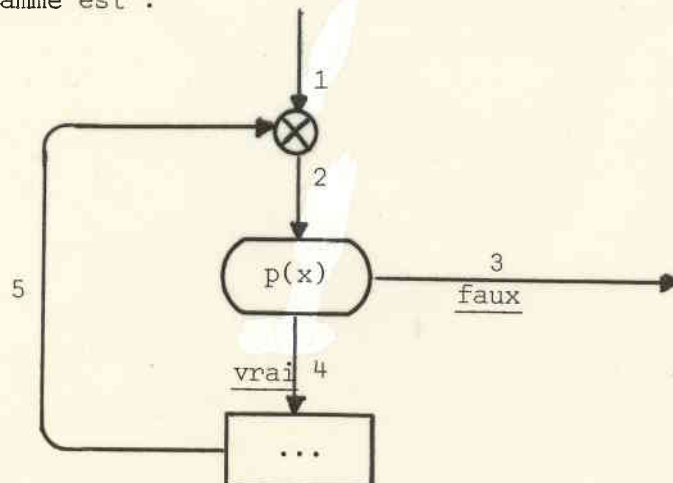
```
if (  $j < 2$  )  $\vee$  (  $j < 11$  ) then  
    write "erreur de programmation" ;  
    stop ;  
fi ;
```

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 :

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

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

$$A2 \quad : (x \cup (y \cup z)) = ((x \cup y) \cup z) \quad \text{associativité}$$

$$A3 \quad : (x \cup y) = (y \cup x) \quad \text{commutativité}$$

$$A4 \quad : (x \cup x) = x \quad \text{idempotence}$$

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

$$PDEF1 \quad : \{x \leq y\} \stackrel{\text{def}}{\iff} \{x \cup y = y\}$$

$$PDEF2 \quad : \{x < y\} \stackrel{\text{def}}{\iff} \{(x \leq y) \wedge (x \neq y)\}$$

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

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

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

lemme P1

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

réflexivité :

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

antisymétrie :

$$\begin{aligned}
 & (x \leq y) \wedge (y \leq x) \\
 \text{(PDEF1)} \Rightarrow & ((x \cup y) = y) \wedge ((y \cup x) = x) \\
 \text{(A3)} \Rightarrow & ((x \cup y) = y) \wedge ((x \cup y) = x) \\
 \Rightarrow & y = x
 \end{aligned}$$

transitivité :

$$\begin{aligned}
 & (x \leq y) \wedge (y \leq z) \\
 \text{(PDEF1)} \Rightarrow & ((x \cup y) = y) \wedge ((y \cup z) = z) \quad (1) \\
 \Rightarrow & ((x \cup y) \cup z) = z \\
 \text{(A2)} \Rightarrow & (x \cup (y \cup z)) = z \\
 (1) \Rightarrow & (x \cup z) = z \\
 \text{(PDEF1)} \Rightarrow & x \leq z
 \end{aligned}$$

lemme P2

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

$$\begin{aligned}
 & ((x \cup y) = (x \cup y)) \wedge ((x \cup x) = x) \quad (A4) \\
 \Rightarrow & ((x \cup x) \cup y) = (x \cup y) \\
 \text{(A2)} \Rightarrow & (x \cup (x \cup y)) = (x \cup y) \\
 \text{(PDEF1)} \Rightarrow & x \leq (x \cup y)
 \end{aligned}$$

lemme P3

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

$$\begin{aligned}
 \text{(P2)} \Rightarrow & x \leq x \cup y \quad (1) \\
 \text{(A7)} \Rightarrow & (x \cup y) \leq (x \nabla y) \\
 \text{(P1, transitivité)} \Rightarrow & x \leq x \nabla y \\
 (1) \text{ et (A3)} \Rightarrow & x \leq (y \cup x) \\
 \text{(A7)} \Rightarrow & (y \cup x) \leq (y \nabla x) \\
 \text{(P1, transitivité)} \Rightarrow & x \leq y \nabla x
 \end{aligned}$$

lemme P4

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

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

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

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

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

(2), (3) (P1, antisymétrie) $\Rightarrow x = x \cup y$, négation de l'hypothèse $x \not\leq x \cup y$, donc absurdité.

lemme P5

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

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

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

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

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

lemme P6

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

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 $y \not\leq y \cup x$ (1)

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

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

b) \in (1), (P4) $\Rightarrow y \not\leq y \vee x$ (3)

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

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

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

lemme P7

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

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

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

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

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

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

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

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

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

lemme P10

{l'opération \cup est compatible avec la relation d'ordre partiel \leq }
 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 $x \cup \emptyset = \emptyset$

$$(A5) \quad x = \emptyset$$

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

lemme P12

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

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

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

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

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

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

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

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

$$(A5) \quad \Rightarrow y = \emptyset$$

lemme P13

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

$$\neg(x \leq y)$$

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

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

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

$$(PDEF2) \quad \Rightarrow y \nmid y \cup (y \cup x)$$

$$(A3) \quad \Rightarrow y \nmid (y \cup x) \cup y$$

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

Pour le deuxième volet du lemme, on a :

$$\neg(x \leq y)$$

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

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

$$(P4) \quad \Rightarrow y \dagger (y \nabla x) \quad (1)$$

mais (P3) $\Rightarrow y \leq (y \nabla x)$

$$(PDEF2) \Rightarrow y \cup (y \nabla x) = (y \nabla x) \quad (2)$$

$$(1), (2) \Rightarrow y \dagger y \cup (y \nabla x)$$

$$(A3) \quad \Rightarrow y \dagger (y \nabla x) \cup y$$

$$(PDEF2) \quad \neg((y \nabla x) \leq y)$$

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 :

$$X \subset \mathbb{N} \quad , \quad @(X) = \left[\begin{array}{c} \text{MIN}(x) \\ x \in X \end{array} \quad , \quad \begin{array}{c} \text{MAX}(x) \\ x \in X \end{array} \right]$$

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 :

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

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

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

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

$$V_a \times V_a \xrightarrow{\quad \bar{u} \quad} V_a$$

Hypothèse - H3

$$\{ @ \text{ est un homomorphisme de } (2^{V_c}, \cup) \text{ dans } (V_a, \bar{u}) \}$$
$$\stackrel{\text{def}}{\iff} \{ \forall (v_1, v_2) \in (2^{V_c})^2, @ (v_1 \cup v_2) = @ (v_1) \bar{u} @ (v_2) \}$$

Définition - DEF4 - γ : concrétisation de valeurs abstraites

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

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

Hypothèse - H5

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

Nous utiliserons également cette hypothèse sous la forme :

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

Hypothèse - H6

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

Lemme - L7

- a) - $\{ @, \text{ est surjective} \} \stackrel{\text{def}}{\iff} \{ \forall y \in V_a, \exists x \subseteq V_c \mid y = @(x) \}$
- b) - $\{ \gamma \text{ est injective} \} \stackrel{\text{def}}{\iff} \{ \{ \forall (x, y) \in V_a^2 \mid \gamma(x) = \gamma(y) \} \Rightarrow \{ x = y \} \}$
- $\stackrel{\text{def}}{\iff} \{ \{ \forall (x, y) \in V_a^2 \mid x \neq y \} \Rightarrow \{ \gamma(x) \neq \gamma(y) \} \}$

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{u} admet un élément neutre

$$\{ \exists \square \in V_a \mid \forall x \in V_a, x \bar{u} \square = \square \bar{u} x = x \}$$

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

Lemme L9 — \bar{u} est idempotente

$$\{ \forall x \in V_a, x \bar{u} x = x \}$$

Lemme L10 — \bar{u} est commutative

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

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

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

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

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

Définition - DEF 13

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

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 :

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

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

donc :

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

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 :

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

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

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

Hypothèse - H14

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

Remarque : L'ensemble V_a muni des opérations $\bar{\cup}$ et $\bar{\vee}$ 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 $(V_a, \bar{\cup}, \bar{\vee})$ les lemmes P1, ..., P13. Nous utiliserons également les propriétés suivantes :

Lemme - L15

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

$$x \in \gamma(v_1)$$

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

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

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

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

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

Lemme - L16

$$\forall (v_1, v_2) \in V_a^2$$

$$\{(v_1 \bar{\leq} v_2) \wedge (x \in \gamma(v_1))\} \Rightarrow \{x \in \gamma(v_2)\}$$

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

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

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

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

Hypothèse - H17

$$\{ \forall (v_1, \dots, v_n, \dots) \in V_a^\infty, \\ \forall (s_0, \dots, s_n, \dots) \in V_a^\infty \mid \\ (s_0 = \square, s_1 = s_0 \bar{\vee} v_1, \dots, s_n = s_{n-1} \bar{\vee} v_n, \dots) \}$$

\Rightarrow {la suite $s_0, s_1, \dots, s_n, \dots$ n'est pas infinie et strictement croissante pour $\bar{<}$ }

Dans notre exemple introductif, on pourrait avoir :

- $\square < \square \bar{\vee} [1, 3] = [1, 3]$
- $< [1, 3] \bar{\vee} [2, 5] = [1, +\infty]$
- $< [1, +\infty] \bar{\vee} [0, 1] = [-\infty, +\infty]$

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 diffèrent 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 \emptyset
- 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 $C(i)$,

$$C(i) \stackrel{\text{def}}{\iff} \{ \underline{\text{si}} (\exists v \in (V_a - \{\square\}) \mid (i, v) \in C) \underline{\text{alors}} v \\ \underline{\text{sinon}} \square \\ \underline{\text{finsi}} ; \}$$

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

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

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

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

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

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) - $\bar{\cup}$: union de contextes abstraits

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

$$\{C_1 \bar{\cup} C_2\} \stackrel{\text{def}}{\iff}$$

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

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

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

$$\{C_1 \bar{\bar{v}} C_2\} \stackrel{\text{def}}{\iff}$$

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

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

Lemme - L 105

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

Remarquons tout d'abord que $C_1 \bar{\bar{u}} C_2$ 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

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

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

Lemme - L 106

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

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

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

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

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

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

Comme \bar{U} 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 :

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

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

Lemme L107

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

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

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

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

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

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

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

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

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

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

Lemme L108

- \bar{U} est 1) associative
- 2) commutative
- 3) idempotente
- 4) \emptyset est son élément neutre

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

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

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

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

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

(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).

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

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 :

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

$$3^{\circ}) - \forall C \in \mathcal{C}$$

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

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

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

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

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

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

$$= C$$

$$4^{\circ}) - \forall \Phi \in \mathcal{C},$$

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

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

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

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

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

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

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

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

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

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

$$\{C_1 \bar{\leq} C_2\} \stackrel{\text{def}}{\iff} \{C_1 \bar{\cup} C_2 = C_2\}$$

Définition - DEF 110

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

$$\{C_1 \bar{<} C_2\} \stackrel{\text{def}}{\iff} \{(C_1 \bar{\leq} C_2) \wedge (C_1 \neq C_2)\}$$

Lemme - L111

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

$$\{C_1 \bar{\leq} C_2\} \iff \{\forall i \in I, C_1(i) \bar{\leq} C_2(i)\}$$

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

DEF 109 $\implies C_1 \bar{\leq} C_2 = (C_1 \bar{\cup} C_2) = C_2$
 $\forall i \in I, (C_1 \bar{\cup} C_2)(i) = C_2(i)$

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

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

b) - réciproque.

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

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

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

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

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

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

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

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

Dans cette identité, $C_1(i)$ et $C_2(i)$ sont des applications de I dans V_a , et $V_a \times V_a \xrightarrow{\bar{\Delta}} V_a$, donc $C_1(i) \bar{\vee} C_2(i)$ est toujours défini, à résultat élément de V_a , éventuellement \square . Mais si $v = C_1(i) \bar{\vee} C_2(i) = \square$, alors

$(i, \square) \notin C_1 \bar{\vee} C_2$, ce qui satisfait (DEF 101, a)

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

Lemme - L113

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

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

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

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

2ème cas : Pour i donné, ($\exists v \in (V_a - \{\square\}) \mid (i, v) \in (C_1 \bar{\vee} C_2)$) est faux, entraîne
 $(C_1 \bar{\vee} C_2)(i) = \square$ et $\{\forall v \in (V_a - \{\square\}), (i, v) \notin (C_1 \bar{\vee} C_2)\}$
 donc $\{\exists u \in (V_a - \{\square\}) \mid u = C_1(i) \bar{\vee} C_2(i)\}$, mais $C_1(i) \bar{\vee} C_2(i) \in V_a$ donc
 $C_1(i) \bar{\vee} C_2(i) = \square$, donc à nouveau

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

Lemme - L114

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

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

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

L 113 \Rightarrow

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

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

$$(\text{DEF } 12) \quad x(i) \bar{\cup} (x(i) \bar{\vee} y(i)) = (x(i) \bar{\vee} y(i))$$

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

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

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

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

Lemme - L115

$$\forall (c_1, c_2) \in \mathcal{C}^2 \quad (c_1 \bar{\cup} c_2) \bar{\subseteq} (c_1 \bar{\vee} c_2)$$

$$c_1 \bar{\vee} c_2 = (c_1 \bar{\vee} c_2) \bar{\cup} c_2 \quad (\text{L114})$$

$$= (c_1 \bar{\cup} (c_1 \bar{\vee} c_2)) \bar{\cup} c_2 \quad (\text{L114})$$

$$= (c_2 \bar{\cup} (c_1 \bar{\cup} (c_1 \bar{\vee} c_2))) \quad (\text{L108, 2})$$

$$= (c_2 \bar{\cup} c_1) \bar{\cup} (c_1 \bar{\vee} c_2) \quad (\text{L108, 1})$$

$$= (c_1 \bar{\cup} c_2) \bar{\cup} (c_1 \bar{\vee} c_2) \quad (\text{L108, 2})$$

$$\Rightarrow (c_1 \bar{\cup} c_2) \bar{\subseteq} (c_1 \bar{\vee} c_2), \quad (\text{DEF 109})$$

Remarque : L'ensemble \mathcal{C} , muni des opérations $\bar{\cup}$ et $\bar{\vee}$ 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 $(\mathcal{C}, \bar{\cup}, \bar{\vee})$ les lemmes P1 à P13. En plus, nous utiliserons les propriétés suivantes :

Lemme - L116

$$\forall (c_1, c_2) \in \mathcal{C}^2 \quad \{c_1 = c_2\} \Leftrightarrow \{\forall i \in I, c_1(i) = c_2(i)\}$$

$$\{c_1 = c_2\} \stackrel{(\text{L108, 3})}{\Leftrightarrow} \{(c_1 = c_1 \bar{\cup} c_2) \wedge (c_2 = c_1 \bar{\cup} c_2)\}$$

DEF 109

$$\Leftrightarrow \{(c_1 \bar{\subseteq} c_2) \wedge (c_2 \bar{\subseteq} c_1)\}$$

L111

$$\Leftrightarrow \{\forall i \in I, (c_1(i) \bar{\subseteq} c_2(i)) \wedge (c_2(i) \bar{\subseteq} c_1(i))\}$$

DEF 12

$$\Leftrightarrow \{\forall i \in I, (c_1(i) = c_1(i) \bar{\cup} c_2(i)) \wedge (c_2(i) = c_1(i) \bar{\cup} c_2(i))\}$$

$$\Leftrightarrow \{\forall i \in I, c_1(i) = c_2(i)\}$$

Lemme - L117

$$\{\forall (c_1, \dots, c_n, \dots) \in \mathcal{C}^\infty$$

$$\forall (s_0, \dots, s_n, \dots) \in \mathcal{E}^\infty \mid$$

$$(s_0 = \emptyset$$

$$s_1 = s_0 \bar{\vee} c_1$$

...

$$s_n = s_{n-1} \bar{\vee} c_n \}$$

\Rightarrow {la suite s_0, s_1, \dots, s_n n'est pas infinie et strictement croissante pour $\bar{\leq}$ }

Montrons d'abord la proposition préliminaire 1 suivante :

$$\forall (c_1, c_2) \in \mathcal{C}^2$$

$$\{c_1 \bar{\leq} c_2\} \Rightarrow \{\exists i \in I \mid c_1(i) \bar{<} c_2(i)\}$$

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

$$\Rightarrow \{\forall i \in I, \neg(c_1(i) \bar{\leq} c_2(i) \wedge c_1(i) \neq c_2(i))\}$$

$$\Rightarrow \{\forall i \in I, \neg(c_1(i) \bar{\leq} c_2(i)) \vee (c_1(i) = c_2(i))\} \quad (1)$$

(L 111)

$$\Rightarrow c_1 \bar{<} c_2 \Rightarrow c_1 \bar{\leq} c_2 \Rightarrow \{\forall i \in I, c_1(i) \bar{\leq} c_2(i)\} \quad (2)$$

$$(1), (2) \Rightarrow \{\forall i \in I, (c_1(i) \bar{\leq} c_2(i))$$

$$\wedge [\neg(c_1(i) \bar{\leq} c_2(i)) \vee (c_1(i) = c_2(i))]\}$$

$$\Rightarrow \{\forall i \in I, (c_1(i) \bar{\leq} c_2(i)) \wedge (c_1(i) = c_2(i))\}$$

$$\Rightarrow \{\forall i \in I, c_1(i) = c_2(i)\}$$

(L116)

$$\Rightarrow c_1 = c_2, \text{ contraire à } c_1 \neq c_2.$$

Si maintenant la suite $S_0, S_1, \dots, S_n, \dots$ 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 :

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

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

En fait

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

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

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

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

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

Mais

$$C_{k_n+1}(j) = C_{k_n}(j) \bar{\vee} V_{k_m+1}$$

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

...

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

Donc la séquence infinie est de la forme :

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

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

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

$\forall n \in N$

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

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

Notons $\text{Card}(\alpha)$, la cardinalité d'un ensemble α .

Hypothèse - H 204

- 1) - $\forall n \in N_a, \text{Card}(\text{arc sortie}(n)) = \text{Card}(\text{arc entrée}(n)) = 1$
- 2) - $\forall n \in N_t, \text{Card}(\text{arc sortie}(n)) = 2, \text{Card}(\text{arc entrée}(n)) = 1$
- 3) - $\forall n \in N_j, \text{Card}(\text{arc sortie}(n)) = 1$
- 4) - $\forall n \in N_s, \text{Card}(\text{arc sortie}(n)) = 0, \text{Card}(\text{arc entrée}(n)) = 1$
- 5) - $\text{Card}(\text{arc sortie}(n_e)) = 1, \text{Card}(\text{arc entrée}(n_e)) = 0$

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, \dots, 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

{ $\forall a \in A, \exists (a_1, \dots, a_n) \in \text{chemins}(G) \mid$
 $(a_1 = \text{arc initial}(G) \wedge (a_n = a))$ }

Définition - DEF 208

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

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

Lemme - L 209

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

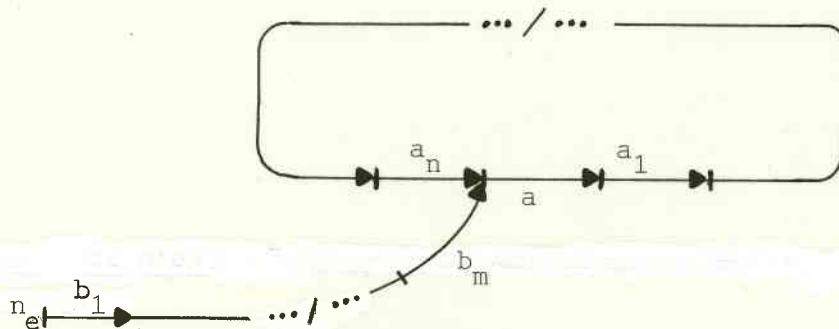
$\Leftrightarrow \{\forall (a_1, \dots, a_n) \in \text{cycles}(G), \exists k \in [1, n] \mid a_k[1] \in N_{j_s} \cup N_{j_b}\}$

- Soit un cycle (a, a_1, \dots, a_n, a) de G . D'après (D 203),
 $a_n \in \text{arc d'entrée}(a[1])$, 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 $(\text{arc initial}(G), b_1, \dots, b_n, a)$ de G .

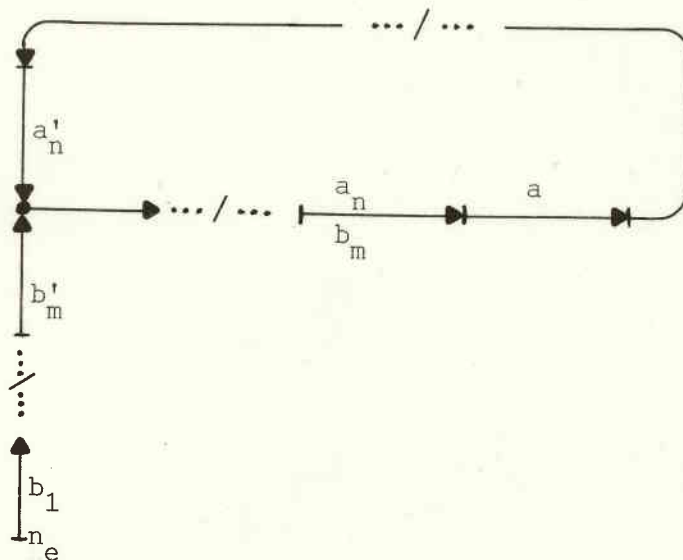
Comme précédemment, $b_n \in \text{arc d'entrée}(a[1])$

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



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

2) - si $a_n = b_m$



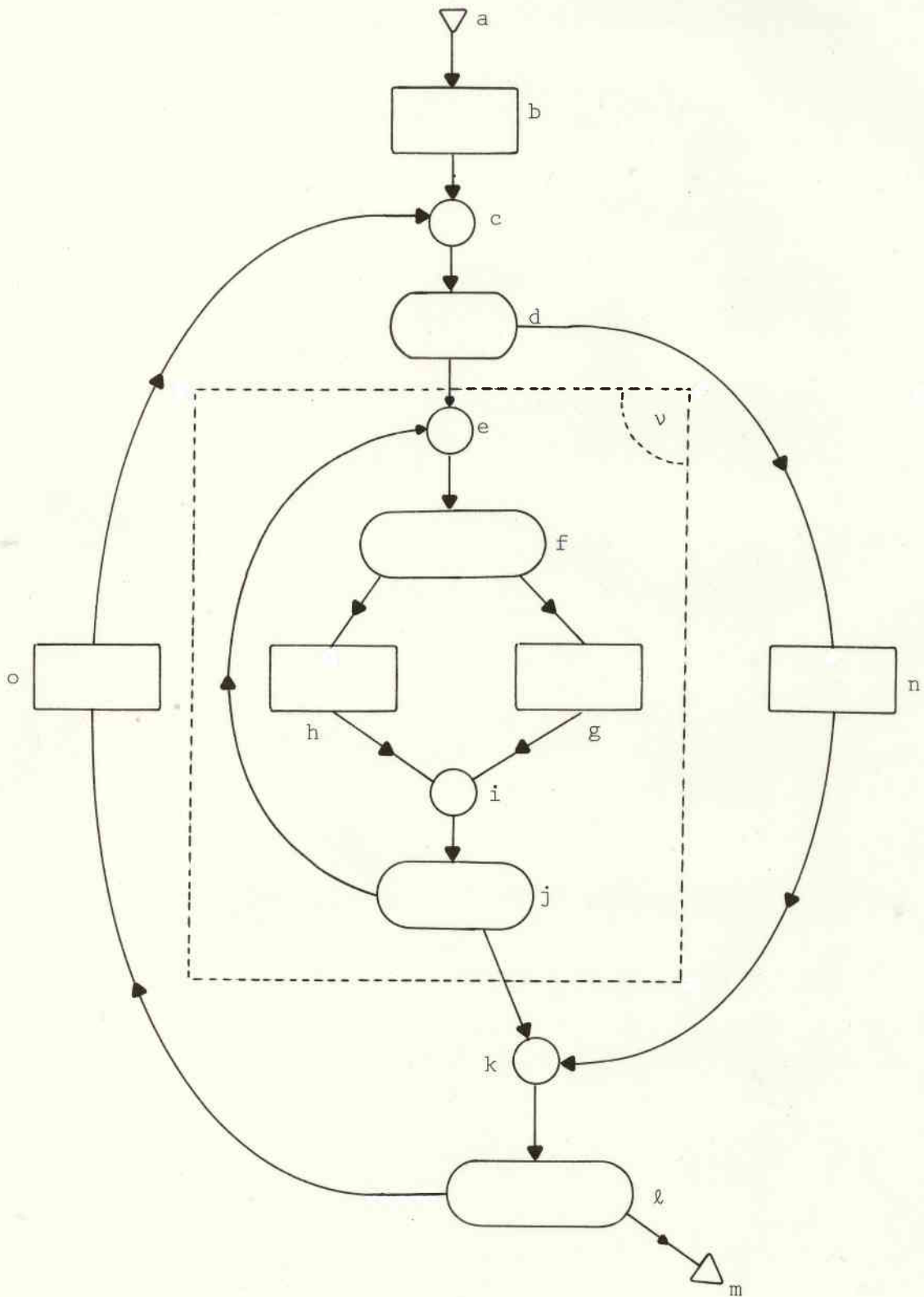
il existe n' et m' uniques, tels que $a_n \neq b_{m'}$ et $a_{n'+1} = b_{m'+1}, \dots, a_n = b_{m'}$ (car $b_1 \neq a, a_1, \dots, a_n$ puisque n_e est le noeud d'entrée). Dans ces conditions, comme précédemment a_n , [2] est nécessairement un noeud de jonction.

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.

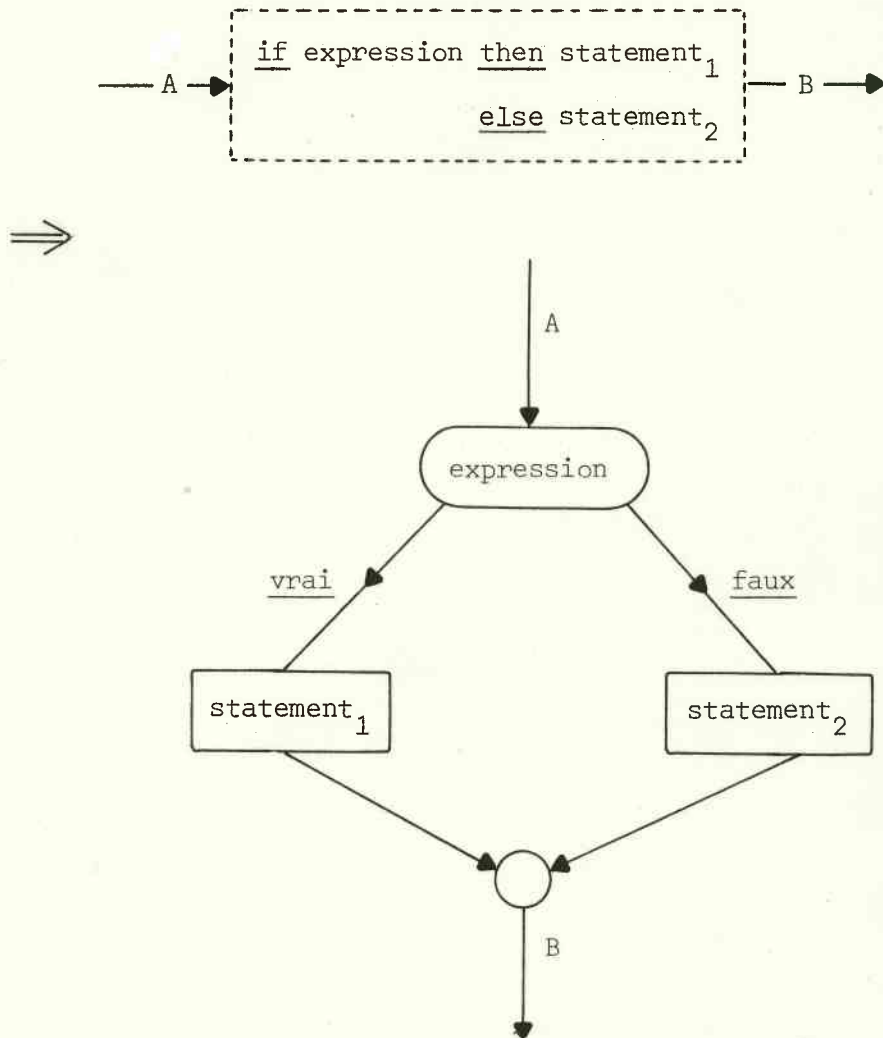
Considérons le graphe de programme suivant :



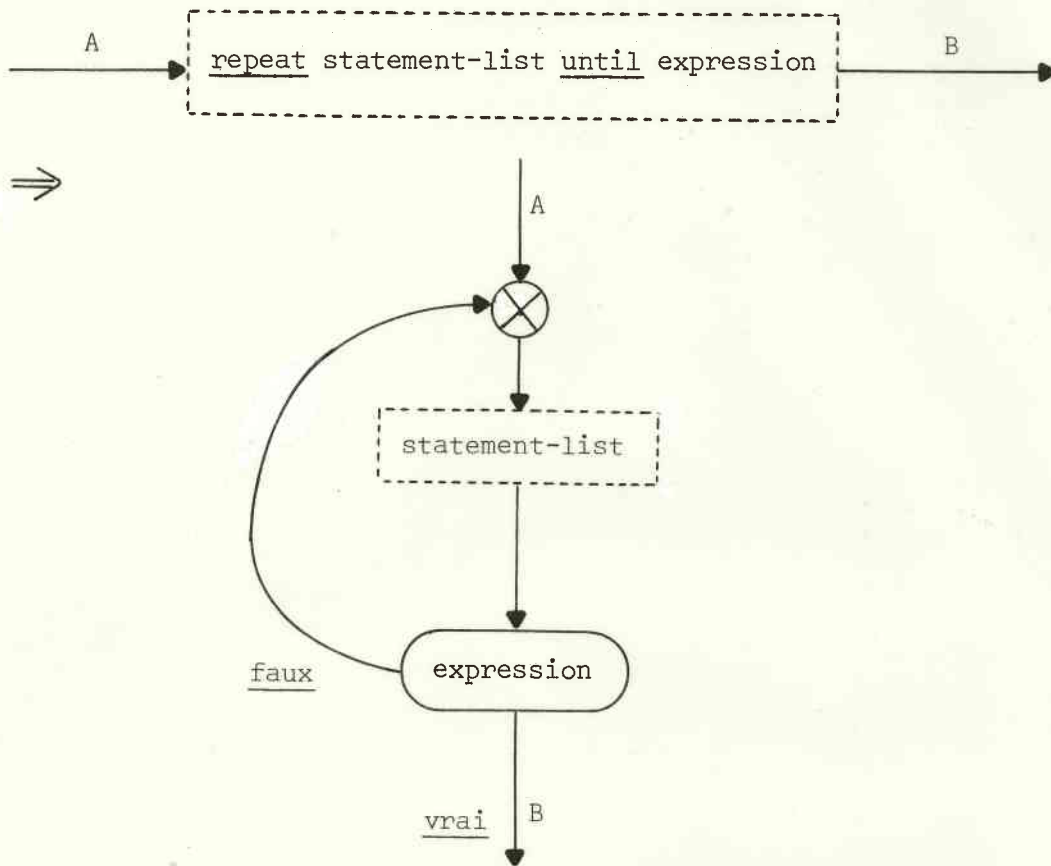
Intuitivement on voit que le noeud de jonction **i** exprime une alternative entre deux calculs **h** ou **g** à la suite du test **f**. Au contraire, le noeud de jonction **e**, correspond à une boucle ou calcul itératif au travers de $e \rightarrow f \rightarrow h \rightarrow i \rightarrow j \rightarrow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

de calcul entre n et le bloc v à la suite du test d, tandis que le noeud c, ferme les boucles c d n k l σ c, ou c d v 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 $X = \{x_1, x_2, \dots, x_n\}$ 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 hypergraphe sur X , si l'on a :

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

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

- E_1, E_2, \dots, E_m sont les arêtes de l'hypergraphe.
- Un ensemble transversal des arêtes d'un hypergraphe $H = (X ; E_1, E_2, \dots, E_m)$ est un ensemble $T \subset X$ avec $T \cap E_i \neq \emptyset$ ($i = 1, 2, \dots, m$).
- 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 $X = \{x_1, x_2, \dots, x_n\}$, on pose :

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

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

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

procédure $\text{Tr } \mathcal{A}$, \mathcal{A} famille de sous-ensembles de X ;

début

procédure $\text{Tr } \{A\}$, A ensemble d'éléments de X ;

début

$\text{Tr}\{A\} := (\{a\} \mid a \in A) ;$

fin ;

$\{A_1, A_2, \dots, A_k\} := \text{Min } \mathcal{A} ;$

$\text{Tr } \mathcal{A} := \text{Tr } \{A_1\} ;$

pour i de 2 à k faire

$\text{Tr } \mathcal{A} := \text{Min } (\text{Tr } \mathcal{A} \vee \text{Tr } \{A_i\}) ;$

refaire ;

fin ;

Exemple :

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

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

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

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

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

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

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

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

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 :

$$\begin{aligned} \Gamma = & \{(c, d, n, k, l, \sigma, c), \\ & (c, d, e, f, h, i, j, k, l, \sigma, c), \\ & (c, d, e, f, g, i, j, k, l, \sigma, c), \\ & (e, f, h, i, j, e), \\ & (e, f, g, i, j, e)\} \end{aligned}$$

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 $\text{Tr } \mathcal{A}$ des ensembles minimaux de la grille de \mathcal{A} .

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

(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 $\text{Tr } \mathcal{A}$).

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

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

Pour faire le choix, diverses heuristiques peuvent être retenues :

- Choisir les ensembles de $\text{Tr } \mathcal{A}$ 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

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

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 :

$$\begin{array}{cccc} \{c, e\}, \{k, e\}, & \text{on choisit } N_{j_b} = \{c, e\} \\ \downarrow \downarrow & \downarrow \downarrow & & \\ \underbrace{2 \quad 4} & \underbrace{7 \quad 4} & & \\ 6 & 11 & & \end{array}$$

- 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 ;

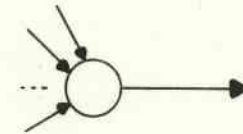


b) - noeuds de sortie N_s ;



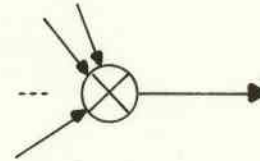
c) - noeuds de jonction simple N_{j_s} ;

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

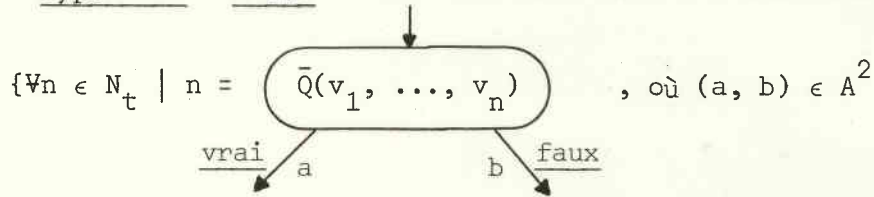
$$\{\forall n \in N_a \mid (n = \boxed{v := \bar{f}(v_1, \dots, v_m)}) \wedge$$

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

- 1) $\{(V_c - \gamma(\square))^m \xrightarrow{\bar{f}} V_c, \text{ où } f \text{ est la fonction dénotée par } \bar{f}\}$
- 2) $\{l'\text{exécution du noeud } n, \text{ ne modifie que la variable } v \text{ (pas d'effets de bord)}\}$

f) - noeuds test ou décisions N_t

Hypothèse - H 214



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

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

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

3) $\{\exists \text{ arc sortie vrai, arc sortie faux} \mid$

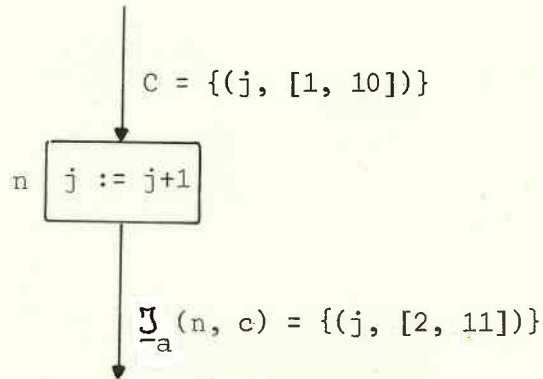
$N_t \xrightarrow{\text{arc sortie vrai}} A ; N_t \xrightarrow{\text{arc sortie faux}} A ; \text{ avec}$

$\underline{\text{arc sortie vrai}}(n) = a \text{ et } \underline{\text{arc sortie faux}}(n) = b\}$

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 $\underline{J}_a(n, c)$ en terme du contexte C de l'arc d'entrée du noeud n. Dans notre exemple introductif, on a vu que :



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

Hypothèse - H 300

- 1) - $\{N_a \times \mathcal{C} \xrightarrow{\underline{J}_a} \mathcal{C}\}$
 $\{\forall n \in N_a \mid n = \boxed{v := \bar{f}(v_1, \dots, v_m)} \rightarrow \}$
- 2) - $\{\forall C \in \mathcal{C}, \forall i \in I, i \neq v, \underline{J}_a(n, C)(i) = C(i)\}$
- 3) - $\{\forall C \in \mathcal{C}, \underline{J}_a(n, C)(v) = @ \left[\begin{array}{l} \{f(v_1, \dots, v_m) \mid \\ (v_1, \dots, v_m) \in \gamma(C(v_1)) \times \dots \times (C(v_m))\} \end{array} \right] \}$

Définition - DEF 304

- $\{\underline{J}_a$ est croissante sur \mathcal{C} pour $\bar{\leq}\}$ $\stackrel{\text{def}}{\iff}$
 $\{(\forall (x, y) \in \mathcal{C}^2) \wedge (\forall n \in N_a) \{x \bar{\leq} y \implies \{\underline{J}_a(n, x) \bar{\leq} \underline{J}_a(n, y)\}\}$

Lemme - L 305

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

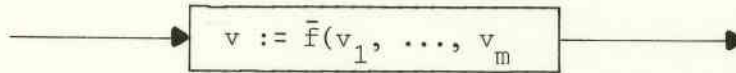
$$\{x \bar{\leq} y\} \Rightarrow \{\underline{J}_a(n, x) \bar{\leq} \underline{J}_a(n, y)\}$$

L111

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

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

Le noeud n est de la forme :



cas 1 - $j \neq v$,

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

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

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

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

et

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

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

L16

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

$$\Rightarrow \{f(v_1, \dots, v_m) \mid (v_1, \dots, v_m) \in \gamma(x(v_1)) \times \dots \times \gamma(x(v_m))\}$$

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

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

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

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

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

$$\stackrel{DEF}{\Rightarrow} @ (V_1) \leq @ (V_2)$$

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

Lemme - L 306

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

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

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

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

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

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

$$\begin{aligned} \underline{J}_a(n, \Phi)(i) &= \Phi(i) \\ &= \{(v, @(\{f()\}))\}(i) = \square \end{aligned}$$

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

$$\underline{J}_a(n, \Phi)(i) = @[\{f(v_1, \dots, v_m) \mid (v_1, \dots, v_m) \in \gamma(\Phi(v_1)) \times \dots \times \gamma(\Phi(v_m))\}]$$

$$(DEF102) \quad = @[\{f(v_1, \dots, v_m) \mid (v_1, \dots, v_m) \in \gamma(\square) \times \dots \times \gamma(\square)\}]$$

2.1) $m \geq 1$

Mais $f(v_1, \dots, v_m)$ n'est pas définie quand un de ses arguments v_k est à valeurs dans $\gamma(\square)$ (voir (H 212-1)). Par conséquent, l'ensemble :

$$\{f(v_1, \dots, v_m) \mid (v_1, \dots, v_m) \in \gamma(\square) \times \dots \times \gamma(\square)\} = \emptyset, \text{ ensemble vide.}$$

Donc :

$$\begin{aligned} \underline{J}_a(n, \Phi)(i) &= @(\emptyset) \\ \text{(L8)} &= \square \\ \text{(DEF 102)} &= \Phi(i) \end{aligned}$$

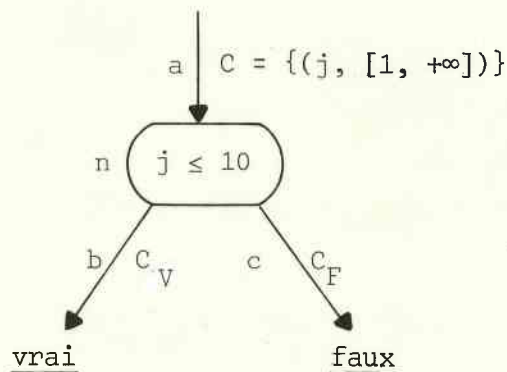
2.2) Si au contraire f , ne dépend d'aucun argument, ($m = 0$), f est constante, toujours définie, donc :

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

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 $(C_V, C_F) = \underline{J}_t(n, c)$.

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



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

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

On peut formaliser les propriétés de \underline{J}_t comme suit :

Hypothèse - H 307

1) - $\{N_t \times \mathcal{C} \xrightarrow{\underline{J}_t} \mathcal{C} \times \mathcal{C}\}$
 $\{(\forall n \in N_t \mid n = \bar{Q}(v_1, \dots, v_m), \text{ où}$

$(v_1, \dots, v_m) \in I^m, \bar{Q} \in B) \wedge$
 $(\forall (C, C_V, C_F) \in \mathcal{C}^3 \mid \underline{J}_t(n, C) = (C_V, C_F))\} \Rightarrow$

2) $(\forall i \in I),$
 $C_V(i) = \mathcal{Q}(\{\tau \mid (\tau \in \gamma(C(i)))$
 $(\exists (v_1, \dots, v_m) \in \gamma(C(v_1)) \times \dots \times \gamma(C(v_m)) \mid$
 $Q(v_1, \dots, v_m))\})$
 $C_F(i) = \mathcal{Q}(\{\tau \mid (\tau \in \gamma(C(i)))$
 $(\exists (v_1, \dots, v_m) \in \gamma(C(v_1)) \times \dots \times \gamma(C(v_m)) \mid$
 $\neg Q(v_1, \dots, v_m))\})$

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

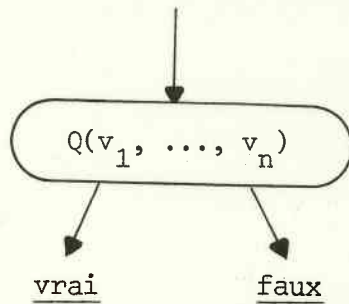
Définition - DEF 311

$\{\underline{J}_t \text{ est croissante sur } \mathcal{C} \text{ pour } \bar{\leq}\} \stackrel{\text{def}}{\iff}$
 $\{\forall (x, y) \in \mathcal{C}^2 \wedge (\forall n \in N_t),$
 $\{x \bar{\leq} y\} \Rightarrow \{(\underline{J}_t(n, x)[1] \bar{\leq} \underline{J}_t(n, y)[1]) \wedge$
 $(\underline{J}_t(n, x)[2] \bar{\leq} \underline{J}_t(n, y)[2])\}$

Lemme - L 312

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

Supposons que le noeud n soit de la forme



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

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

L 111

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

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

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

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

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

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

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

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

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

Lemme - L 313

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

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

On a d'après (H307-2),

$\forall i \in I,$

$$\begin{aligned} \underline{J}_t(n, \Phi) [1] (i) &= @[\{\tau \mid (\tau \in \gamma(\Phi(i))) \wedge \\ &\quad (\exists (v_1, \dots, v_m) \in \gamma(\Phi(v_1) \times \dots \times \Phi(v_m)) \mid \\ &\quad Q(v_1, \dots, v_m))\}] \\ &= @[\{\tau \mid (\tau \in \gamma(\square)) \wedge \\ &\quad (\exists (v_1, \dots, v_m) \in \gamma(\square) \times \dots \times \gamma(\square) \mid Q(v_1, \dots, v_m))\}] \end{aligned}$$

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

$$(\exists (v_1, \dots, v_m) \in \gamma(\square) \times \dots \times \gamma(\square) \mid Q(v_1, \dots, v_m))$$

n'est jamais vérifiée, soit

$$\begin{aligned} \underline{J}_t(n, \Phi) [1] (i) &= @[\{\tau \mid (\tau \in \gamma(\square)) \wedge \underline{\text{faux}}\}] \\ &= @[\emptyset] \\ &= \square && (L8) \\ &= \Phi(i) && (DEF 102) \end{aligned}$$

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

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 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 (Na, Nt, Njs, Njb, Ns, {ne})))
[2] début
[3]   pour chaque arc de A faire contexte local (arc) := ∅ refaire ;
[4]   chemins à exécuter := {arc initial (graphe)} ; jonctions := ∅ ;
[5]   tantque (chemins à exécuter ≠ ∅) faire
[6]     tantque (chemins à exécuter ≠ ∅) 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 ∈ Na →
[12]      |   calcul contexte sortie (arc sortie(noeud traité),
[13]      |   Ja(noeud traité, contexte local(arc))) ;
[14]      | noeud traité ∈ Nt →
[15]      |   (V, F) := Jt(noeud traité, contexte local(arc)) ;
[16]      |   calcul contexte sortie (arc sortie vrai (noeud traité), V) ;
[17]      |   calcul contexte sortie (arc sortie faux (noeud traité), F) ;
[18]      | noeud traité ∈ (Njs ∪ Njb) → jonctions ∪ := {noeud traité} ;
[19]      | noeud traité ∈ Ns → ;
[20]      fincas ;
[21]   refaire ;
[22]   pour chaque noeud de jonctions faire
[23]     contexte sortie := ∪ contexte local (prédécesseur)
[24]     prédécesseur arcs d'entrée (noeud)
[25]     si ¬(contexte sortie ⊆ contexte local (arc sortie(noeud))) alors
[26]       cas
[27]       | noeud ∈ Njs →
[28]       |   calcul contexte sortie (arc sortie (noeud),
[29]       |   contexte sortie) ;
[30]       | noeud ∈ Njb →
[31]       |   calcul contexte sortie (arc sortie(noeud),
[32]       |   contexte local (arc sortie(noeud)) ∩ contexte sortie) ;
[33]       fincas ;
[34]     finsi ;
[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] :

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

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] :

$$\begin{aligned} & \neg(\text{contexte sortie} \stackrel{\bar{=}}{\bar{\leq}} \text{contexte local} (\underline{\text{arc sortie}} (\text{noeud}))) \\ \text{(P13)} \Rightarrow & \neg((\text{contexte local} (\underline{\text{arc sortie}} (\text{noeud}))) \bar{\vee} \text{contexte sortie}) \\ & \stackrel{\bar{=}}{\bar{\leq}} \text{contexte local} (\underline{\text{arc sortie}} (\text{noeud}))) \end{aligned}$$

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 $(C_0, C_1, C_2, \dots, C_m)$ 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 :

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

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] :

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

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

$$C_{m_0+1} = (C_{m_0} \bar{\vee} \text{contexte sortie}) \quad (2)$$

$$(1) \wedge (P6) \implies C_{m_0} \leq (C_{m_0} \bar{v} \text{ contexte sortie})$$

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

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 \bar{J}_a , \bar{J}_t , \bar{U} , \bar{V} et \bar{X} 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 $a = a_0, a_1, \dots, a_m = a$. 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 $a_0, a_1, \dots, a_i, \dots, a_m$, 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" $\neq \emptyset$ }. Comme à la suite de [6] - [20] (qui termine) on a {"chemins à exécuter" = \emptyset }, 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, \dots tels que la séquence CL_0, CL_1, \dots des contextes locaux de a soit de la forme $CL_0 = \emptyset, CL_1 = CL_0 \bar{\vee} CS_1, \dots, CL_n = CL_{n-1} \bar{\vee} CS_n, \dots$ 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 $a = a_0, a_1, \dots, a_n = a$ du graphe. D'après (L210) ce cycle contient au moins un noeud n_{j_b} 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 $(C_{\alpha 0}, C_{\alpha 1}, \dots, C_{\alpha m})$ la suite des "contexte locaux" affectés à un arc α quelconque du graphe, par la procédure d'"interprétation abstraite", alors $(\forall \alpha, \forall m \geq 0), \Phi = C_{\alpha 0} \stackrel{\leftarrow}{\sim} \dots \stackrel{\leftarrow}{\sim} C_{\alpha m}$.

- 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] \sim [35].

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

$$\begin{aligned} \underline{J}(n, C) = & \text{cas} \\ & \left| \begin{array}{l} n \in N_a \rightarrow \underline{J}_a(n, C) ; \\ (n \in N_t) \wedge (\alpha = \text{arc sortie vrai}(n)) \rightarrow \underline{J}_t(n, C) [1] ; \\ (n \in N_t) \wedge (\alpha = \text{arc sortie faux}(n)) \rightarrow \underline{J}_t(n, C) [2] ; \end{array} \right. \\ & \text{fincas} ; \end{aligned}$$