Détermination automatique de relations linéaires vérifiées par les variables d'un programme

Nicolas Halbwachs

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR DE 3ème CYCLE
Informatique

par

Nicolas HALBWACHS

DETERMINATION AUTOMATIQUE DE RELATIONS LINEAIRES VERIFIEES PAR LES VARIABLES D'UN PROGRAMME

Thèse soutenue le 12 mars 1979 devant la commission d'examen

M. SAKAROVITCH	Président
P. COUSOT	Rapporteur
PH. JORRAND M. NIVAT C. PAIR	Examinateurs

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Joseph KLEIN : Vice-Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	AMBLARD Pierre	Clinique de dermatologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	I.S.N.
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale A
	BEAUDOING André	Clinique de pédiatrie et puériculture
	BELORIZKY Elie	Physique
	BARNARD Alain	Mathématiques pures
Mme	BERTRANDIAS Françoise	Mathématiques pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques pures
	BEZES Henri	Clinique chirurgicale et traumatologie
	BLAMBERT Maurice	Mathématiques pures
	BOLLIET Louis	Informatique (I.U.T. B)
	BONNET Jean-Louis	Clinique ophtalmologie
	BONNET-EYMARD Joseph	Clinique hépato-gastro-entérologie
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONVEL Louis	Mathématiques pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie

.../...

MM.	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques pures
	CHARACHON Robert	Clinique ot-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	COUDERC Pierre	Anatomie pathologique
	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DODU Jacques	Mécanique appliquée (I.U.T. I)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	FONTAINE Jean-Marc	Mathématiques pures
	GAGNAIRE Didier	Chimie physique
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques pures
	KOSZUL Jean-Louis	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LE ROY Philippe	Mécanique (I.U.T. I)

.../...

MM.	LLIBOUTRY Louis	Géophysique
	LOISEAUX Jean-Marie	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mlle	LUTZ Elisabeth	Mathématiques pures
MM.	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAYNARD Roger	Physique du solide
	MAZARE Yves	Clinique Médicale A
	MICHEL Robert	Minéralogie et pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEGRE Robert	Mécanique
	NOZIERES Philippe	Spectrométrie physique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques pures.
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET Jean	Séméiologie médicale (neurologie)
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-Chirurgie
	SARRAZIN Roger	Clinique chirurgicale B
	SEIGNEURIN Raymond	Microbiologie et hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique (I.U.T. I)
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique biophysique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale

.../...

PROFESSEURS ASSOCIES

MM.	CRABBE Pierre	CERMO
	SUNIER Jules	Physique

PROFESSEURS SANS CHAIRE

Mlle	AGNIUS-DELORS Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Gilbert	Géographie
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (I.U.T. I)
	BUISSON René	Physique (I.U.T. I)
	BUTEL Jean	Orthopédie
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique (I.U.T. I)
	DELOBEL Claude	M.I.A.G.
	DEPASSEL Roger	Mécanique des fluides
	GAUTRON René	Chimie
	GIDON Paul	Géologie et minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques pures
	JULLIEN Pierre	Mathématiques appliquées
Mme	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques appliquées
	KUHN Gérard	Physique (I.U.T. I)
	LUU DUC Cuong	Chimie organique - pharmacie
	MICHOULIER Jean	Physique (I.U.T. I)
Mme	MINIER Colette	Physique (I.U.T. I)

.../...

MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	PFISTER Jean-Claude	Physique du solide
Mlle	PIERY Yvette	Physiologie animale
MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STIEGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	ARMAND Yves	Chimie (I.U.T. I)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (I.U.T. B) (Personne étrangère habilitée à être directeur de thèse)
	BERNARD Pierre	Gynécologie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	COLIN DE VERDIERE Yves	Mathématiques pures
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROÛZET Guy	Radiologie

.../...

MM.	CYROT Michel	Physique du solide
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GIDON Maurice	Géologie
	GROS Yves	Physique (I.U.T. I)
	GUIGNIER Michel	Thérapeutique
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	JALBERT Pierre	Histologie
	JUNIEN-LAVILLAVROY Claude	O.R.L.
	KOLODIE Lucien	Hématologie
	LE NOC Pierre	Bactériologie-virologie
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MALLION Jean-Michel	Médecine du travail
	MARECHAL Jean	Mécanique (I.U.T. I)
	MARTIN-BOUYER Michel	Chimie (CUS)
	MASSOT Christian	Médecine interne
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (I.U.T. I)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (I.U.T. B) (Personnalité étrangère habilitée à être directeur de thèse)
	PEFFEN René	Métallurgie (I.U.T. I)
	PERRIER Guy	Géophysique-glaciologie
	PHELIP Xavier	Rhumatologie
	RACHALL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
Mme	RENAUDET Jacqueline	Bactériologie (pharmacie)
MM.	ROBERT Jean-Bernard	Chimie-physique
	ROMIER Guy	Mathématiques (I.U.T. B) (Personnalité étrangère habilitée à être directeur de thèse)
	SAKAROVITCH Michel	Mathématiques appliquées

.../...

MM.	SCHAERER René	Cancérologie
Mme	SEIGLE-MURANDI Françoise	Crytogamie
MM.	STOEBNER Pierre	Anatomie pathologie
	STUTZ Pierre	Mécanique
	VROUSOS Constantin	Radiologie

MAITRES DE CONFERENCES ASSOCIES

MM.	DEVINE Roderick	Spectro Physique
	KANEKO Akira	Mathématiques pures
	JOHNSON Thomas	Mathématiques appliquées
	RAY Tuhina	Physique

MAITRE DE CONFERENCES DELEGUE

M.	ROCHAT Jacques	Hygiène et hydrologie (pharmacie)

Fait à Saint Martin d'Hères, novembre 1977

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1977-1978

Président : M. Philippe TRAYNARD

Vice-présidents : M. René PAUTHENET

M. Georges LESPINARD

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Electronique - automatique
	BESSON Jean	Chimie minérale
	BLOCH Daniel	Physique du solide - cristallographie
	BONNETAIN Lucien	Génie chimique
	BONNIER Etienne	Métallurgie
*	BOUDOURIS Georges	Electronique - automatique
	BRISSONNEAU Pierre	Physique du solide - cristallographie
	BUYLE-BODIN Maurice	Electronique - automatique
	COUMES André	Electronique - automatique
	DURAND Francis	Métallurgie
	FELICI Noël	Electronique - automatique
	FOULARD Claude	Electronique - automatique
	LANCIA Roland	Electronique - automatique
	LONGEQUEUE Jean-Pierre	Physique nucléaire corpusculaire
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie - physique
	PAUTHENET René	Electronique - automatique
	PERRET René	Electronique - automatique
	POLOUJADOFF Michel	Electronique - automatique
	TRAYNARD Philippe	Chimie - physique
	VEILLON Gérard	Informatique fondamentale et appliquée

* en congé pour études

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuël	Electronique - automatique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electronique - automatique
	GUYOT Pierre	Métallurgie physique
	LACOUME Jean-Louis	Electronique - automatique
	JOUBERT Jean-Claude	Physique du solide - cristallographie

.../...

MM.	ROBERT André	Chimie appliquée et des matériaux
	ROBERT François	Analyse numérique
	ZADWORNY François	Electronique - automatique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	CHARTIER Germain	Electronique - automatique
	CHIAVERINA Jean	Biologie, biochimie, agronomie
	IVANES Marcel	Electronique - automatique
	LESIEUR Marcel	Mécanique
	MORET Roger	Physique nucléaire - corpusculaire
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie Physique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahim	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique) E.N.S.E.E.G.

MM.	BISCONDI Michel	Ecole des Mines St. Etienne (dépt. Métallurgie)
	BOOS Jean-Yves	Ecole des Mines St. Etienne (Métallurgie)
	DRIVER Julian	Ecole des Mines St. Etienne (Métallurgie)

.../...

MM.	KOBYLANSKI André	Ecole des Mines St. Etienne (Métallurgie)
	LE COZE Jean	Ecole des Mines St. Etienne (Métallurgie)
	LESBATS Pierre	Ecole des Mines St. Etienne (Métallurgie)
	LEVY Jacques	Ecole des Mines St. Etienne (Métallurgie)
	RIEU Jean	Ecole des Mines St. Etienne (Métallurgie)
	SAINFORT	C.E.N. Grenoble (Métallurgie)
	SOUQUET	U.S.M.G.
	CAILLET Marcel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	COULON Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	GUILHOT Bernard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LALAUZE René	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LANCELOT Francis	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SARRAZIN Pierre	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SOUSTELLE Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THEVENOT François	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THOMAS Gérard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TOUZAIN Philippe	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TRAN MINH Canh	Ecole des Mines St. Etienne (Chim. Min. Ph.)

E.N.S.E.R.G.

MM.	BOREL	Centre d'études nucléaires de Grenoble
	KAMARINOS	Centre national recherche scientifique

E.N.S.E.G.P.

M.	BORNARD	Centre national recherche scientifique
Mme	CHERUY	Centre national recherche scientifique
MM.	DAVID	Centre national recherche scientifique
	DESCHIZEAUX	Centre national recherche scientifique

Je tiens à remercier Monsieur M.SAKAROVITCH, Directeur du Laboratoire IMAG d'avoir enrichi cette thèse de ses critiques et d'avoir accepté d'en présider le jury.

Je remercie vivement Messieurs M.NIVAT, Professeur à l'Université de Paris VII, et C.PAIR, Président de l'Institut National Polytechnique de Lorraine, de m'avoir fait l'honneur de participer à ce jury.

Je remercie particulièrement Monsieur Ph.JORRAND, Maître de Recherche au CNRS, qui m'a encouragé durant ce travail, et a accepté de le juger.

Monsieur P.COUSOT, Chargé de recherche au CNRS, m'a accueilli dans son équipe de recherche. Cette thèse est fondée sur ses idées, et n'aurait pu être menée à son terme sans ses critiques fécondes qui m'ont guidé tout au long de sa réalisation. Il m'a fait profiter de sa compétence et de sa culture, et ses remarques au cours de la rédaction, ont considérablement clarifié la présentation, tant au niveau de la forme que du contenu. Pour toutes ces raisons, je lui témoigne ici ma profonde gratitude.

Je remercie également Madame R.COUSOT qui s'est intéressée à ce travail et m'a fait profiter de ses connaissances en programmation linéaire.

Que Mesdames G.BICAIS et H.DIAZ, qui ont effectué un remarquable travail de dactylographie, ainsi que les membres du Service de reprographie, trouvent ici l'expression de ma sincère reconnaissance.

SOMMAIRE

INTRODUCTION

L'importance de l'analyse sémantique des programmes s'accroît avec l'apparition de langages de programmation de plus en plus évolués, et ceci principalement pour deux raisons:

. D'une part les langages modernes permettent au programmeur de spécifier des propriétés sémantiques des objets manipulés, notamment en restreignant le domaine de valeurs des variables par un mécanisme de types. Lorsque le compilateur est incapable de prouver statiquement ces propriétés, il place dans le corps du programme des tests dynamiques destinés à détecter au cours de l'exécution du programme les violations éventuelles des propriétés spécifiées. Or le temps passé à ces tests dynamiques de cohérence représente une part croissante du temps total d'exécution du programme: [Wirth 76a] note que les tests relatifs à la cohérence des accès aux éléments de tableaux peuvent être à l'origine de 20 à 30% du temps d'exécution d'un programme Pascal. L'intérêt présenté par les méthodes de preuves de propriétés à la compilation est donc évident.

. D'autre part, les langages évolués permettent au programmeur de résoudre un problème à un haut niveau d'abstraction, c'est-à-dire sans se préoccuper des caractéristiques élémentaires du calcul sur machine. Si la tâche de programmation s'en trouve plus facile et plus sûre, par contre certaines optimisations des programmes rendues possibles par une programmation de bas niveau, sont maintenant inaccessibles au programmeur. Ceci réduit l'avantage des langages évolués lorsque les contraintes sur les performances des programmes sont fortes (microprocesseurs, programmation système, systèmes temps réel ...). D'où l'intérêt de détecter à la compilation les conditions d'application des techniques d'optimisation.

Ce travail décrit un système d'analyse automatique des programmes capable de détecter les relations linéaires d'égalité et d'inégalité que vérifient les valeurs affectées aux variables en un point d'un programme, quelles que soient les données d'entrée du programme. A titre d'exemple, nous donnons ci-dessous les résultats de l'analyse automatique du programme "FIND" ([Hoare 71]) par notre système. Les relations trouvées figurent entre accolades.

```
spécification d'entrée: 1≤F≤N
D := N ; G := 1 ;   {1≤F≤N, G=1, D=N}
tantque G<D faire   {1≤G≤F≤D≤N, G+1≤D}
   R:= B ⟦F⟧ ;I := G ; J := D ;
   tantque I≤J faire   {1≤G≤F≤D≤N, G≤I≤J≤D, G+1≤D, G+1≤I+J}
       tantque B ⟦I⟧<R faire
           I := I+1 ;  {1≤G≤F≤D≤N, G+1≤I≤N+1, J≤D, G+1≤D}
       fait ;   {1≤G≤F≤D≤N, G+1≤D, G≤I, J≤D, I≤N}
       tantque R<B⟦J⟧ faire
           J := J-1 ;   {1≤G≤F≤D≤N, G+1≤D, G≤I, 0≤J≤D-1}
       fait ;
       si I≤J alors  {1≤G≤F≤D≤N, G≤I≤J≤D, G+1≤D}
           W := B ⟦I⟧ ; B ⟦I⟧ := B ⟦J⟧ ; B ⟦J⟧ := W ;
           I := I+1 ; J := J-1 ;
       finsi ;
   fait ;
   si F≤J alors D := J   {1≤G≤F≤J = D≤N, G+1≤N, G+1≤I+J, J+1≤I}
   sinon
   si I≤F alors G := I   {1≤J+1≤I = G≤F≤D≤N, 2≤D, 2≤I+J}
   sinon stop
   finsi ;
fait ;
```

Les relations trouvées peuvent servir à l'optimisation et à la détection d'erreurs: Dans l'exemple ci-dessus la validité de tous les accès au tableau B, sauf deux, a été prouvée (les doubles crochets représentent les accès prouvés: la notation B ⟦ à la cinquième ligne signifie qu'on a prouvé I≥1 mais pas I≤N). Il en résulte que le compilateur peut économiser 12 tests à l'exécution, dans ce programme.
Le problème de la caractérisation du comportement exact d'un programme étant indécidable, nous utiliserons une analyse sémantique approchée.

Les transformations que nous effectuerons sur les systèmes de relations linéaires font appel à la théorie des polyèdres convexes. Le Chapitre 1 rappelle des élément

de cette théorie, notamment comment un polyèdre convexe peut être caractérisé d'une part comme l'ensemble des solutions d'un système de contraintes linéaires, et d'autre part comme l'enveloppe convexe d'un système générateur fini. Au Chapitre 2 sont définies les principales opérations sur les polyèdres dont nous aurons besoin pour l'analyse des programmes. On étudiera notamment des algorithmes d'enveloppe convexe et de recherche des éléments extrémaux d'un polyèdre.

La méthode d'analyse sémantique approchée que nous proposons est fondée sur la théorie de l'analyse sémantique développée dans [Cousot 76,77a,b,c,78b,79]. Au Chapitre 3, nous caractérisons l'ensemble des propriétés étudiées, en relation avec la sémantique du langage. Au Chapitre 4, est définie l'interprétation d'un ensemble d'instructions simples, c'est-à-dire que si $X=(X_1 \ldots X_n)$ sont les variables numériques du programme, et si P est un polyèdre, pour toute instruction I nous caractérisons un polyèdre Q(I,P) tel que si $X \in P$ avant l'exécution de I, alors $X \in Q(I,P)$ après cette exécution. On caractérise ainsi l'ensemble des polyèdres cherchés comme les plus petites solutions d'un système d'équations récurrentes. La résolution de ce système n'étant pas automatisable, on étudie au Chapitre 5 une méthode de résolution approchée, illustrée par des exemples. Le Chapitre 6 traite de l'analyse sémantique en arrière: alors que l'intepretation en avant décrit ce qui peut se passer au cours d'une exécution du programme, l'analyse en arrière caractérise ce qui doit se passer pour que le programme se termine conformément à ses spécifications de sortie.
L'analyse des programmes utilisant des primitives de haut niveau (structure de bloc, procédure récursives, types abstraits) est considérée au chapitre 7.
Nous parlerons enfin de l'utilisation des résultats de l'analyse (Chapitre 8), de l'implémentation de la méthode (Chapitre 9) et nous situerons ce travail dans le contexte des travaux récents sur l'analyse statique des programmes (Chapitre 10).

PREMIERE PARTIE

ELEMENTS DE GEOMETRIE DES POLYEDRES CONVEXES

1 - DEFINITION ET RESULTATS FONDAMENTAUX SUR LES POLYEDRES CONVEXES

Nous rappelons ici comment un polyèdre convexe peut être caractérisé soit par un système de contraintes linéaires, soit par un système générateur.
[Weyl 50] et [Lanery 66] sont les références générales de ce chapitre.

Notations

On se place dans l'espace affine $\mathbb{R}^n$. Cependant pour la commodité de l'exposé, un point X de cet espace pourra être interprété comme le vecteur $\vec{OX}$ de l'espace vectoriel associé.
Si A est une matrice à m lignes et n colonnes, si I et J sont deux ensembles d'indices, tels que $I \subset \{1..m\}$, $J \subset \{1..n\}$, on désigne par A_I^J la sous-matrice de A formée de tous les éléments situés dans une ligne $i \in I$ et une colonne $j \in J$.
Si X et Y sont deux vecteurs de $\mathbb{R}^n$, $\langle X|Y \rangle$ désignera leur produit scalaire $\Sigma_{i=1}^{n} X_i Y_i$.

1.1. Systèmes de contraintes linéaires

1.1.1. *Définitions*

Soit U un vecteur de $\mathbb{R}^n$ et b un réel.

- L'ensemble $H=\{X \in \mathbb{R}^n | \langle U|X \rangle = b\}$ est un *hyperplan* de $\mathbb{R}^n$. L'intersection de m hyperplans est une *variété linéaire* de $\mathbb{R}^n$.
- L'ensemble $E=\{X \in \mathbb{R}^n | \langle U|X \rangle \leq b\}$ est un *demi-espace fermé* de $\mathbb{R}^n$. L'intersection de m demi-espaces fermés ($m \geq 0$) est un *polyèdre convexe* fermé de $\mathbb{R}^n$.

On désignera par "contraintes linéaires" indifféremment les relations de la forme $\langle U|X \rangle = b$ ou $\langle U|X \rangle \leq b$. Dans un but d'uniformité, une équation sera souvent considérée comme deux inéquations opposées.

1.1.2. Si P_0 est un polyèdre contenant l'origine, la *forme normale* du système de contraintes de P_0 est constituée de trois ensembles finis A1, A2, A3, de vecteurs de $\mathbb{R}^n$ interprétés comme suit:

$$X \in P_0 \iff \begin{cases} \langle U|X\rangle \leq 1 \ , \ \forall U \in A1 \\ \langle U|X\rangle \leq 0 \ , \ \forall U \in A2 \\ \langle U|X\rangle = 0, \ , \ \forall U \in A3 \end{cases}$$

D'autre part, de tout polyèdre P caractérisé par le système de contraintes linéaires $AX \leq B$ (A étant une matrice à m lignes et n colonnes, et B un vecteur de $\mathbb{R}^n$), on sait déduire par translation un polyèdre contenant l'origine: si $X_0 \in P$, alors $AX \leq (B-AX_0)$ est un système de contraintes d'un tel polyèdre.

1.2. Caractérisation extrémale des polyèdres convexes

1.2.1. *Définitions*

Soient $X_1 \ldots X_p$ des vecteurs de $\mathbb{R}^n$ $(p \geq 1)$.

. Un vecteur de la forme $\Sigma_{i=1}^{p} \nu_i X_i$, où les ν_i sont tous des réels, est appelé *combinaison linéaire* des vecteurs X_i. L'ensemble des combinaisons linéaires des vecteurs X_i est le *sous-espace* de $\mathbb{R}^n$ engendré par les X_i. Un sous-ensemble minimal des X_i engendrant ce sous-espace est une *base* du sous-espace.

. Un vecteur de la forme $\Sigma_{i=1}^{p} \mu_i X_i$, où les μ_i sont tous des réels positifs, est appelé *combinaison positive* des vecteurs X_i. L'ensemble des combinaisons positives des X_i est l'*enveloppe conique* des X_i.

. Un vecteur de la forme $\Sigma_{i=1}^{p} \lambda_i X_i$, où les λ_i sont tous des réels compris entre 0 et 1, et tels que $\Sigma_{i=1}^{p} \lambda_i = 1$, est appelé *combinaison convexe* des X_i. L'ensemble des combinaisons convexes des X_i est l'*enveloppe convexe* des X_i.

1.2.2. *Définitions*

. Une partie C de $\mathbb{R}^n$ est dite *convexe* si elle est égale à l'enveloppe convexe de ses points. Un point, un demi-espace, une variété linéaire, une enveloppe conique, l'espace $\mathbb{R}^n$ lui-même, sont des exemples d'ensembles convexes.

Soit C une partie convexe de $\mathbb{R}^n$.

. Un *sommet* de C est un point X de C qui ne peut être obtenu comme combinaison convexe d'autres points de C :

$$\left.\begin{array}{l} X = \Sigma_{i=1}^{p} \lambda_i X_i, \ \Sigma_{i=1}^{p} \lambda_i = 1, p \geq 1 \\ \forall i=1 \ldots p, \lambda_i \in [0,1] \text{ et } X_i \in C \end{array}\right\} \implies \begin{cases} \forall i=1 \ldots p \\ \lambda_i = 0 \text{ ou } X_i = X \end{cases}$$

. Un vecteur r de $\mathbb{R}^n$ est un *rayon* de C si et seulement si il existe une demi-droite de vecteur directeur r, entièrement contenue dans C:

$$\forall \mu \geq 0,\ (X \in C) \Rightarrow (X+\mu r \in C)$$

Deux rayons parallèles et de même sens sont considérés comme égaux.

. Un *rayon extremal* de C est un rayon r de C qui ne peut être obtenu comme combinaison positive d'autres rayons de C:

$$\left.\begin{array}{l} r = \Sigma_{i=1}^{p} \mu_i r_i \text{ et } \forall i=1\ldots p \\ \mu_i \geq 0 \text{ et } r_i \text{ rayons de C} \end{array}\right| \Rightarrow \quad (\forall i=1\ldots p,\ \mu_i=0 \text{ ou } r_i=r)$$

. On dira qu'un vecteur d de $\mathbb{R}^n$ est une *droite* de C si et seulement si d est vecteur directeur d'une droite entièrement contenue dans C (noter l'abus de langage qui nous fait assimiler une droite à son vecteur directeur):

$$\forall \nu \in \mathbb{R},\ (X \in C) \Rightarrow (X+\nu d \in C)$$

Un ensemble convexe qui contient une droite est appelé un *cylindre*. Le plus grand *sous-espace parallèle* à un cylindre est le sous-espace engendré par toutes ses droites. On appelle *section* d'un cylindre C, l'intersection de C avec un sous-espace supplémentaire du plus grand sous-espace parallèle à C (on rappelle que deux sous-espaces E_1 et E_2 de $\mathbb{R}^n$ sont *supplémentaires* si et seulement si leur intersection se réduit au vecteur nul et tout vecteur de $\mathbb{R}^n$ est la somme d'un vecteur de E_1 et d'un vecteur de E_2). Une section ne contient donc pas de droites. Par extension, une section d'un polyèdre ne contenant pas de droite, est égale au polyèdre tout entier.

1.2.3. *Théorème*

. Une condition nécessaire et suffisante pour qu'un ensemble convexe C soit un polyèdre est que toute section de C admette un nombre fini de sommets et de rayons extrémaux.

1.2.4. *Théorème*

Soit P un polyèdre convexe, et P' une section de P. Soient:

- $D=\{d_1\ldots d_\delta\}$ une base du plus grand sous-espace parallèle à P,
- $R=\{r_1\ldots r_\rho\}$ l'ensemble des rayons extrémaux de P',
- $S=\{s_1\ldots s_\sigma\}$ l'ensemble des sommets de P',

Alors tout point de P peut être obtenu comme somme d'une combinaison convexe des s_i, d'une combinaison positive des r_j et d'une combinaison linéaire des d_k:

$$X \in P \iff \begin{cases} \exists \lambda_1 \ldots \lambda_\sigma \in [0,1] \text{ tels que } \Sigma_{i=1}^{\sigma} \lambda_i = 1 \\ \exists \mu_1 \ldots \mu_\rho \in \mathbb{R}^+ , \exists \nu_1 \ldots \nu_\delta \in \mathbb{R} \text{ tels que} \\ \quad X = \Sigma_{i=1}^{\sigma} \lambda_i s_i + \Sigma_{j=1}^{\rho} \mu_j r_j + \Sigma_{k=1}^{\delta} \nu_k d_k \end{cases}$$

Les ensembles S, R et D constituent un *système générateur* de P. Par abus de langage les éléments de S et R seront appelés sommets et rayons extrémaux de P, étant sous-entendu que, si D est non vide, il s'agit des sommets et des rayons extrémaux d'une section de P.

1.2.5. *Exemple*

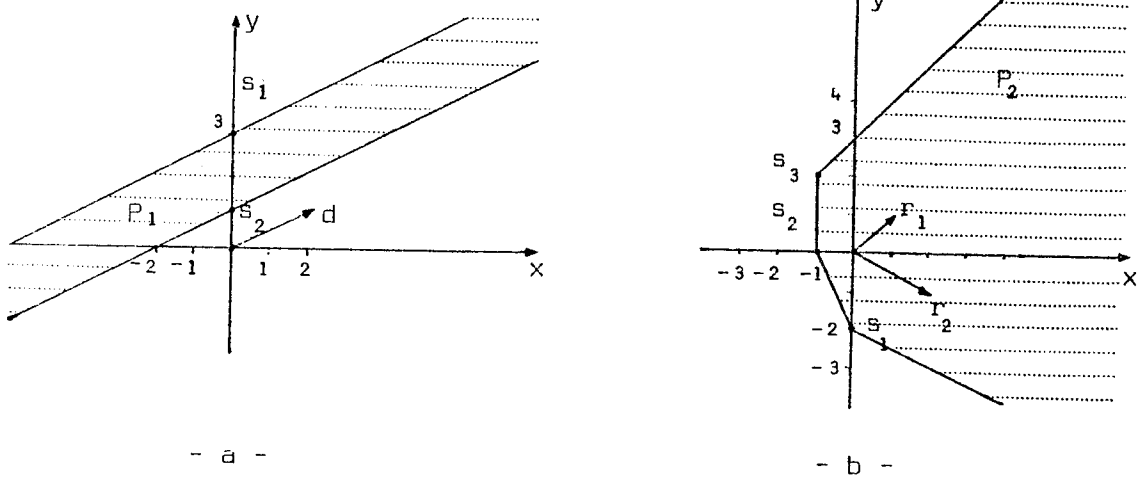
Soient dans $\mathbb{R}^2$ les polyèdres P_1 et P_2 définis respectivement par:

$P_1 = \{(x,y) | -6 \leq x-2y \leq -2\}$

$P_2 = \{(x,y) | x \geq -1, x-y \geq -3, 2x+y \geq -2, x+2y \geq -4\}$

Sous forme matricielle, les systèmes de contraintes s'écrivent:

$$P_1 = \{X \in \mathbb{R}^2 | \begin{bmatrix} -1 & 2 \\ 1 & -2 \end{bmatrix} X \leq \begin{bmatrix} 6 \\ -2 \end{bmatrix}\} \quad , \quad P_2 = \{X \in \mathbb{R}^2 | \begin{bmatrix} -1 & 0 \\ -1 & 1 \\ -2 & -1 \\ -1 & -2 \end{bmatrix} X \leq \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}\}$$



- a -

- b -

figure 1.1.

Le polyèdre P_1 (fig.1.1.a) est un cylindre: le vecteur $d=(2,1)$ est une droite de P_1. Le segment s_1s_2 est une section de P_1 et les points $s_1=(0,1)$ et $s_2=(0,3)$ sont les sommets de cette section. Les sommets s_1, s_2 et la droite d constituent un système générateur de P_1.
Le polyèdre P_2 (fig.1.1.b) possède trois sommets: $s_1=(0,-2)$, $s_2=(-1,0)$, $s_3=(-1,2)$ et deux rayons extrêmes: $r_1=(2,-1)$, $r_2=(1,1)$.

1.3. Correspondance entre les deux représentations

On dispose donc de deux caractérisations duales d'un polyèdre convexe. Nous verrons que certaines opérations sur les polyèdres ne sont possibles que si l'on connait à la fois les deux représentations.

1.3.1. Soit P un polyèdre convexe défini par le système d'inéquations $AX \leq B$. Il résulte de la définition des rayons et des droites que:

- r est un rayon de P si et seulement si $Ar \leq 0$
- d est une droite de P si et seulement si $Ad = 0$

On dit qu'un point X (respectivement: un rayon r) *sature* la ℓ-ième contrainte de P, si et seulement si $A_\ell X = B_\ell$ (resp. $A_\ell r = 0$).

1.3.2. Simplification d'un système de contraintes

1.3.2.1. *Définition*

Une contrainte $A_{\ell_0} X \leq B$ est dite *redondante* dans le système $AX \leq B$ si et seulement si:

$$\{X \mid AX \leq B\} = \{X \mid A_\ell X \leq B_\ell,\ \ell \neq \ell_0\}$$

Nous allons voir comment, connaissant un système générateur d'un polyèdre P, on peut éliminer du système de contraintes $AX \leq B$ de P toutes les inéquations redondantes.

1.3.2.2. Soit $\{1 \ldots m_1\}$ l'ensemble des indices des inéquations qui ne sont pas des équations:

$$\forall \ell = 1 \ldots m_1,\ \exists X \in P \mid A_\ell X < B_\ell$$
$$\forall \ell = m_1+1 \ldots m,\ \forall X \in P \mid A_\ell X = B_\ell$$

La relation $\mathcal{R}$, définie ci-dessous, est un préordre sur l'ensemble $\{1...m_1\}$:
$\forall \ell_1, \ell_2 \in \{1...m_1\}$

$$\ell_1 \mathcal{R} \ell_2 \iff \left\{ \text{Tout sommet ou rayon de P qui sature la contrainte d'indice } \ell_1 \text{ sature aussi la contrainte d'indice } \ell_2. \right.$$

1.3.2.3. *Proposition*

Une inéquation d'indice ℓ_1 est redondante dans le système $AX \leq B$ si et seulement si il existe une autre inéquation, d'indice ℓ_2 ($\ell_1 \neq \ell_2$) telle que $\ell_1 \mathcal{R} \ell_2$.
Si $\ell_1 \mathcal{R} \ell_2$ et $\neg(\ell_2 \mathcal{R} \ell_1)$ la contrainte d'indice ℓ_1 est dite *intrinséquement redondante*, sinon les deux contraintes sont *mutuellement redondantes*. Un système minimal d'inéquations définissant P correspond à un ensemble maximal d'indices non comparables par la relation $\mathcal{R}$.

1.3.2.4. *Remarque*

Extraire un sous-système minimal du système d'équations $\{A_\ell X = B_\ell, \ \ell = m_1+1...m\}$, revient à chercher une base de l'espace vectoriel engendré par les vecteurs A_ℓ, $(\ell = m_1+1...m)$. On peut utiliser pour cela l'étape d'initialisation du simplexe. Cependant, dans les systèmes que nous aurons à manipuler l'apparition d'équations redondantes est beaucoup moins fréquente que celle des inéquations. C'est pourquoi nous avons choisi de ne pas simplifier les systèmes d'équations.

1.3.3. Simplification d'un système générateur

1.3.3.1. *Définitions*

Si S est interprété comme un ensemble de points, R comme un ensemble de rayons, D comme un ensemble de droites, on notera *env*(S,R,D) le plus petit ensemble convexe tel que tout point de S (resp.: tout vecteur de R, de D) soit un point (resp.: un rayon, une droite) de *env*(S,R,D).
Si (S,R,D) est un système générateur d'un polyèdre P, un sommet $s \in S$, un rayon $r \in R$, une droite $d \in D$ seront dits *inutiles* dans ce système générateur, si et seulement si:

$$env(S-\{s\},R,D) = env(S,R-\{r\},D) = env(S,R,D-\{d\}) = env(S,R,D)$$

1.3.3.2. On suppose que les droites ont été distinguées des rayons:

$\forall r \in R, \ \exists \ell = 1...m$ tel que $A_\ell r < 0$

Alors les relations entre éléments de S d'une part, et entre éléments de R d'autre part, que nous notons $\mathcal{R}$ ci-dessous, sont des préodres sur S et R:

$\forall s_1, s_2 \in S$,

$(s_1 \mathcal{R} s_2) \Leftrightarrow$ (toute contrainte saturée par s_1 est aussi saturée par s_2)

$\forall r_1, r_2 \in R$,

$(r_1 \mathcal{R} r_2) \Leftrightarrow$ (toute contrainte saturée par r_1 est aussi saturée par r_2).

1.3.3.3. *Proposition*

Un ensemble maximal de points de S et de rayons de R, incomparables par les relations $\mathcal{R}$, constitue un ensemble minimal de sommets et de rayons extrémaux pour le polyèdre P.

1.3.3.4. Extraire un ensemble minimal de droites de D se ramène encore à la recherche d'une base d'un sous-espace vectoriel (cf. 1.3.2.4.).

1.3.4. Polyèdre polaire

Le paragraphe précédent met en lumière l'étroite correspondance entre les propriétés des systèmes de contraintes et celles des systèmes générateurs. Ceci s'explique par l'existence, pour tout polyèdre P, d'un polyèdre P^+ dont le système générateur se déduit du système de contraintes de P, et dont le système de contraintes se déduit du système générateur de P:

1.3.4.1. *Définition*

On appelle *polyèdre polaire* de P, le polyèdre P^+ défini par:

$$P^+ = \{X \mid \forall Y \in P, \quad \langle X|Y\rangle \leq 1\}$$

1.3.4.2. *Propriétés*

- P^+ contient toujours l'origine
- Si P contient l'origine, alors $P^{++}=P$
- Si (S,R,D) constitue un système générateur de P, alors:

$$X \in P^+ \Leftrightarrow \begin{cases} \forall s \in S, \ \langle s|X\rangle \leq 1 \\ \forall r \in R, \ \langle r|X\rangle \leq 0 \\ \forall d \in D, \ \langle d|X\rangle = 0 \end{cases}$$

. Si P contient l'origine et si (A_1,A_2,A_3) est la forme normale de son système de contraintes (cf.1.1.2.), alors (A_1,A_2,A_3) constitue un système générateur de P^+.

De toute méthode ou propriété s'appliquant à l'une des représentations, on peut donc déduire immédiatement une propriété duale de l'autre représentation, par passage au polyèdre polaire.

1.3.5. Graphe des éléments extrémaux

1.3.5.1. *Définitions*

Soit P un polyèdre, et δ la dimension du plus grand sous-espace parallèle à P.

. Deux sommets de P sont *adjacents*, s'ils saturent au moins $n-\delta-1$ contraintes en commun.

. Un sommet et un rayon extrémal de P sont adjacents s'ils saturent au moins $n-\delta-1$ contraintes en commun.

. Deux rayons extrémaux de P sont adjacents s'ils saturent au moins $n-\delta-2$ contraintes en commun.

1.3.5.2. *Proposition*

Si S est l'ensemble des sommets et R l'ensemble de rayons extrémaux d'un polyèdre P, le graphe de la relation d'adjacence notée A, sur $S \cup R$, est connexe.

2 - OPERATIONS SUR LES POLYEDRES CONVEXES

Nous allons maintenant étudier certaines opérations sur les polyèdres, qui seront nécessaires pour l'analyse des programmes traitée dans la deuxième partie. Afin de décrire aussi précisément que possible l'implémentation de notre analyseur, les algorithmes réalisant ces opérations seront également détaillés et discutés dans ce chapitre.

L'algorithme de projection (§ 2.1) a été publié dans [Kuhn 56], et souvent utilisé depuis (par exemple [King 69], [Ibramsha 78]). Les méthodes de calcul d'une enveloppe convexe (§ 2.2) et de recherche d'un système générateur par intersections successives (§ 2.3.1.) sont, semble-t-il, originales. Le § 2.3.2. décrit brièvement une méthode de recherche d'un système générateur tirée de [Lanery 66].

2.1 - Projection d'un polyèdre [Kuhn 56]

Il sera utile dans la suite d'éliminer une variable d'un système de contraintes, sans perdre d'information sur les autres variables. Par exemple, l'élimination de la variable y du système {x≤y, y≤z} doit fournir la relation {x≤z}. Le résultat de cette opération peut être considéré comme un système de contraintes de la *projection* du polyèdre initial sur un sous-espace $\mathbb{R}^{n-1}$.

2.1.1. - *Définition*

Si P est un polyèdre convexe de $\mathbb{R}^n$, on appellera projection de P selon la i-ième variable, le polyèdre *proj* (P,X_i) défini par :

$$proj\ (P,X_i) = \{(X_1 \ldots X_n) \in \mathbb{R}^n \mid \exists\ z \in \mathbb{R} \text{ tel que } (X_1 \ldots X_{i-1} z X_{i+1} \ldots X_n) \in P\}.$$

2.1.2. - Soit (S,R,D) un système générateur du polyèdre P. Soit d(i) le vecteur de $\mathbb{R}^n$ défini par :

- $d(i)_j = 0 \quad \forall j \neq i$
- $d(i)_i = 1$

Alors (S,R,D ∪ {d(i)}) constitue un système générateur -en général non minimal- du polyèdre $proj(P,X_i)$.

2.1.3. - Soit {AX≤B} un système de contraintes définissant P. Alors un système de contraintes {A'X≤B'} de $proj(P,X_i)$ est défini comme suit :

- $\forall \ell \in 1..m$ tel que $A^i_\ell = 0$, $\exists\ \ell'$ tel que $A'_{\ell'} = A_\ell$ et $B'_{\ell'} = B_\ell$
- $\forall \ell_1, \ell_2 \in 1..m$ tels que $A^i_{\ell_1} > 0$ et $A^i_{\ell_2} < 0$, $\exists \ell'$ tel que
  $A'_{\ell'} = A^i_{\ell_1} . A_{\ell_2} - A^i_{\ell_2} . A_{\ell_1}$ et $B'_{\ell'} = A^i_{\ell_1} . B_{\ell_2} - A^i_{\ell_2} . B_{\ell_1}$
- Toutes les contraintes du système {A'X≤B'} sont obtenues par les règles précédentes.

En d'autres termes on effectue des combinaisons positives de lignes de A, de manière à annuler la i-ième colonne.

Ni le système de contraintes ni le système générateur du polyèdre projection ainsi calculés ne sont minimaux. Ils doivent donc être simplifiés (§ 1.3.2. et 1.3.3.). Nous verrons (§ 2.2.4.) comment effectuer les simplifications au cours de la projection.

2.2 - Enveloppe convexe d'un ensemble fini de points, de rayons et de droites

Etant donnés trois ensembles : $S = \{s_1 \ldots s_\sigma\}$, $R = \{r_1 \ldots r_\rho\}$, $D = \{d_1 \ldots d_\delta\}$ de vecteurs de $\mathbb{R}^n$, on se propose de trouver un système de contraintes linéaires caractérisant le polyèdre convexe P, dont le triplet (S,R,D) est un système générateur.

2.2.1. - On va construire les systèmes de contraintes $(A_{(1)}X \leq B_{(1)})\ldots (A_{(\sigma+\rho+\delta)}X \leq B_{(\sigma+\rho+\delta)})$ d'une suite de polyèdres $P_1 \ldots P_{\sigma+\rho+\delta}$, caractérisés comme suit :

- $P_1 = \{s_1\}$
- $\forall i=2\ldots\sigma,\ P_i = env(P_{i-1}, s_i)$
- $\forall j= \sigma+1\ldots \sigma+\delta,\ P_j = env(P_{j-1}, r_j)$
- $\forall k= \sigma+\rho+1\ldots \sigma+\rho+\delta,\ P_k = env(P_{k-1}, d_k)$

où la fonction *env* associe à un polyèdre P et à un élément e le plus petit polyèdre contenant P et e, conformément à l'interprétation -sommet, rayon ou droite- donnée à e. Le système de contraintes de P_1 est évidemment $\{X=s_1\}$.

2.2.2. - *Proposition*

Soit $(AX \leq B)$ le système de contraintes d'un polyèdre P.

. Soit s un vecteur de $\mathbb{R}^n$, interprété comme un sommet, alors
$X \in env(P,s) \iff \exists \lambda$ tel que $0 \leq \lambda \leq 1$ et $AX+\lambda(AS-B) \leq As$

. Soit r un vecteur de $\mathbb{R}^n$, interprété comme un rayon, alors
$X \in env(P,r) \iff \exists \mu$ tel que $\mu \geq 0$ et $AX-\mu Ar \leq B$

. Soit d un vecteur de $\mathbb{R}^n$, interprété comme une droite, alors
$X \in env\ (P,d) \iff \exists \nu$ tel que $AX-\nu Ad \leq B$

Démonstration : $(X \in env(P,s)) \Rightarrow (\exists \lambda \in [0,1], \exists X_1 \in P$ tels que $X = \lambda X_1+(1-\lambda)s) \Rightarrow (\exists \lambda \in [0,1], \exists\ X_1 \in P$, tels que $AX= \lambda AX_1+(1-\lambda)As)$ $\Rightarrow (\exists \lambda \in [0,1]$ tels que $AX \leq \lambda B+(1-\lambda)As)$. Donc $AX+\lambda(As-B) \leq As$. Inversement, $(\exists \lambda \in [0,1]$ tel que $AX+\lambda(As-B) \leq As) \Rightarrow$ $(A(X-s) \leq 0)$ ou $(\exists \lambda \in]0,1]$ tel que $A((X-(1-\lambda)s)/\lambda) \leq B)$. Or, $(A(X-s)=0) \Rightarrow ((X-s)$ est un rayon de P), donc puisque $s \in env(P,s)$, alors $s+(X-s)= X \in env(P,s)$. D'autre part $(\exists \lambda \in]0,1]$ tel que $A((X-(1-\lambda)s)/\lambda) \leq B) \Rightarrow (X_1=(X-(1-\lambda)s)/\lambda \in P)$. Or $X= \lambda X_1+(1-\lambda)s$ donc $X \in env(P,s)$. Les démonstrations relatives à l'enveloppe d'un polyèdre et d'un rayon et l'enveloppe d'un polyèdre et d'une droite sont analogues. □

Donc, on sait déduire d'un système de contraintes d'un polyèdre P de $\mathbb{R}^n$, un système de contraintes d'un polyèdre de $\mathbb{R}^{n+1}$ dont la projection sur $\mathbb{R}^n$ est égale à *env*(P,e), e étant un sommet un rayon ou une droite. Il suffit dès lors d'éliminer de ce système la variable supplémentaire (λ,μ ou ν), selon la procédure 2.1.3., pour obtenir le système cherché.

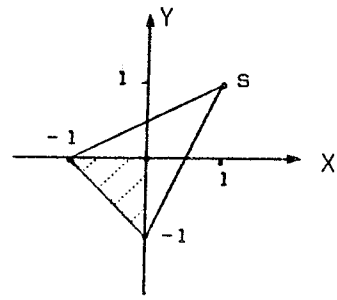
2.2.3. - *Exemple*

Soit P le polyèdre défini dans $\mathbb{R}^2$ par $\{x\leq 0,\ y\leq 0,\ -x-y\leq 1\}$.
Cherchons le système de contraintes du plus petit polyèdre P' contenant P et le point s de coordonnées (1,1).
On a :

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix},\quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\quad As = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix},\quad As-B = \begin{bmatrix} 1 \\ 1 \\ -3 \end{bmatrix}$$

Donc, d'après ce qui précède

$$X \in P' \iff \exists\lambda \text{ tel que } \begin{cases} x + \lambda \leq 1 \\ y + \lambda \leq 1 \\ -x-y-3\lambda \leq -2 \\ -\lambda \leq 0 \\ \lambda \leq 1 \end{cases}$$



L'élimination de λ dans ce système donne :

$$X \in P' \iff \begin{cases} 2x - y \leq 1 \\ -x+2y \leq 1 \\ -x - y \leq 1 \\ x \leq 1 \\ y \leq 1 \end{cases} \quad \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{ contraintes redondantes}$$

2.2.4. - Système de contraintes minimal d'une enveloppe convexe

L'exemple précédent montre que les systèmes de contraintes obtenus ne sont en général pas minimaux. Il est indispensable d'éviter la prolifération de contraintes redondantes. Pour cela, il est possible d'utiliser les résultats du § 1.3.2. sans pour autant calculer à chaque étape la valeur des prédicats $\Sigma(\ell,e)$= "la ℓ-ième contrainte est saturée par l'élément e". L'exposé détaillé de cet algorithme est intéressant pour évaluer le coût du calcul de l'enveloppe. Ce paragraphe peut cependant être omis par le lecteur pressé.

2.2.4.1. - *Définition :* soit P un polyèdre caractérisé d'une part par un système minimal de m contraintes $\{AX \leq B\}$, et d'autre part par un système générateur $(S = \{s_1 \ldots s_\sigma\}$, $R = \{r_1 \ldots r_\rho\}$, $D = \{d_1 \ldots d_\delta\})$. La *matrice de saturation* Σ de P est définie comme suit :

- $\forall c \in \{1 \ldots \sigma\}$, $\Sigma^c = As_c - B$
- $\forall c \in \{\sigma+1 \ldots \sigma+\rho\}$, $\Sigma^c = Ar_{c-\sigma}$

Connaissant la matrice de saturation, d'après les § 1.3.2. et 1.3.3., on sait déterminer les inéquations redondantes, ainsi que les éléments générateurs inutiles de P. On veut donc, à partir de (A,B,S,R,D,Σ) calculer un système minimal de contraintes, ainsi que la matrice de saturation du polyèdre *env*(P,e), où e est interprété soit comme un sommet, soit comme un rayon, soit comme une droite.

2.2.4.2. - *Algorithme*

- Si e est un sommet, on pose V = Ae-B, sinon V=Ae.
- On construit alors deux systèmes matriciels :
  - le système A1, B1, Σ1 formé à partir des contraintes de P qui seront encore des contraintes de *env*(P,e) * :

(*) Si M et N sont deux matrices ayant le même nombre de lignes, on note [M,N] la matrice dont chaque ligne $[M,N]_\ell$ est la concaténation de la ligne M_ℓ et de la ligne N_ℓ.

$\forall \ell \in \{1...m\}$ tel que $V_\ell \leq 0$ ($V_\ell = 0$ si e est une droite), $\exists \ell_1$ tel que

$$A1_{\ell_1} = A_\ell, \quad B1_{\ell_1} = B_\ell, \quad \Sigma 1_{\ell_1} = [\Sigma_\ell, V_\ell]$$

- Le système développé dans $\mathbb{R}^{n+1}$, A2,B2,Σ2, formé des m_2 lignes définies par :

.si e est un sommet, alors $\forall \ell$ tel que $V_\ell \neq 0$, $\exists \ell_2 \leq m_2 - 1$ tel que

$$A2_{\ell_2} = [A_\ell, V_\ell], \quad B2_{\ell_2} = V_\ell + B_\ell, \quad \Sigma 2_{\ell_2} = [\Sigma_\ell, V_\ell]$$

D'autre part $A2^c_{m_2} = 0$ $\forall c = 1..n$, $A2^{n+1}_{m_2} = -1$, $B2_{m_2} = 0$, $\Sigma 2^c_{m_2} = -1$, $\forall c = 1...\sigma$, $\Sigma 2^c_{m_2} = 0$ $\forall c = \sigma+1...\sigma+\rho$, $\Sigma 2^{\sigma+\rho+1}_{m_2} = -1$.

.Si e est un rayon ou une droite, alors $\forall \ell$ tel que $V_\ell \neq 0$, $\exists \ell_2 \leq m_2$ tel que

$$A2_{\ell_2} = [A_\ell, V_\ell], \quad B2_{\ell_2} = B_\ell, \quad \Sigma 2_{\ell_2} = [\Sigma_\ell, V_\ell] .$$

. On effectue alors la projection de {A2,B2} sur $\mathbb{R}^n$ (mise à zéro de la n+1-ième colonne de A2 (§ 2.1.3.)), et chaque fois qu'on effectue une combinaiton positive de lignes sur A2 et B2, on effectue la même combinaison sur les lignes correspondantes de Σ2. Chaque combinaison fournit une ligne U,b,ξ, et on détermine alors, en comparant ξ et avec toutes les lignes de Σ1 si la contrainte $UX \leq b$ est redondante par rapport au système $A1X \leq B1$:

(i) S'il existe un indice ℓ tel que $(\xi^c = 0) \Rightarrow (\Sigma 1^c_\ell = 0)$, $\forall c$, alors la contrainte $UX \leq b$ est ignorée.

(ii) Dans le cas contraire, et s'il existe un indice ℓ tel que $\forall c$, $(\Sigma 1^c_\ell = 0) \Rightarrow (\xi^c = 0)$, on remplace dans A1, B1, Σ1, la ligne d'indice ℓ par U,b,ξ .

(iii) Si aucun des cas précédent n'est vérifié, la ligne U,b,ξ est adjointe au système A1,B1, Σ1.

Lorsque la projection est terminée, le système $A1.X \leq B1$ ne contient pas d'inéquation redondante et Σ1 est la matrice de saturation correspondante.

2.2.4.3. - *Justifications*

Il faut montrer que chaque fois qu'une ligne U,b,ξ est obtenue au cours de la projection, ξ représente bien la ligne de saturation correspondant à la contrainte UX≤b. Nous montrons ceci lorsque e est un sommet, la démonstration des autres cas étant analogue.

Soient ℓ_1 et ℓ_2 les indices des lignes dont la combinaison positive fournit U, b et ξ ($A2^{n+1}_{\ell_1}>0$ et $A2^{n+1}_{\ell_2}<0$).

. Si ℓ_1 et $\ell_2 \in \{1...m_2-1\}$ alors $\exists \ell_1'$ et ℓ_2' tels que

$A2_{\ell_i} = [A_{\ell_i'}, V_{\ell_i'}]$, $B2_{\ell_i} = B_{\ell_i'}+V_{\ell_i'}$, $\Sigma 2_{\ell_i} = [\Sigma_{\ell_i'}, V_{\ell_i'}]$, i=1,2.

On a, pour i=1,2 :

$\forall c \in \{1...\sigma\}, \Sigma^c_{\ell_i'} = A_{\ell_i'} s_c - B_{\ell_i'}$ et $\forall c \in \{\sigma+1... \sigma+\rho\}, \Sigma^c_{\ell_i'} = A_{\ell_i'} r_{c-\sigma}$

$U = V_{\ell_1'} A_{\ell_2'} - V_{\ell_2'} A_{\ell_1'}$, $b = V_{\ell_1'} B_{\ell_2'} - V_{\ell_2'} B_{\ell_1'}$, $\xi = V_{\ell_1'} \Sigma_{\ell_2'} - V_{\ell_2'} \Sigma_{\ell_1'}$

Donc :

$\forall c \in \{1...\sigma\}$, $\xi_c = (V_{\ell_1'} A_{\ell_2'} - V_{\ell_2'} A_{\ell_1'}) s_c - (V_{\ell_1'} B_{\ell_2'} - V_{\ell_2'} B_{\ell_1'}) = Us_c - b$

$\forall c \in \{\sigma+1... \sigma+\rho\}$, $\xi_c = (V_{\ell_1'} A_{\ell_2'} - V_{\ell_2'} A_{\ell_1'}) r_{c-\sigma} = Ur_{c-\sigma}$

. Si $\ell_1 \in \{1...m_2-1\}$ et $\ell_2 = m_2$, $\exists \ell_1'$ comme précédemment et

$U = A_{\ell_1'}$, $b = B_{\ell_1'}+V_{\ell_1'}$, $\xi = V_{\ell_1'} \cdot \Sigma 2_{m_2} + \Sigma_{\ell_1'}$

$\forall c \in \{1...\sigma\}$, $\Sigma 2^c_{m_2} = -1$, donc $\xi = A_{\ell_1'} s_c - (B_{\ell_1'}+V_{\ell_1'}) = Us_c - b$

$\forall c \in \{\sigma+1... \sigma+\rho\}$, $\Sigma 2^c_{m_2} = 0$, donc $\xi = A_{\ell_1'} r_{c-\sigma} = Ur_{c-\sigma}$.

2.2.4.4. - *Exemple*

Reprenons l'exemple 2.2.3. :

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad S = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}, \quad R = D = \emptyset$$

$$\Sigma = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} \quad s = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad As = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} \quad V = As - B = \begin{bmatrix} 1 \\ 1 \\ -3 \end{bmatrix}$$

• s satisfait la 3-ième contrainte ($V_3 \leq 0$) :

$A1 = [-1\ -1]$, $B1 = [1]$ $\Sigma 1 = [-1\ \ 0\ \ 0\ \ -3]$

• Le système développé dans $\mathbb{R}^{n+1}$ s'écrit :

$$A2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & -1 & -3 \\ 0 & 0 & -1 \end{bmatrix} \quad B2 = \begin{bmatrix} 1 \\ 1 \\ -2 \\ 0 \end{bmatrix} \quad \Sigma 2 = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & -3 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

• Projection :

- Combinaison lignes 1 et 3 : $\xi = [-1\ \ -3\ \ 0\ \ 0]$
  cas (iii) $a = [2\ \ -1]$ $b = [1]$

$$A1 = \begin{bmatrix} -1 & -1 \\ 2 & -1 \end{bmatrix} \quad B1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \Sigma 1 = \begin{bmatrix} -1 & 0 & 0 & -3 \\ -1 & -3 & 0 & 0 \end{bmatrix}$$

- Combinaison lignes 1 et 4 : $\xi = [-1\ \ -2\ \ -1\ \ 0]$
  cas (i) : redondance avec la 2-ième ligne de $\Sigma 1$.

- Combinaison lignes 2 et 3 : $\xi = [-1\ \ 0\ \ -3\ \ 0]$
  cas (iii) $a = [-1\ \ 2]$ $b = [1]$

$$A1 = \begin{bmatrix} -1 & -1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \quad B1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \Sigma 1 = \begin{bmatrix} -1 & 0 & 0 & -3 \\ -1 & -3 & 0 & 0 \\ -1 & 0 & -3 & 0 \end{bmatrix}$$

- Combinaison lignes 2 et 4 : $\xi = [-1\ \ -1\ \ -2\ \ 0]$
  cas (i) : redondance avec les 2-ième et 3-ième lignes de $\Sigma 1$.

Le système obtenu est minimal. L'examen de $\Sigma 1$ révèle de plus l'inutilité du sommet s_1, qui ne sature aucune contrainte.

2.2.5. - Enveloppe convexe de deux polyèdres

La méthode de calcul de l'enveloppe convexe d'un ensemble fini d'éléments peut, trivialement, être utilisée pour trouver un système de contraintes de l'enveloppe convexe de deux polyèdres connaissant un système de contraintes de l'un et les deux systèmes générateurs (voir exemple au § 4.2.4.).

2.3 - Recherche d'un Système Générateur

On sait donc trouver le système de contraintes d'un polyèdre connaissant son système générateur. La transformation inverse a fait l'objet de beaucoup d'études, fondées en général sur les méthodes de programmation linéaire [Charnes 53], [Balinski 61], [Lanery 66], [Manas 68], [Matheiss 73],[Dyer 77].

Nous décrirons brièvement (§ 2.3.2.) la méthode de E. Lanery qui est la seule à traiter le problème dans sa généralité, mais auparavant nous esquissons une autre approche, montrant l'équivalence de ce problème avec celui du calcul de l'enveloppe convexe.

2.3.1. - Recherche d'un Système Générateur par Intersections successives

Un polyèdre étant l'intersection de m demi-espaces, si l'on sait trouver d'une part un système générateur d'un demi-espace, d'autre part un système générateur de l'intersection d'un polyèdre et d'un demi-espace connaissant leurs systèmes générateur, on saura alors trouver un système générateur de n'importe quel polyèdre.

2.3.1.1. - *Théorème*

L'enveloppe convexe des polaires de deux polyedres contenant l'origine est la polaire de l'intersection de ces polyèdres :

$$env\ (P_1^+,\ P_2^+) = (P_1 \cap P_2)^+$$

Démonstration : Soient (A_1,A'_1,A''_1) et (A_2,A'_2,A''_2) les formes normales des systèmes de contraintes respectifs de P_1 et P_2. Alors $(A_1 \cup A_2,\ A'_1 \cup A'_2,\ A''_1 \cup A''_2)$ est la forme normale du système de $P_1 \cap P_2$. D'après les propriétés (1.3.4.2.) des polyèdres polaires, P_1^+, P_2^+ et $(P_1 \cap P_2)^+$ admettront donc respectivement (A_1,A'_1,A''_1), $(A_2,\ A'_2,A''_2)$ et $(A_1 \cup A_2,\ A'_1 \cup A'_2,\ A''_1 \cup A''_2)$ comme systèmes générateurs. Ceci exprime que $(P_1 \cap P_2)^+ = env(P_1^+,\ P_2^+)$.

2.3.1.2. - *Proposition*

Si (S,R,D) est un système générateur d'un polyèdre P, et si (S_o,R_o,D_o) est un système générateur de l'intersection de P avec l'hyperplan $H = \{X \mid aX=b\}$ alors :

. Pour tout sommet $s_o \in S_o$,

- soit $s_o \in S$
- soit $\exists s_1$ et $s_2 \in S$ tels que $as_1>b$ et $as_2<b$, et $s_o = \lambda s_1+(1-\lambda)s_2$ avec $\lambda = (b-as_2)/(as_1-as_2)$
- soit $\exists s \in S$ et $r \in R$ tels que $(as-b).(ar)<0$ et $s_o = s+\mu r$ avec $\mu = (b-as)/ar$.
- soit $\exists s \in S$ et $d \in D$ tels que $ad \neq 0$ et $s_o = s+\nu d$, avec $\nu = (b-as)/ad$.

. Pour tout rayon $r_o \in R_o$,

- soit $r_o \in R$
- soit $\exists r_1$ et $r_2 \in R$ tels que $ar_1>0$ et $ar_2<0$ et $r_o = r_1+\mu r_2$ avec $\mu = -ar_1/ar_2$
- soit $\exists r \in R$ et $d \in D$ tels que $ad \neq 0$ et $r_o = r+\nu d$, avec $\nu = -ar/ad$.

. Pour toute droite $d_o \in D_o$

- soit $d_o \in D$
- soit $\exists d_1,d_2 \in D$ telles que $ad_1 \neq 0$ et $ad_2 \neq 0$ et $d_o = d_1+\nu d_2$, avec $\nu = -ad_1/ad_2$.

Principe de la démonstration

Soit X_0 un point quelconque de $P \cap H$. (il est facile de savoir si $P \cap H$ est vide en regardant si tous les sommets s de P vérifient $as<b$ -resp. $as>b$-, si tous les rayons r de P vérifient $ar\leq 0$ -resp. $ar\geq 0$- et si toutes les droites d de P vérifient $ad=0$). On pose $P_0 = \{X-X_0 | X \in P\}$, $H_0 = \{X-X_0 | X \in H\}$. Alors $P_0 \cap H_0$ contient l'origine, et $X \in P_0 \cap H_0 \Rightarrow X+X_0 \in P \cap H$.
P_0 a comme système générateur $(\{ s-X_0 | s \in S\}, R, D)$: $(P_0 \cap H_0)^+$ est égal (2.3.1.1.) à l'enveloppe convexe de P_0^+ et de la droite de vecteur directeur a. Son système développé s'écrit donc :

$$z \in (P_0 \cap H_0)^+ \Leftrightarrow \begin{cases} \exists \nu \in \mathbb{R} \text{ tel que} \\ \forall s \in S, \ \langle s-X_0 | z\rangle - \langle s-X_0 | a\rangle \nu \leq 1 \\ \forall r \in R, \ \langle r | z\rangle - \langle r | a\rangle \leq 0 \\ \forall d \in D, \ \langle d | z\rangle - \langle d | a\rangle = 0 \end{cases}$$

En étudiant la forme des contraintes obtenues en éliminant ν de ce système et en refaisant le changement de variables $X \leftarrow X+X_0$, on obtient les résultats proposés.

2.3.1.3. - *Remarque*

On peut renforcer la proposition (2.3.1.2.), en disant que tout élément extrémal de $P \cap H$ est obtenu par combinaison d'éléments extrémaux adjacents de P. Cependant, le gain d'efficacité que permet cette restriction est annulé par la nécessité de calculer la relation d'adjacence sur P.

2.3.1.4. - Proposition

Soit (S,R,D) un système générateur d'un polyèdre P et soit $E = \{X | aX \leq b\}$ un demi espace de $\mathbb{R}^n$. Alors $(S_0 \cup S_1, R_0 \cup R_1 \cup D_1 \cup D_2, D_0)$ est un système générateur de $P \cap E$, où :

- $S_1 = \{s \in S \mid as<b\}$
- $R_1 = \{r \in R \mid ar<0\}$
- $D_1 = \{d \in D \mid ad<0\}$
- $D_2 = \{d \mid -d \in D \text{ et } ad<0\}$
- (S_o, R_o, D_o) est un système générateur de $P \cap H$, où H est l'hyperplan d'équation $aX=b$.

Les propositions 2.3.1.3. et 2.3.1.4. permettent de calculer un système générateur de l'intersection d'un polyèdre et d'un demi-espace, connaissant un système générateur du polyèdre. Ce processus peut être itéré à partir de $P_o = \mathbb{R}^n$ pour calculer un système générateur de l'intersection P_m de m demi-espaces, c'est-à-dire de n'importe quel polyèdre.

2.3.1.5. - *Exemple*

Cherchons un système générateur du polyèdre P, défini dans $\mathbb{R}^3$ par :

$$X \in P \iff \begin{cases} -X_1 + X_2 - X_3 \leq 0 \\ -X_1 \leq -1 \\ -X_1 - X_2 + X_3 \leq 0 \\ -X_2 + X_3 \leq 3 \end{cases}$$

Soit $P_o = \mathbb{R}^3$:

s_1^0	d_1^0	d_2^0	d_3^0	
0	1	0	0	} système générateur de P_o
0	0	1	0	
0	0	0	1	
0	-1	1	-1	..... saturations par rapport à la première contrainte de P : $-X_1+X_2-X_3 \leq 0$

Soit $P_1 = P_o \cap \{X | -X_1+X_2-X_3 \leq 0\}$. s_1^0 est l'unique sommet de P_1. d_1^0, $-d_2^0$, d_3^0 sont des rayons de P_1. $d_1^0+d_1^0$ et $d_2^0+d_3^0$ sont des droites de P_1 :

s_1^1	r_1^1	r_2^1	r_3^1	d_1^1	d_2^1	
0	1	0	0	1	0	} système générateur de P_1
0	0	-1	0	1	1	
0	0	0	1	0	1	
0	-1	-1	-1	0	0	..... saturations par rapport à la contrainte de P_1
1	-1			-1	0	.... saturations par rapport à la contrainte suivante de P : $-X_1 \leq -1$.

r_2^1 et r_3^1 sont des éléments inutiles.

Soit $P_2 = P_1 \cap \{X | -X_1 \leq -1\}$. r_1 est un rayon de P_2, ainsi que d_1^1 et $r_1^1-d_1^1$. P_2 possède deux sommets, résultants des combinaisons $s_1^1+r_1^1$ et $s_1^1+d_1^1$. d_2^1 est une droite de P_2.

s_1^2	s_2^2	r_1^2	r_2^2	r_3^2	d_3^2	
1	1	1	1	0	0	} système générateur de P_2
0	1	0	1	-1	1	
0	0	0	0	0	1	
-1	0	-1	0	-1	0	} saturations par rapport aux contraintes de P_2
0	0	-1	-1	0	0	
	-2		-2	1	0	..... saturations par rapport à la contrainte suivante de P : $-X_1-X_2+X_3 \leq 0$

s_1^2 et r_1^2 sont inutiles.

Soit $P_3 = P_2 \cap \{X | -X_1-X_2+X_3 \leq 0\}$

P_3 a deux sommets : s_2^2 et $s_2^2+2r_3^2$, deux rayons : r_2^2 et $r_2^2+2r_3^2$ et une droite : d_1^2.

s_1^3	s_2^3	r_1^3	r_2^3	d_1^3	
1	1	1	1	0	} Système générateur de P_3
1	-1	1	-1	1	
0	0	0	0	1	
0	-2	0	-2	0	} Saturations par rapport aux contraintes de P_3
0	0	-1	-1	0	
-2	0	-2	0	0	
-4	-2	-1	1	0	....Saturations par rapport à la contrainte suivante de P : $-X_2+X_3 \leq 3$.

Soit enfin $P_4 = P_3 \cap \{X | -X_2+X_3 \leq 3\} = P$.

P_4 a 4 sommets : s_1^3, s_2^3, $s_1^3+4r_2^3$, $s_2^3+2r_2^3$ deux rayons : r_1^3 et $r_1^3+r_2^3$, et une droite : d_1^3.

s_1^4	s_2^4	s_3^4	s_4^4	r_1^4	r_2^4	d_1^4	
1	1	5	3	1	2	0	} Système générateur de P
1	-1	-3	-3	1	0	1	
0	0	0	0	0	0	1	
0	-2	-8	-6	0	-2	0	} Matrice de saturation de P.
0	0	-4	-2	-1	-2	0	
-2	0	-2	0	-2	-2	0	
-4	-2	0	0	-1	0	0	

$s_3^4 R s_4^4$ donc s_4^4 est inutile.

Le système générateur trouvé est le suivant :

S = {(1 1 0), (1 -1 0), (3 -3 0)}

R = {(1 1 0), (2 0 0)}

D = {(0 1 1)}

2.3.2. - Recherche d'un système générateur par la méthode du pivot

Nous indiquons maintenant très brièvement les principes de la recherche d'un système générateur d'un polyèdre convexe donné par son système de contraintes, par application de la méthode du pivot. Pour plus de détails, nous renvoyons à [Lanery 66]. Les éléments de programmation linéaire ([Sakarovitch 73], Simmonard 73]) sont supposés connus : nous n'en rappelons que l'essentiel.

2.3.2.1. - *Rappels*

Un système de m inéquations linéaires dans $\mathbb{R}^n$ est mis sous forme de système d'équations dans $\mathbb{R}^{m+n}$ par adjonction de m *variables d'écart* $\{X_{n+1} \ldots X_{n+m}\}$ astreintes à rester positives. Un système AX=B est *sous forme canonique par rapport à la base* I, si I est un ensemble de m indices et si les colonnes de A^I forment une permutation de la matrice unité. Pour tout $i \in I$, on définit alors $\sigma(i)$ comme le seul indice de ligne tel que $A^i_{\sigma(i)}=1$. La base I est dite *réalisable* si pour toute variable d'écart X_i en base ($i \in I$), on a $B_{\sigma(i)} \geq 0$. La première phase de la méthode du simplexe permet de trouver une base réalisable si le polyèdre est non vide. L'opération de *pivotage* autour d'un élément $A^i_j \neq 0$ permet alors de passer de la forme canonique par rapport à la base I, à la forme canonique par rapport à la base $I-\{\sigma^{-1}(j)\} \cup \{i\}$.

2.3.2.2. - Algorithme de recherche du système générateur

- Le système de contraintes de P est transformé en système d'équations et mis sous forme canonique par rapport à une base réalisable, s'il en existe (sinon P est le polyèdre vide).

- Pour toute variable initiale (non variable d'écart) X_i hors base, on effectue, si c'est possible un pivotage faisant entrer X_i en base, à la place d'une variable d'écart. S'il est impossible de faire ainsi entrer toutes les variables initiales en base, le polyèdre P est un cylindre.

. Dans ce cas, $\forall i \notin I$, $i \leq n$ (X_i variable initiale), on définit le vecteur $d(i) \in \mathbb{R}^n$ par :

$$d(i)_j = 0 \quad \forall j \leq n,\ j \notin I,\ j \neq i$$
$$d(i)_j = -A^i_{\sigma(j)} \quad \forall j \leq n,\ j \in I$$
$$d(i)_i = 1$$

Alors $D = \{d(i) \mid i \leq n,\ i \notin I\}$ est une base du plus grand sous espace parallèle à P ([Lanery 66], Théorèmes 3 & 5).
En adjoignant au système de contraintes de P le système d'équations $\{\langle d(i) \mid X \rangle = 0 \mid \forall i \leq n,\ i \notin I\}$, on définit une section de P qui ne contient pas de droite. Il est alors possible de mettre ce système sous forme canonique par rapport à une base réalisable contenant toutes les variables initiales.

. Soit AX=B la forme canonique par rapport à une base I réalisable et contenant toutes les variables initiales. Alors

- le vecteur s défini par : $\forall j \leq n$, $s_j = B_{\sigma(j)}$ est un sommet du polyèdre.
- $\forall i > n$, $i \notin I$, si $\forall k > n$, $k \in I$ on a $A^i_{\sigma(k)} \leq 0$ alors le vecteur r(i) défini par $r(i)_j = -A^i_{\sigma(j)}$, $\forall j \leq n$ est un rayon extrémal du polyèdre.
- Enfin, $\forall i > n$, $i \in I$, n'ayant pas permis la découverte d'un rayon, il existe dans la colonne i des pivots susceptibles de conduire à de nouvelles bases réalisables contenant toutes les variables initiales.

Le graphe des sommets étant connexe (1.3.5.2.) il suffit d'effectuer tous ces pivotages, en prenant garde de ne pas réexplorer une base déjà trouvée, pour calculer de proche en proche tous les sommets et tous les rayons du polyèdre.
Un bon algorithme de parcours exhaustif des bases est décrit dans [Dyer 77].

2.3.2.3. - *Exemple*

Reprenons l'exemple 2.3.1.5.

$$[S1].\ X \in P \iff \begin{cases} -X_1 + X_2 - X_3 \leq 0 \\ -X_1 \leq -1 \\ -X_1 - X_2 + X_3 \leq 0 \\ -X_2 + X_3 \leq 3 \end{cases}$$

Le système d'équations correspondant s'écrit, sous forme matricielle :

[S2]

X_1	X_2	X_3	X_4	X_5	X_6	X_7	
-1	1	-1	1	0	0	0	0
-1	0	0	0	1	0	0	-1
-1	-1	1	0	0	1	0	0
0	-1	1	0	0	0	1	3

[S2] est sous forme canonique par rapport à la base irréalisable {4,5,6,7}. L'étape d'initialisation du simplexe fournit le système [S3] sous forme canonique par rapport à la base réalisable {4,1,6,7} :

[S3]

X_1	X_2	X_3	X_4	X_5	X_6	X_7	
0	1	-1	1	-1	0	0	1
1	0	0	0	-1	0	0	1
0	-1	1	0	-1	1	0	1
0	-1	1	0	0	0	1	3

On fait alors rentrer le plus possible de variables initiales en base, ce qui donne la base {2,1,6,7} :

[S4]

X_1	X_2	X_3	X_4	X_5	X_6	X_7	
0	1	-1	1	-1	0	0	1
1	0	0	0	-1	0	0	1
0	0	0	1	-2	1	0	2
0	0	0	1	-1	0	1	4

La variable initiale X_3 reste hors base, et ne peut y entrer sans faire sortir X_2. Donc le vecteur $d = (-A^3_{\sigma(1)}, -A^3_{\sigma(2)}, 1) = (0, 1, 1)$ est une droite de P. Une section P' de P est définie par :
$P' = \{X \in P | \langle d | X \rangle = 0\} = \{X | X \in P \wedge X_2 + X_3 = 0\}$. Ce nouveau système est alors mis sous forme canonique par rapport à la base {2,3,1,6,7}, à partir de laquelle on effectue le parcours du graphe:

[S5]

X_1	X_2	X_3	X_4	X_5	X_6	X_7	
0	1	0	1/2	-1/2	0	0	1/2
0	0	1	-1/2	1/2	0	0	-1/2
1	0	0	0	-1	0	0	1
0	0	0	1	-2	1	0	2
0	0	0	1	-1	0	1	4

Base : {2,3,1,6,7}. Sommet : $s_1 = (B_{\sigma(1)}, B_{\sigma(2)}, B_{\sigma(3)}) = (1, 1/2, -1/2)$
Rayon : $(-A^5_{\sigma(1)}, -A^5_{\sigma(2)}, -A^5_{\sigma(3)}) = (1, 1/2, -1/2)$
Base réalisable adjacente : {2,3,1,4,7}

[S6]

X_1	X_2	X_3	X_4	X_5	X_6	X_7	
0	1	0	0	1/2	-1/2	0	-1/2
0	0	1	0	-1/2	1/2	0	1/2
1	0	0	0	-1	0	0	1
0	0	0	1	-2	1	0	2
0	0	0	0	1	-1	1	2

Base : {2,3,1,4,7}. Sommet : $s_2 = (1, -1/2, 1/2)$. Pas de rayon.
Bases réalisables adjacentes : {2,3,1,6,7} déjà trouvée et {2,3,1,4,5}.

[S8]

X_1	X_2	X_3	X_4	X_5	X_6	X_7	
0	1	0	0	0	0	-1/2	-3/2
0	0	1	0	0	0	-1/2	3/2
1	0	0	0	0	-1	1	3
0	0	0	1	0	-1	2	6
0	0	0	0	1	-1	1	2

Base : {2,3,1,4,5}. Sommet : s_3 = (3, -3/2, 3/2)
Rayon : $r_2 = (-A^6_{\sigma(1)}, -A^6_{\sigma(2)}, -A^6_{\sigma(3)}) = (1, 0, 0)$
Base réalisable adjacente : {2,3,1,4,7} déjà trouvée.

Toutes les bases réalisables contenant les variables initiales ont été trouvées. L'algorithme s'arrête avec le système générateur suivant pour P :

$$S = \{(1, 1/2, -1/2), (1, -1/2, 1/2), (3, -3/2, 3/2)\},$$
$$R = \{(1, 1/2, -1/2), (1, 0, 0)\}, \quad D = \{(0, 1, 1)\}$$

Les résultats sont différents de ceux trouvés par l'autre méthode parce que la section choisie n'est pas la même. On vérifie toutefois que pour tout sommet s, ou tout rayon r trouvés par une méthode il existe un sommet s' ou un rayon r' dans l'autre système générateur, et un réel ν, tels que $s = s' + \nu d$ ou $r = r' + \nu d$, avec d = (0,1,1).

2.3.3. - Evaluation et comparaison des deux méthodes

Sur des systèmes de contraintes de taille réduite, tels que ceux que nous considèrerons, les deux méthodes ont des performances comparables. Le tableau ci-contre donne les temps de calcul des systèmes générateurs de deux familles de polyèdres, les hypercubes K(n) et les cones C(n), définis comme suit :

$$\forall n>0,\ K(n) = \{X \in \mathbf{R}^n \mid -1 \le X_1 \le 1,\ \forall i=1...n\}$$
$$C(n) = \{X \in \mathbf{R}^n \mid -X_1 \le X_i \le X_1,\ \forall i=2...n\}$$

Tableau comparatif des temps d'exécution des deux méthodes

Polyèdre étudié	Dimension de l'espace	Nombre de contraintes	Nombre de sommets	Nombre de rayons	Méthode des intersections successives σ=	Méthode du pivot : π=	Rapport des deux méthodes π/σ
K(2)	2	4	4	0	0,0195 s.	0,0449 s.	2,3
K(3)	3	6	8	0	0,0742 s.	0,1641 s.	2,21
K(4)	4	8	16	0	0,2734 s.	0,5352 s.	1,95
K(5)	5	10	32	0	0,9883 s.	2,0254 s.	2,04
C(2)	2	2	1	2	0,0078 s.	0,0117 s.	1,5
C(3)	3	4	1	4	0,0273 s.	0,0332 s.	1,21
C(4)	4	6	1	8	0,0898 s.	0,1230 s.	1,37
C(5)	5	8	1	16	0,2773 s.	0,3516 s.	1,27
C(6)	6	10	1	32	0,9473 s.	1,1934 s.	1,26

Le choix de la méthode dépendra des informations que l'on possède au préalable sur le polyèdre à étudier. Deux cas se présenteront :

. On cherche un système générateur du polyèdre obtenu par adjonction d'une ou plusieurs contraintes à un polyèdre dont on connait les deux représentations. La méthode de calcul par intersections successives permet de tirer parti de cette connaissance préalable.

. Si par contre, on enlève des contraintes à un polyèdre connu, on aura avantage pour calculer le système générateur du polyèdre résultant, à utiliser la méthode du pivot. Dans ce cas, la phase d'initialisation peut être simplifiée puisqu'on connait des points appartenant au polyèdre.

En conclusion, nous noterons que ces méthodes de conversion d'une représentation dans l'autre, sont très coûteuses. Il nous faudra limiter leur emploi au minimum pour obtenir des performances acceptables.

On aurait pu donner les deux méthodes correspondantes pour le calcul de l'enveloppe convexe. La méthode du pivot est semblable à celle de [Chand 70]. En réalité nous n'aurons à calculer que l'enveloppe convexe de deux polyèdres connaissant chacune de leurs représentations, c'est pourquoi nous n'avons exposé au § 2.2 que la méthode par enveloppes successives (correspondant à une recherche par intersections successives sur les polyèdres polaires).

DEUXIEME PARTIE

METHODE D'ANALYSE AUTOMATIQUE DES RELATIONS LINEAIRES VERIFIEES PAR LES VARIABLES D'UN PROGRAMME

Avertissement:

La méthode d'analyse statique des programmes que nous proposons dans cette deuxième partie, est fondée sur la théorie de l'analyse sémantique des programmes développée par P.& R.COUSOT[Cousot 76,77a,77b,77c,78b,79]. Nous ne rappellerons de cette théorie que les éléments indispensables à la compréhension de l'exposé.

3. CONTEXTES ABSTRAITS

3.1. Programmes

3.1.1. Pour la commodité de l'exposé, un programme sera considéré comme un graphe connexe, dont les noeuds représentent les instructions élémentaires du programme, et les arcs les passages possibles du contrôle d'une instruction à l'autre. Dans un premier temps on se limite à des graphes formés de cinq types de noeuds (le cas des langages plus évolués sera considéré au chapitre 7): Noeud d'*entrée* (tout graphe de programme en possède un et un seul), noeuds de *sortie*, d'*affectation*, de *test* et de *jonction* (correspondant à la jonction de plusieurs chemins d'exécution).
A un noeud d'affectation est attachée l'information relative à la variable but et à l'expression affectée, à un noeud de test est attachée la condition sur laquelle porte le test, et aucune information n'est attachée aux noeuds d'entrée, de sortie et de jonction.

3.1.2. *Exemple*
La figure 3.1. représente le graphe correspondant au fragment de programme suivant, écrit dans un langage de type ALGOL :

```
I:=0 ; J:=0 ;
tantque I≤100 faire
   si A[I]≤A[J] alors I:=I+2 ; J:=J+1
   sinon I:=I+4
   finsi;
fait;
```

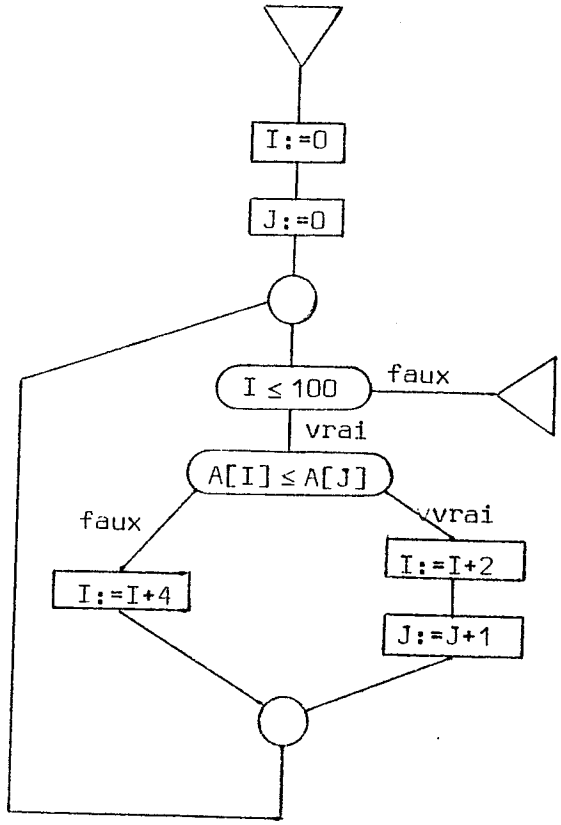


figure 3.1.

3.2. Contextes associés aux arcs

3.2.1. Etats du calcul

Soient $v_1 \ldots v_p$ les variables du programme, déclarées respectivement de types $T_1 \ldots T_p$. Soient $a_1 \ldots a_\alpha$ les arcs du programme. Alors un état du calcul du programme est caractérisé par un vecteur de valeurs $(X_1 \ldots X_p) \in T_1 x \ldots x T_p$ affecté aux variables $v_1 \ldots v_p$, et par un arc $a_i (1 \leq i \leq \alpha)$.

La sémantique du programme définit une relation partielle de succession σ sur l'ensemble des états du calcul, telle que $<(X_1 \ldots X_p), a_i> \sigma <(X'_1 \ldots X'_p), a_j>$ si et seulement si:

- le noeud extrémité de l'arc a_i est l'origine de l'arc a_j
- si $v_1=X_1 \ldots v_p=X_p$ alors en effectuant l'instruction représentée par ce noeud on obtient $v_1=X'_1 \ldots v_p=X'_p$.

Le langage considéré étant déterministe tout état a au plus un état successeur.

On convient de noter σ^* la fermeture transitive réflexive de la relation de succession.

3.2.2. Analyse sémantique des programmes

Soient ϕ et ψ des ensembles de vecteurs de $T=T_1 x \ldots x T_p$. Si a_1 est l'arc issu du noeud d'entrée du programme π, alors l'*analyse sémantique en avant* de π à partir de ϕ consiste à caractériser l'ensemble des états $<Y,a_i>$ tels que:

$\exists X \in \phi$ tel que $<X,a_1> \sigma^* <Y,a_i>$

Soit $E(\pi,\phi)$ cet ensemble.

A chaque arc a_i du programme correspond alors un *contexte concret en avant* $Q(a_i,\phi) = \{Y \mid <Y,a_i> \in E(\pi,\phi)\}$.

Inversement si $a_{k_1}, a_{k_2} \ldots a_{k_s}$ sont les arcs incidents aux noeuds de sortie de π, l'*analyse sémantique en arrière* de π à partir de ψ caractérise l'ensemble $S(\pi,\psi)$ des états $<Z,a_i>$ tels que:

$\exists X \in \psi$, $\exists j \in \{1 \ldots s\}$ tels que $<Z,a_i> \sigma^* <X,a_{k_j}>$

Le *contexte concret en arrière* associé à l'arc a_i est alors

$Q'(a_i,\psi) = \{Z \mid <Z,a_i> \in S(\pi,\psi)\}$

3.2.3. Analyse sémantique approchée

Le calcul des contextes concrets n'est pas automatisable dès que le programme contient des boucles et que l'ensemble des états de calcul est infini.[Cousot 78b] propose de calculer un contexte abstrait, approximation supérieure du contexte concret. Le choix de cette approximation caractérise l'analyse sémantique approchée que l'on veut effectuer. Notre but étant d'analyser les relations linéaires entre les variables numériques, il nous faut faire correspondre à tout prédicat Q sur les variables du programme ($Q \in \mathcal{P}(T_1 x \ldots x T_p)$) un système de contraintes P sur les

variables numériques, tel que si $X \in Q$, alors le vecteur X', formé des composantes numériques de X, vérifie P. Cette correspondance sera réalisée par les fonctions d'abstraction et de concrétisation.

3.2.3.1. *Proposition*

L'ensemble $P(T)$ est un treillis complet pour l'inclusion:
La partie vide et la partie $T_1 x \ldots x T_p$ en sont respectivement le minimum et le maximum, et si $\{P_1, P_2 .. P_k \ldots\}$ est un ensemble, éventuellement infini, de parties de T, alors l'intersection $\bigcap_{k \geq 0} \{P_k\}$ et l'union $\bigcup_{k \geq 0} \{P_k\}$ de ces parties sont encore des parties de T.

3.2.3.2. *Définitions*

Supposons que $T_1 \ldots T_n$ soient des types numériques (entiers, réels, intervalles.. et que $T_{n+1} \ldots T_p$ soient des types non numériques (chaîne, pointeur, structure, tableau...). On note $C(\mathbb{R}^n)$ l'ensemble des parties convexes de $\mathbb{R}^n$. Alors, conformément à [Cousot 78b], on définit les *fonctions d'abstraction* α et de *concrétisation* γ, comme suit:

- $\alpha : P(T) \rightarrow C(\mathbb{R}^n)$
  $\alpha = env(\{(X_1 \ldots X_n) \mid \exists (X_{n+1} \ldots X_p)$ tels que $(X_1 \ldots X_p) \in Q\})$
- $\gamma : C(\mathbb{R}^n) \rightarrow P(T)$
  $\gamma = \lambda C.(\{(X_1 \ldots X_p) \in T \mid (X_1 \ldots X_n) \in C\})$

3.2.3.3. *Proposition*

Les fonctions α et γ vérifient les hypothèses de [Cousot 78b]:

(i) - α et γ sont isotones pour l'inclusion

(ii) - $\forall Q \in P(T)$, $Q \subseteq \gamma(\alpha(Q))$

(iii) - $\forall P \in C(\mathbb{R}^n)$, $P = \alpha(\gamma(P))$

Il en résulte [Cousot 79] que $\gamma \circ \alpha$ est une fermeture sur le treillis complet $P(T)$, $C(\mathbb{R}^n)$ est un treillis complet pour l'inclusion, α est un morphisme complet pour l'union et surjectif, γ est un morphisme complet pour l'intersection et injectif.

3.2.3.4.

La fonction d'abstraction α permet d'associer à tout contexte concret un contexte abstrait qui est une partie convexe de $\mathbb{R}^n$. La détermination de relations

de ce type entre les variables (approximation convexe du domaine) paraît encore trop complexe pour être automatisée de manière satisfaisante. Nous nous sommes restreints à l'étude des relations linéaires, c'est-à-dire que nous nous intéressons à l'ensemble $K(\mathbb{R}^n)$ des polyèdres convexes de $\mathbb{R}^n$, qui est un sous treillis de $C(\mathbb{R}^n)$. Ce sous-treillis n'est pas complet:

Contre-exemple:

Pour tout $\theta\in[\theta,2\pi]$, posons $P(\theta)=\{X\in\mathbb{R}^2 \mid X_1\cos\theta+X_2\sin\theta\leq 1\}$. Pour tout θ, $P(\theta)$ est un demi-espace de $\mathbb{R}^2$, mais l'intersection de ces demi-espaces, lorsque θ prend toutes les valeurs entre 0 et 2π, est le disque de centre 0 et de rayon 1, qui est convexe mais n'est pas un polyèdre. □

Deux remarques permettent de résoudre ce problème:

. De par les limites de la machine, les nombres manipulés par un programme appartiennent à des ensembles finis. Donc pour toute partie Q de T, l'ensemble $\{(X_1\ldots X_n)\mid\exists(X_{n+1}\ldots X_p)$ tels que $(X_1\ldots X_p)\in Q\}$ est fini. Donc $\alpha(Q)$ est l'enveloppe convexe d'un ensemble fini de points de $\mathbb{R}^n$, c'est donc un polyèdre convexe (théorème 1.2.3.).

. D'autre part les contextes abstraits vont être approchés au moyen de séquences finies de polyèdres dont la limite sera donc toujours un polyèdre.

3.3. Représentation des Contextes Abstraits

Notre but est d'associer automatiquement à tout arc Q_i d'un graphe de programme, un polyèdre convexe $P(a_i,\phi)$ tel que

- soit $P(a_i,\phi)\supseteq Q(a_i,\phi)$ (analyse en avant, chap. 4 et 5)
- soit $P(a_i,\phi)\supseteq Q'(a_i,\phi)$ (analyse en arrière, chap.6)

Nous avons vu (chap.1) que si P était un polyèdre convexe de $\mathbb{R}^n$ alors:

- d'une part il existait une matrice A et un vecteur B tels que

  $(X\in P)\iff(AX\leq B)$ (*système de contraintes*)

- d'autre part il existait un *système générateur* de P, formé de l'ensemble $S=\{s_1\ldots s_\sigma\}$ des sommets, de l'ensemble $R=\{r_1\ldots r_\rho\}$ des rayons extrémaux et de l'ensemble $D=\{d_1\ldots d_\delta\}$ des droites, tels que:

$$X \in P \iff \begin{cases} \exists\ \lambda_1 \ldots \lambda_\sigma \in [0,1],\ \mu_1 \ldots \mu_\rho \in \mathbb{R}^+,\ \nu_1 \ldots \nu_\delta \in \mathbb{R} \\ \text{tels que } \sum_{i=1}^{\sigma} \lambda_i = 1 \text{ et } X = \sum_{i=1}^{\sigma} \lambda_i\ s_i + \sum_{j=1}^{\rho} \mu_j\ r_j + \sum_{k=1}^{\delta} \nu_k\ d_k \end{cases}$$

On a remarqué d'autre part:

- que le calcul de l'enveloppe convexe de deux polyèdres nécessitait la connaissance des systèmes générateurs de ces polyèdres.
- que pour déterminer si un polyèdre P_1 était inclus dans un polyèdre P_2 , il était utile de connaître un système générateur de P_1 et un système de contraintes de P_2.
- que les opérations de base sur chacune des représentations faisaient apparaître une grande quantité d'éléments redondants et donc, qu'aucune représentation ne pouvait être utilisée sans simplification.
- que pour simplifier un système de contraintes, ou un système générateur, il était capital de connaître la représentation duale.
- qu'enfin, le passage d'une représentation à l'autre risquait de coûter fortcher.

En conséquence, pour éviter d'avoir à effectuer des conversions trop fréquentes, nous conserverons dans nos contextes abstraits l'information (redondante) relative aux deux représentations. Toute fonction agissant sur les contextes abstraits devra donc être définie d'une part sur les systèmes de contraintes, et d'autre part sur les systèmes générateurs.

Un contexte abstrait sera représenté par un quintuplet (A,B,S,R,D) où

- A est une matrice réelle à m lignes et n colonnes,
- B est un m-vecteur réel,
- S, R et D sont trois ensembles de n-vecteurs:
  $S=\{s_1, s_2 \ldots s_\sigma\}$, $R=\{r_1, r_2 \ldots r_\rho\}$, $D=\{d_1, d_2 \ldots d_\delta\}$

Nous verrons au chapitre 9 que, quoique le modèle abstrait soit $K(\mathbb{R}^n)$, les contextes pourront toujours être exprimés au moyen de nombre rationnels, ce qui est plus conforme aux limitations de calcul sur machine.

4. SYSTEME D'EQUATIONS EN AVANT ASSOCIE A UN PROGRAMME

4.1. Introduction

Nous sommes maintenant en mesure de préciser comment le contexte abstrait associé à un arc d'un graphe de programme peut se déduire des contextes associés aux arcs "voisins", conformément à la sémantique du langage. Nous avons vu qu'il existait deux types d'analyses sémantiques:

. L'analyse sémantique en avant, qui caractérise l'ensemble des états accessibles à partir d'une spécification d'entrée. A l'arc a_i sera alors associé un contexte abstrait calculé à partir des contextes associés aux arcs incidents au noeud origine de a_i.

. L'analyse sémantique en arrière (qui fera l'objet du chapitre 6), qui caractérise l'ensemble des états à partir desquels on peut atteindre une spécification de sortie. Le contexte associé à l'arc a_i sera déduit des contextes des arcs issus du noeud extrémité de a_i.

Si a est un arc issu du noeud N et si $b_1 \ldots b_k$ sont les arcs incidents à N, notons $P, Q_1 \ldots Q_k$ les contextes abstraits respectivement associés à $a, b_1 \ldots b_k$. D'autre part, notons $post(P_1 \ldots P_k, N, a)$ la plus forte post condition de l'instruction N impliquée sur l'arc a par les prédicats incidents $P_1 \ldots P_k$. La fonction *post* est définie à partir des "conditions de vérifications" de Floyd[Floyd 67].

Le choix optimal pour P est $\alpha(post(\gamma(Q_1), \ldots, \gamma(Q_k), N, a)$ [Cousot 79]. Il nous reste donc à développer cette expression pour chaque type de noeud, ce qui donne un algorithme pour calculer P. Pour simplifier cet algorithme, nous serons parfois amenés à choisir une approximation supérieure de cette expression.

4.2. Interprétation en avant des constructions élémentaires du langage

4.2.1. Noeud d'entrée

Si l'on connaît une spécification d'entrée ϕ du programme, sous forme de contraintes linéaires, ces contraintes formeront le contexte associé à l'arc

d'entrée du programme, moyennant une recherche du système générateur correspondant. Si l'on ne possède aucune information sur les valeurs initiales des variables, on fait l'hypothèse que le vecteur des valeurs des variables peut être n'importe quel vecteur de $\mathbb{R}^n$.

$\mathbb{R}^n$ est le polyèdre à zéro contrainte . Un système générateur en est donné par:

- un seul sommet qui est par exemple l'origine de l'espace,
- pas de rayon,
- les droites d(1)...d(n) définies par:

  $\forall i,j = 1...n,\ (i \neq j) \Rightarrow (d(i)_j = 0 \text{ et } d(i)_i = 1)$

4.2.2. Noeud d'affectation

Soit (A,B,S,R,D) la représentation du contexte associé à l'arc incident à un noeud correspondant à l'affectation $X_i := E(X)$.

On sait que

$$post(Q, X_i := E(X)) = (\exists X_0 \in T_i \mid def(E(X_1 \ldots X_{i-1}\ X_0 \ldots X_p)) \wedge Q(X_1 \ldots X_{i-1}, X_0, X_{i+1} \ldots X_p) \wedge X_i = E(X_1 \ldots X_{i-1}, X_0, X_{i+1} \ldots X_p))$$

def(E) dénote l'ensemble des conditions pour que le calcul de E s'effectue correctement. Pour simplifier, nous négligerons souvent *def*(E) mais la prise en compte des conditions de cohérence telles que *def*(E) fera l'objet du paragraphe 4.2.5.

Le contexte P, associé à l'arc issu du noeud d'affectation doit donc être:

. si i>n (affectation à une variable non numérique):

$$P \supseteq \alpha(\exists X_0 \mid def(E(X_1 \ldots X_n \ldots X_{i-1}\ X_0 \ldots X_p)) \wedge (AX_{i \ldots n} \leq B) \wedge X_i = E(X_1 \ldots X_n \ldots X_{i-1}\ X_0 \ldots X_p))$$

Il en résulte que les variables numériques $X_1 \ldots X_n$ satisfont aux mêmes contraintes qu'avant l'affectation. Cependant les conditions pour que E soit bien définie peuvent induire des contraintes supplémentaires que nous ne considérerons pas en détail.

. si i≤n

$$P \supseteq \alpha(\exists X_0 \mid def(E(X_1 \ldots X_{i-1},\ X_0 \ldots X_p)) \wedge (\bigwedge_{k=1}^{m} A_k^1 X_1 + \ldots + A_k^{i-1} X_{i-1} + A_k^i X_0 + \ldots + A_k^n X_n \leq B_k) \wedge X_i = E(X_1 \ldots X_0 \ldots X_n))$$

P va alors dépendre de la forme particulière de l'expression E.

4.2.2.1. Affectation d'une expression non linéaire

4.2.2.1.1. - Si E n'est pas une expression linéaire des variables numériques, le modèle ne permet pas en général de rendre compte des contraintes vérifiées par la valeur de cette expression. Le traitement de cas particuliers peut être facilement incorporé (par exemple on sait que $(X-k+1)/k \leq X \mathit{div} k \leq X/k$ ou que $-1 \leq \cos X \leq 1$, ou que $X \geq 1 \Rightarrow \log X \geq 0$) mais en général toute connaissance sur la valeur de X_i est perdue:

$$P = (\exists X_0 \mid \bigwedge_{k=1}^{m} A_k^1 X + \ldots + A_k^{i-1} X_{i-1} + A_k^i X_0 + \ldots + A_k^n X_n \leq B_k)$$

La variable X_i est donc éliminée du système $(AX \leq B)$ par projection (cf.2.1.) et la droite d(i) (cf. 4.2.1.) est adjointe au système générateur (S,R,D). Les deux représentations obtenues doivent alors être simplifiées (cf. 1.3.2. & 1.3.3.).

4.2.2.1.2. *Exemple*

Supposons que le programme ne manipule que deux variables et que le contexte incident soit

$$\begin{cases} X_1 - X_2 \geq -1 \\ X_2 \geq 1 \\ X_1 + X_2 \geq 5 \end{cases} \quad S = \{\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}4\\1\end{pmatrix}\} \quad R = \{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}\} \quad D = \emptyset$$

Ce contexte est représenté sur la fig. 4.1.a.

Alors, après l'affectation "$X_2 := X_1 * X_2$" le contexte résultant sera (fig. 4.1.b)

$$\{X_1 \geq 2 \qquad S = \{\begin{pmatrix}2\\0\end{pmatrix}\} \quad R = \{\begin{pmatrix}1\\0\end{pmatrix}\} \quad D = \{\begin{pmatrix}0\\1\end{pmatrix}\}$$

L'approximation est assez grossière, puisqu'en substituant X_2/X_1 à X_2 dans le système initial on obtient (fig. 4.1.c.)

$$\begin{cases} (X_1)^2 + X_1 \geq X_2 \\ X_1 \leq X_2 \\ -(X_1)^2 + 5X_1 \leq X_2 \end{cases}$$

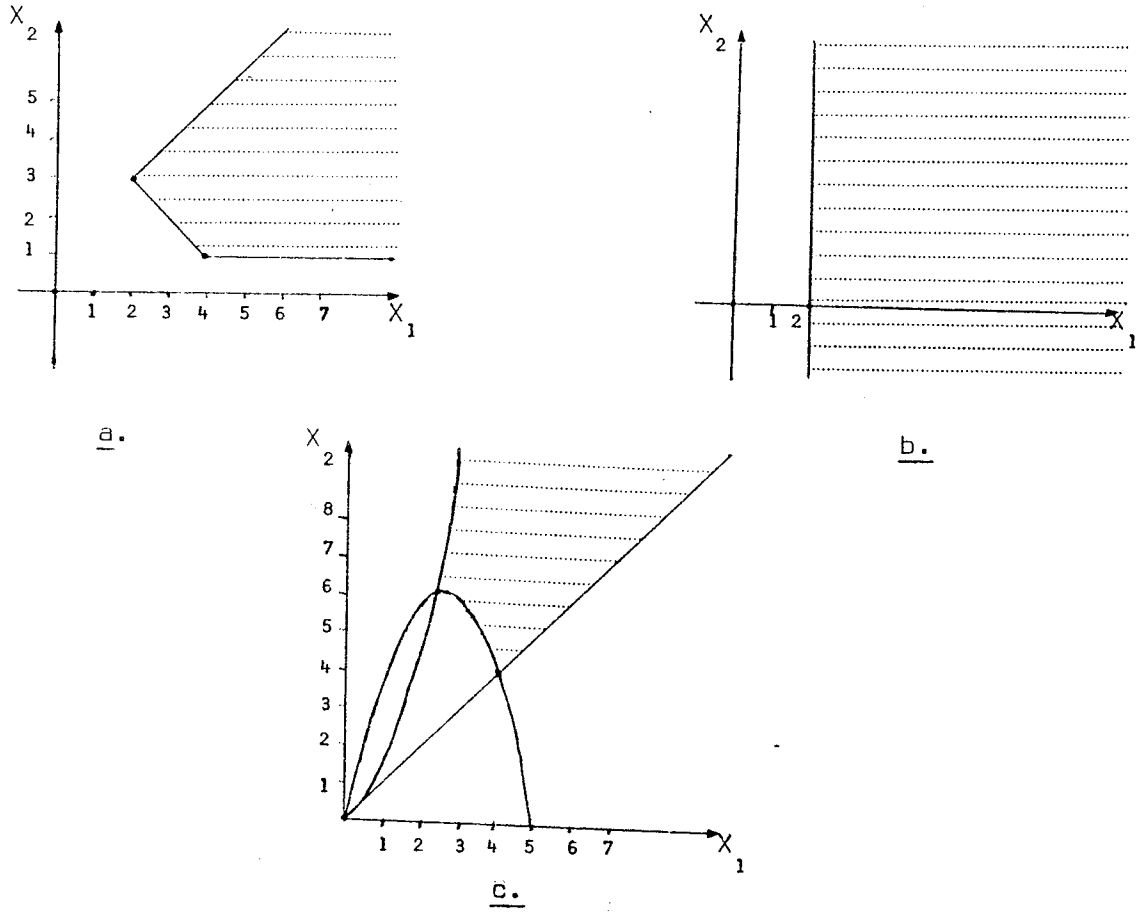


figure 4.1.

4.2.2.2. Affectation d'une expression linéaire

4.2.2.2.1. *Proposition*

Si (S,R,D) est un système générateur du polyèdre Q attaché à l'arc incident à un noeud d'affectation $X_i := aX+b$, où a est un n-vecteur ligne de réels et b est un réel, alors le plus petit polyèdre que l'on puisse associer à l'arc issu de ce noeud est le polyèdre P dont un système générateur (S',R',D') est défini comme suit:

- $S' = \{s'_1 \ldots s'_\sigma\}$, avec $\forall k=1\ldots\sigma$, $(j\neq i) \Rightarrow (s'_{kj}=s_{kj})$ et $s'_{ki}=as_k+b$
- $R' = \{r'_1 \ldots r'_\rho\}$, avec $\forall k=1\ldots\rho$, $(j\neq i) \Rightarrow (r'_{kj}=r_{kj})$ et $r'_{ki}=ar_k$
- $D' = \{d'_1 \ldots d'_\delta\}$, avec $\forall k=1\ldots\delta$, $(j\neq i) \Rightarrow (d'_{kj}=d_{kj})$ et $d'_{ki}=ad_k$

Démonstration

Si l'on néglige *def*(aX+b), le plus petit polyèdre P conforme à la sémantique de l'affectation est défini par:

$P=\alpha(\exists X_0 \mid (X_1 \ldots X_{i-1}\ X_0 \ldots X_n) \in P$ et $X_i=a_1X_1+\ldots+a_{i-1}\ X_{i-1}+a_iX_0+\ldots+a_nX_n)$

Donc:

$P=\alpha(\exists X_0, \exists(\lambda_1 \ldots \lambda_\sigma) \in [0,1]^\sigma, \exists(\mu_1 \ldots \mu_\rho) \in \mathbb{R}^{+\rho}, \exists(\nu_1 \ldots \nu_\delta) \in \mathbb{R}^\delta$ tels que

$\Sigma_{k=1}^{\sigma}\ \lambda_k=1$ et $\forall j\neq i$, $X_j=\Sigma_{k=1}^{\sigma}\ \lambda_k s_{kj}+\Sigma_{k=1}^{\rho}\ \mu_k r_{kj}+\Sigma_{k=1}^{\delta}\ \nu_k d_{kj}$, et

$X_0=\Sigma_{k=1}^{\sigma}\ \lambda_k s_{ki}+\Sigma_{k=1}^{\rho}\ \mu_k r_{ki}+\Sigma_{k=1}^{\delta}\ \nu_k d_{ki}$, et $X_i=a_1X_1+\ldots+a_iX_0+\ldots+a_nX_n)$

Il vient:

$X_i=\Sigma_{\ell=1}^{n}(\Sigma_{k=1}^{\sigma}\ \lambda_k a_\ell s_{k\ell}+\Sigma_{k=1}^{\rho}\ \mu_k a_\ell r_{k\ell}+\Sigma_{k=1}^{\delta}\ \nu_k a_\ell d_{k\ell})$

Donc:

$P=\alpha(\exists(\lambda_1 \ldots \lambda_\sigma) \in [0,1]^\sigma, \exists(\mu_1 \ldots \mu_\rho) \in \mathbb{R}^{+\rho}, \exists(\nu_1 \ldots \nu_\delta) \in \mathbb{R}^\delta$, tels que

$\Sigma_{k=1}^{\sigma}\ \lambda_k=1$ et $\forall j\neq i$, $X_j=\Sigma_{k=1}^{\sigma}\ \lambda_k s_{kj}+\Sigma_{k=1}^{\rho}\ \mu_k r_{kj}+\Sigma_{k=1}^{\delta}\ \nu_k d_{kj}$, et

$X_i=\Sigma_{k=1}^{\sigma}\ \lambda_k a.s_k+\Sigma_{k=1}^{\rho}\ \mu_k a.r_k+\Sigma_{k=1}^{\delta}\ \mu_k\ a.d_k)$

Ceci exprime que les ensembles (S',R',D') de la proposition formant un système générateur de P. □

4.2.2.2.2.*Proposition*

Si Q est le polyèdre attaché à l'arc incident au noeud d'affectation $X_i:=aX+b$, où a est un n vecteur ligne de réels et b est un réel et si $a_i=0$, alors $P=(proj(Q,X_i) \cap \{X_i=aX+b\})$ est le plus petit polyèdre que l'on puisse associer à l'arc issu de ce noeud. Dans ce cas l'affectation est dite *non inversible.*

Démonstration

Soit $(AX \leq B)$ un système de contraintes de Q. La condition de validité s'écrit:

$$P = \alpha(\exists X_0 \mid \bigwedge_{k=1}^{m} A_k^1X_1+\ldots A_k^{i-1}\ X_{i-1}+A_k^i\ X_0+\ldots+A_k^n \leq B_k \text{ et}$$

$$X_i = a_1X_1+\ldots+a_{i-1}\ X_{i-1}+a_{i+1}X_{i+1}+\ldots+a_nX_n)$$

Ceci exprime que P est la projection sur $\mathbb{R}^n$ selon X_0 du polyèdre en $(X_0, X_1 \ldots X_n)$ dont un système de contrainte est donné ci-dessus. Comme la dernière équation ne contient pas de terme en X_0 , elle est recopiée telle quelle dans le système projeté. Le reste du système à projeter est égal au système de contraintes de Q dans lequel la variable X_i a été renommée X_0 , ce qui ne change rien au résultat de la projection. □

4.2.2.2.3. *Proposition*

Avec les mêmes hypothèses, mais en supposant maintenant que $a_i\neq 0$ (*affectation inversible)* , si $(AX \leq B)$ est un système de contraintes de Q, alors $(A'X \leq B')$ est

un système de contraintes de P, où

- $\forall k = 1...m,\ B'_k = B_k + (b/a_i)A^i_k$
- $\forall k = 1...m,\ \forall j=1...n,\ (i \neq j) \Rightarrow (A'^j_k = A^j_k - (a_j/a_i).A^i_k)$
- $\forall k = 1...m,\ A'^i_k = A^i_k/a_i$

Démonstration

$$P = \alpha(\exists X_0 \mid \bigwedge_{k=1}^{m} A^1_k X_1 + ... A^{i-1}_k X_{i-1} + A^i_k X_0 + ... + A^n_k X_n \leq B_k \text{ et } X_i = a_1 X_1 + ... + a_i X_0 + ... + a_n X_n + b)$$

$$P = (\bigwedge_{k=1}^{m} A^1_k X_1 + ... + A^{i-1}_k X_{i-1} + A^i_k((X_i - a_1 X_1 - ... - a_{i-1} X_{i-1} - a_{i+1} X_{i+1} - ... - a_n X_n - b/a_i) + A^{i+1}_k X_{i+1} + ... + A^n_k X_n \leq B_k)$$

$$P = (\bigwedge_{k=1}^{m} (A^1_k - (a_1/a_i)A^i_k)X_1 + ... + (A^{i-1}_k - (a_{i-1}/a_i)A^i_k)X_{i-1} + (A^i_k/a_i)X_i + (A^{i+1}_k - (a_{i+1}/a_i)A^i_k)X_{i+1} + A^n_k - (a_n/a_i)A^n_k \leq B_k + (b/a_i)A^i_k)$$

$$P = (\bigwedge_{k=1}^{m} A'_k X \leq B'_k)$$

□

4.2.2.2.4. *Exemples*

Soit Q le contexte incident défini par (fig.4.2.a):

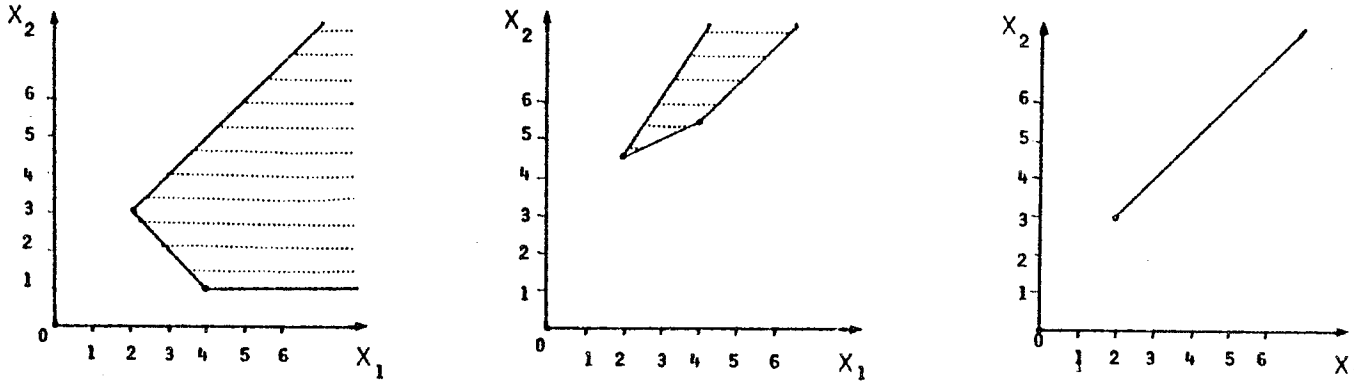
$$\begin{cases} X_2 \geq 1 \\ X_1 + X_2 \geq 5 \\ X_1 - X_2 \geq -1 \end{cases} \qquad S = \{\begin{pmatrix}2\\3\end{pmatrix}, \begin{pmatrix}4\\1\end{pmatrix}\},\ R = \{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}\} \qquad D = \emptyset$$

. Si l'on effectue dans ce contexte l'affectation inversible $X_2 := X_1 + X_2/2 + 1$ on obtient comme contexte de sortie (fig. 4.2.b)

$$\begin{cases} -2X_1 + 2X_2 \geq 3 \\ -X_1 + 2X_2 \geq 7 \\ 3X_1 - 2X_2 \geq 3 \end{cases} \qquad S' = \{\begin{pmatrix}2\\9/2\end{pmatrix}, \begin{pmatrix}4\\11/2\end{pmatrix}\}, R' = \{\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}1\\3/2\end{pmatrix}\},\ D' = \emptyset$$

. Dans le même contexte Q, on effectue l'affectation $X_2 := X_1 + 1$, qui n'est pas inversible. On obtient (fig.4.2.c)

$$\begin{cases} X_1 \geq 2 \\ -X_1+X_2=1 \end{cases} \qquad S''=\{\binom{2}{3}\} \ , \ R''=\{\binom{1}{1}\} \ , \ D''=\emptyset$$



a. b. c.

figure 4.2.

4.2.2.2.5. *Remarques*

. Dans le cas d'une affectation linéaire, à une précondition qui est un polyèdre conrespond une plus forte postcondition qui est aussi un polyèdre dont on sait calculer les deux représentations. La transformation s'effectue donc sans perte d'information (sauf qu'il n'est pas tenu compte des débordements arithmétiques).

. Si le contexte incident était sous une représentation minimale et si l'affectation est inversible, les systèmes obtenus sont minimaux. Dans le cas d'une affectation non inversible, il convient de simplifier le système de contraintes et le système générateur obtenus.

4.2.2.3. Isotonie de l'interprétation de l'affectation

La transformation que l'on vient de définir sur les contextes abstraits sera notée *affect*$(Q,X_i:=E(X))$.

4.2.2.3.1. *Théorème*

Quelle que soit l'instruction d'affectation $X_i:=E(X)$, la fonction $\lambda Q.$*affect*$(Q,X_i:=E(X))$ est isotone (=monotone croissante) sur le treillis des polyèdres convexes.

Démonstration

. Si E(X) est non linéaire, il existe un vecteur d de $\mathbb{R}^n$ tel que $\forall Q \in K(\mathbb{R}^n)$, $(X \in affect(Q, X_i := E(X)) \iff (\exists v \in \mathbb{R}, Y \in P \mid X = Y + vd)$. Alors si $Q_1 \subseteq Q_2$, $X \in affect(Q_1, X_i := E(X)) \iff (\exists v \in \mathbb{R}, Y \in Q_1 \mid X = Y + vd) \implies (\exists v \in \mathbb{R}, y \in Q_2 \mid X = Y + vd) \iff X \in affect(Q_2, X_i := E(X))$

. Si E(X) est linéaire, $affect(Q, X_i := E(X)) = \alpha(post(\gamma(Q), X_i := E(X))$. Or α, γ et *post* sont isotones, donc *affect* l'est aussi.

4.2.3. Noeud de test

Un noeud de test possède un arc prédécesseur et deux arcs successeurs correspondant respectivement aux branchements selon la valeur "vrai" ou "faux"de la condition C sur laquelle porte le test. Soit Q le contexte associé à l'arc incident. On veut en déduire les deux contextes $P_V = vrai(C, Q)$ et $P_F = faux(C, Q)$ que l'on peut associer respectivement aux arcs successeurs.

Les conditions de validité sont les suivantes (en négligeant les conditions nécessaires à la bonne définition de l'expression C):

- $P_V \supseteq \alpha(\gamma(Q) \wedge C)$
- $P_F \supseteq \alpha(\gamma(Q) \wedge \neg C)$

4.2.3.1. Condition non linéaire

Si C n'est pas une relation linéaire des variables numériques, une étude particulière permet souvent de trouver deux polyèdres K_1 et K_2 tels que:

$$\begin{array}{l} \{X \in \mathbb{R}^n \mid C(X)\} \subseteq K_1 \\ \{X \in \mathbb{R}^n \mid \neg C(X)\} \subseteq K_2 \end{array}$$

(Par exemple C=(LogX ≥ 0), $K_1 = \{1 \leq X\}$, $K_2 = \{0 \leq X \leq 1\}$). Alors $P_V = Q \cap K_1$, $P_F = Q \cap K_2$ sont des solutions convenables. En général, si rien n'a été prévu pour déduire K_1 et K_2, on prendra $P_V = P_F = Q$, solutions qui satisfont les conditions de validité.

4.2.3.2. Condition linéaire

4.2.3.2.1. *Proposition*

Si Q est le contexte incident à un noeud de test portant sur la condition "aX=b", où a est un n-vecteur ligne, et b un réel, alors les plus petits polyèdres convexes

que l'on peut associer respectivement aux arcs "vrai" et "faux" de ce noeud sont les suivants:

- $P_V = Q \cap H$
- Si $Q \subseteq H$ alors $P_F = \emptyset$ sinon $P_F = Q$

où $H=\{X \mid aX=b\}$

Démonstration

$P_V = \alpha(\gamma(Q) \wedge (aX=b)) = Q \cap H$

$P_F = \alpha(\gamma(Q) \wedge (aX \neq b))$

si $\forall X \in Q$, $aX=b$, alors $P_F = \alpha(faux) = \emptyset$

sinon P_F est le plus petit ensemble convexe et fermé contenant tous les points de Q qui n'appartiennent pas à H, lequel est de dimension strictement inférieure à la dimension de Q. On démontre ([Von Hohenbalken 78], théorèmes 1.1. & 1.2.) que tout point de $Q \cap H$ est combinaison convexe de points de Q-H, sauf éventuellement les points des faces de $Q \cap H$. Or si f est une face d'un polyèdre P, de dimension strictement inférieure à la dimension de P, la fermeture de P-f est égale à P. Donc $Q \cap H \subseteq P_F$, et $P_F = Q$.

4.2.3.2.2. *Proposition*

Avec les notations de la proposition 4.2.3.2.1.,mais si la condition est de la forme "$aX \leq b$", on a $P_V = Q \cap E_1$ et $P_F = Q \cap E_2$, avec $E_1 = \{X \mid aX \leq b\}$ et $E_2 = \{X \mid aX \geq b\}$.

Il résulte des deux propositions précédentes que si la condition C est linéaire, le calcul de *vrai*(C,Q) et *faux*(C,Q) revient à calculer l'intersection de Q avec un hyperplan ou un demi-espace. Nous savons effectuer ce calcul pour les deux représentations (cf. 2.3.1.2. & 2.3.1.4.).

Jusqu'à présent on a toujours considéré que les variables prenaient leurs valeurs dans $\mathbb{R}$, c'est-à-dire que $\gamma(Q):\{X \in \mathbb{R}^n \mid X \in Q\}$. Une prise en compte plus précise des domaines des valeurs permet souvent une interprétation plus fine, surtout lorsque ces domaines sont discrets.

4.2.3.2.3. *Proposition*

Soit a un n-vecteur ligne d'entiers tel que $\forall i=1...n$, si l'ensemble des valeurs de type T_i n'est pas inclus dans l'ensemble des entiers (relatifs), alors $a_i=0$.

Soit b un entier. On pose:

$$\hat{a}_0 = \min_{1\le i\le n}(|a_i|)$$

$$H = \{X \in \mathbb{R}^n \mid aX=b\}$$

$$E_1 = \{X \in \mathbb{R}^n \mid aX\le b\},\ E_2=\{X \in \mathbb{R}^n \mid aX\ge b\}$$

$$E_1' = \{X \in \mathbb{R}^n \mid aX\le b-a_0\},\ E_2' =\{X \in \mathbb{R}^n \mid aX\ge b+a_0\}$$

Alors, avec les notations de la proposition 4.2.3.2.1.:

- Si la condition est "aX=b", on a

  $P_V=Q \cap H$

  Si $Q \subseteq E_1$ alors $P_F=Q \cap E_1'$ sinon si $Q \subseteq E_2$ alors $P_F=Q \cap E_2'$ sinon $P_F=Q$
- Si la condition est "aX≤b", on a

  $P_V=P \cap E_1$, $P_F=P \cap E_2'$

Démonstration

Il suffit d'observer que

$\alpha(\gamma(Q) \cap \{X \mid aX\neq b\}) = env(Q \cap \{X \mid aX\le b-a_0\},\ Q \cap \{X \mid aX\ge b+a_0\})$

et que

$\alpha(\gamma(Q) \cap \{X \mid aX<b\}) = Q \cap \{X \mid aX\le b-a_0\}$

4.2.3.2.4. *Exemple*

$Q = (\{X_2\ge 1, X_1+X_2\ge 5,\ X_1-X_2\ge -1\},\ S=\{(4,1),(2,3)\},\ R=\{(1,0),(1,1)\},\ D=\emptyset)$

$C = "X_1+6\le 2X_2"$.

- Si X_1 et X_2 sont de type *réel* on obtient (fig.4.3.a):

  $P_V = (\{X_1-X_2\ge -1, -X_1+2X_2\ge 6\},\ S=\{(4,5)\},\ R=\{(1,1),(2,1)\},\ D=\emptyset)$

  $P_F = (\{X_2\ge 1, X_1+X_2\ge 5,\ X_1-X_2\ge -1,\ X_1-2X_2\ge -6\},$

  $S = \{(4,1),(2,3),(4,5)\},\ R=\{(1,0),(2,1)\},\ D=\emptyset)$
- Par contre si X_1 et X_2 sont de type entier, le domaine P_F obtenu est plus précis (fig. 4.3.b):

  $P_F = (\{X_2\ge 1, X_1+X_2\ge 5, X_1-X_2\ge -1, X_1-2X_2\ge -5\},$

  $S = \{(4,1),(2,3),(3,4)\},\ R=\{(1,0),(2,1)\},\ D=\emptyset)$

D'après les propriétés élémentaires de l'intersection des ensembles, il est clair que les fonctions $\lambda Q.vrai(C,Q)$ et le $\lambda Q.faux(C,Q)$ sont isotones sur le treillis des polyèdres, quelle que soit la condition C.

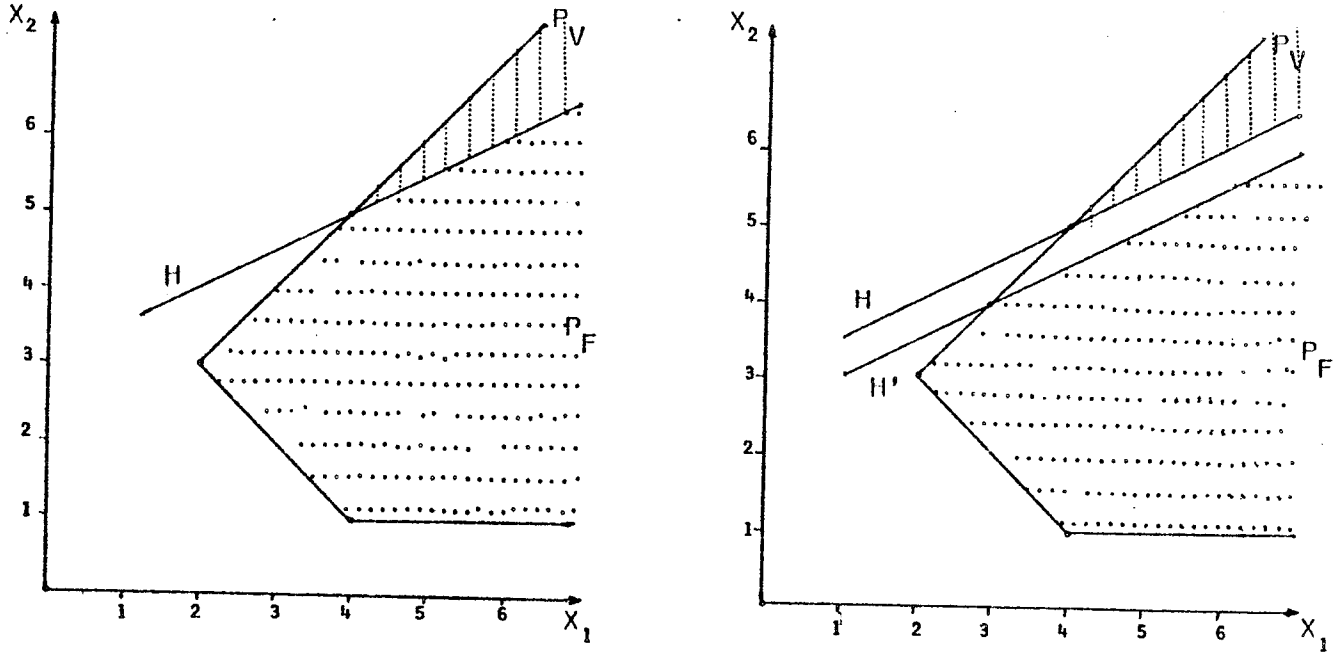


figure 4.3.

4.2.4. Noeud de jonction

Soient $Q_1, Q_2 \ldots Q_r$ les contextes associés respectivement aux arcs incidents à un noeud de jonction. Le contexte P associé à l'arc issu de ce noeud doit vérifier:

$$P \supseteq \alpha\left(\bigvee_{k=1}^{r} \gamma(Q_k)\right)$$

Donc, le plus petit polyèdre qu'on puisse associer à cet arc est

$$\alpha\left(\bigvee_{k=1}^{r} \gamma(Q_k)\right) = env(Q_1 \ldots Q_k)$$

Connaissant les systèmes générateurs de $Q_1 \ldots Q_k$, on sait (cf. 2.2.5) calculer un système générateur et un système de contraintes de $env(Q_1 \ldots Q_k)$.

Exemple

n=2, k=2

Q_1 = ({$X_1 \geq 0, X_2 \geq 0, X_1 + X_2 \leq 1$}, S_1={(0,0),(1,0),(0,1)}, $R_1 = D_1 = \emptyset$)

Q_2 = ({$X_1 \geq 1, X_2 = 2$}, S_2={(1,2)}, R_2={(1,0)}, $D_2 = \emptyset$)

◦ L'enveloppe convexe Q_3 de Q_1 et du sommet s=(1,2) de Q_2 est obtenue en éliminant λ du système développé:

$\{0 \leq \lambda \leq 1,\ A(1)X + \lambda(A(1)s - B(1)) \leq A(1)s\}$

$\{A(1)X \leq B(1)\}$ étant le système de contraintes de Q_1. On obtient le système $\{A(3)X \leq B(3)\}$ de Q_3:

$\{X_2 - X_1 \leq 1,\ 0 \leq X_1 \leq 1,\ X_2 \geq 0\}$

. $env(Q_1,Q_2) = env(Q_3,r)$, où $r=(1,0)$ est le rayon de Q_2.

En éliminant μ du système développé:

$\{\mu \geq 0,\ A(3)X - \mu A(3)r \leq B(3)\}$

on obtient le système de contraintes de $env(Q_1,Q_2)$:

$\{0 \leq X_2 \leq 2,\ X_1 \geq 0,\ X_2 - X_1 \leq 1\}$

. $(S_1 \cup S_2,\ R_1 \cup R_2,\ D_1 \cup D_2)$ est un système générateur de $env(Q_1,Q_2)$. Après simplification, on trouve:

$S = \{(0,0),(0,1),(1,2)\},\quad R=\{(1,0)\},\quad D=\emptyset$

4.2.5. Exploitation des tests dynamiques

Si l'on tient compte du fait qu'avant d'être exécuté, le programme va être traité par un compilateur, qui va placer dans le corps du programme un certain nombre de tests destinés à détecter des erreurs à l'exécution, on peut tirer parti de ces tests dans l'élaboration des contextes abstraits. En effet, après le calcul d'une expression, soit l'exécution du programme a été arrêtée par la détection d'une condition exceptionnelle, soit tous les tests dynamiques relatifs au calcul de cette expression ont donné une réponse positive, ce qui peut être incorporé au contexte successeur.

Par exemple, si un tableau T[1...N] a été déclaré, après tout accès à un élément T[I], on peut affirmer que $1 \leq I \leq N$.

4.3. Système d'équations sémantiques approchées

Soient $\{a_1 \ldots a_\alpha\}$ l'ensemble des arcs d'un graphe de programme π. Pour tout $k=1\ldots\alpha$, on connait maintenant une expression $F_k(P_1 \ldots P_\alpha)$ du contexte abstrait P_k que l'on veut associer à a_k, en fonction du noeud origine de a_k et des contextes associés aux arcs incidents à ce noeud. Les contextes cherchés sont donc la plus

petite solution $\overline{P}$ du système d'équations $\{P_k=F_k(P_1\ldots P_\alpha)\,|\,k=1\ldots\alpha\}$ associé à π. Comme toutes les fonctions F_k sont isotones, une telle solution existe dans le treillis complet $C(\mathbb{R}^n)^\alpha$. $\overline{P}$ n'est en général pas calculable, mais si $\tilde{P}$ est tel que $\forall k=1\ldots\alpha,\ \tilde{P}_k \geq F_k(\tilde{P}_1\ldots\tilde{P}_\alpha)$ alors $\tilde{P}$ est une approximation correcte de $\overline{P}(\forall k=1\ldots\alpha, \tilde{P}_k \geq \overline{P}_k)$. Le chapitre suivant applique les résultats de [Cousot 77a,78b], pour calculer un tel $\tilde{P}$, dans $K(\mathbb{R}^n)^\alpha$, comme limite d'une itération croissante approchée supérieurement, et pour améliorer le résultat obtenu par une itération décroissante approchée supérieurement.

5 - ANALYSE APPROCHEE EN AVANT DES PROGRAMMES

5.1 - Introduction

La meilleure solution $\bar{P}$ du système d'équations sémantiques approchées en avant n'étant en général pas calculable, nous allons calculer une approximation $\tilde{P}$ de $\bar{P}$, telle que $\forall k = 1 \ldots \alpha$, $F_k(\tilde{P}_1 \ldots \tilde{P}_k) \subseteq \tilde{P}_k$. Chacun des $\tilde{P}_k$ sera obtenu comme limite d'une séquence finie de polyèdres P_k^i (§ 5.2). Pour construire ces séquences d'approximation, nous définissons un opérateur d'élargissement sur les polyèdres (§ 5.3) et nous étudions une stratégie de calcul (§ 5.4). La solution approchée $\tilde{P}$ est alors améliorée par une séquence décroissante (§ 5.5). La théorie générale décrivant ce procédé est développée dans [Cousot 78b, Ch. 4].

5.2 - Séquence Croissante d'Approximation

Rappelons les grands traits de la méthode d'approximation par une itération croissante approchée supérieurement ([Cousot 77a, 78b]). Pour obtenir la solution approchée $\tilde{P}$, nous allons définir une fonction d'interprétation approchée $\tilde{F}$ telle que : $\forall k = 1 \ldots \alpha$:

- si $P_k \supseteq F_k(P_1 \ldots P_\alpha)$ alors $\tilde{F}_k(P_1 \ldots P_\alpha) = P_k$
- sinon $\tilde{F}_k(P_1 \ldots P_\alpha) \supseteq env\ (P_k, F_k(P_1 \ldots P_k))$

Alors on construira une séquence $P_{k_1}^1 \ldots P_{k_s}^s \ldots$ de polyèdres tels que :

. $(1 \leq s \leq \alpha) \Rightarrow (k_s = s \text{ et } P^s_s = \emptyset)$

. $(s > \alpha) \Rightarrow (P^s_{k_s} = \tilde{F}_{k_s}(P_1^{\Delta(1,s)},\ldots,P_\alpha^{\Delta(\alpha,s)}))$

où $\Delta(k,s) = \max \{s' \mid s' < s \text{ et } k_{s'} = k\}$

. $\forall k = 1\ldots\alpha$, $\forall s > 0$, si $P_k^{\Delta(k,s)} \neq \tilde{F}_k(P_1^{\Delta(1,s)}\ldots P_\alpha^{\Delta(\alpha,s)})$ alors

$\exists\, s' \geq s$ tel que $k_{s'} = k$

La fonction $\tilde{F}$ sera choisie de telle sorte que cette séquence se stabilise au bout d'un nombre fini de termes, c'est-à-dire que :

$$\exists\, \bar{s} \in \mathbb{N} \text{ tel que } \forall s_1, s_2 \geq \bar{s},\ (k_{s_1} = k_{s_2}) \Rightarrow (P^{s_1}_{k_{s_1}} = P^{s_2}_{k_{s_2}})$$

Pour tout $k = 1\ldots\alpha$, on note $\tilde{P}_k$ la valeur commune de tous les $P^s_{k_s}$ tels que $s \geq \bar{s}$ et $k_s = k$.
Alors ([Cousot 78b, Ch. 2.§.4.]), quel que soit l'ordre d'évaluation $k_1\ldots k_s\ldots$ choisi, on a $\tilde{P}_k \supseteq \bar{P}_k$, $\forall k = 1\ldots\alpha$. Cependant la vitesse de convergence et la précision du résultat dépendent de cet ordre d'évaluation.
Pour exprimer la vitesse de convergence de la séquence, celle-ci sera divisée en *étapes*, chaque étape correspondant à un intervalle maximal d'entiers $\{s_1, s_1+1\ldots s_2\}$ tel que

$$(s_1 \leq s \leq s_2 \wedge s_1 \leq s' \leq s_2 \wedge s \neq s') \Rightarrow (k_s \neq k_{s'})$$

Pour construire $\tilde{F}$, [Cousot 77a] propose de définir un opérateur *d'élargissement* ∇ vérifiant les propriétés suivantes :

. $\nabla \in (K(\mathbb{R}^n) \times K(\mathbb{R}^n) \to K(\mathbb{R}^n))$

. $\forall P_1, P_2 \in K(\mathbb{R}^n)$, $P_1 \nabla P_2 \supseteq env\ (P_1, P_2)$

. Pour toute suite $P_0, P_1\ldots P_i\ldots$ de polyèdres, la séquence $Q_0, Q_1,\ldots Q_i\ldots$ définie par $Q_0 = P_0$ et $\forall i \geq 1$, $Q_i = Q_{i-1} \nabla P_i$, n'est pas strictement croissante (" condition de chaîne").

On choisit alors un ensemble W de noeuds de jonctions dans le graphe de programme, appelés *noeuds d'élargissement*, de telle manière que toute boucle du programme contienne au moins un noeud d'élargissement, et que Card(W) soit minimal. Un tel ensemble n'est pas forcément unique ([Cousot 78b, § 4.1.2.0.2.]), mais peut être facilement déterminé automatiquement.

Alors, $\forall k = 1...\alpha$, si N_k est le noeud origine de l'arc a_k :

- $N_k \notin W \Rightarrow \tilde{F}_k = F_k$
- $N_k \in W \Rightarrow \tilde{F}_k(P_1...P_\alpha) = P_k \nabla env(P_{i_1}...P_{i_r})$

où $a_{i_1}...a_{i_r}$ sont les arcs incidents au noeud de jonction N_k.

5.3 - Elargissement

5.3.1. - Le choix de l'opérateur ∇ résultant d'un compromis entre la rapidité et la précision de l'analyse, il n'est justifiable que par expérimentation sur des exemples de programmes. Un exemple trivial d'opérateur satisfaisant les propriétés de l'élargissement consisterait à poser : $\forall P_1, P_2 \in K(\mathbb{R}^n)$, $P_1 \nabla P_2 = \mathbb{R}^n$. Cet opérateur assure une convergence rapide de la séquence, mais avec des résultats totalement imprécis et inutilisables.

5.3.2. - L'idée initiale sur laquelle est fondée la définition de l'élargissement que nous choisirons, est de ne garder comme contraintes de $P_1 \nabla P_2$ que les contraintes de P_1 qui sont vérifiées par P_2. Le problème qui se pose alors est qu'il peut exister plusieurs systèmes de contraintes minimaux différents définissant P_1, et que donc le résultat de $P_1 \nabla P_2$ dépend de la représentation choisie pour P_1, ce qui n'est guère satisfaisant :

Exemple :

$P_1 = \{(x,y) \mid y=0,\ y \le x \le y+1\} = \{(x,y) \mid y = 0,\ 0 \le x \le 1\}$

$P_2 = \{(x,y) \mid y \ge 0,\ x \ge y,\ x+y \le 2\}$

$P_1 \nabla P_2 = \{y \ge 0,\ y \le x\}$ ou $\{y \ge 0,\ x \ge 0\}$

5.3.3. - *Définition*

Soient (A,B,S,R,D), (A',B',S',R',D') les représentations respectives de deux polyèdres P_1 et P_2 de $\mathbb{R}^n$. Soient m_1 le nombre de contraintes de P_1 et m_2 le nombre de contraintes de P_2, en supposant que dans les deux systèmes, toutes les équations apparaissent sous forme de couples d'inéquations opposées.

. Soit M_1 le sous-ensemble de $\{1...m_1\}$ défini par :

$\forall i \in M_1, \forall x \in P_2, A_i x \leq B_i$

. Soit M_2 le sous-ensemble de $\{1...m_2\}$ défini par :
$\forall i \in M_2 \; \exists \; j \in \{1...m_1\}$ tel que
$P_1 = \{x \mid A_k x \leq B_k, \; k \in \{1...j-1,j+1...m_1\}$ et $A'_i x \leq B'_i\}$.

(En d'autres termes, on peut sans modifier P_1 remplacer la j-ième contrainte de P_1 par la i-ième contrainte de P_2).
Alors si $P_1 = \emptyset$ on pose $P_1 \nabla P_2 = P_2$, sinon le polyèdre $P_1 \nabla P_2$ est l'ensemble des solutions du système de contraintes ($A_{M_1} X \leq B_{M_1}$ et $A'_{M_2} X \leq B'_{M_2}$)

5.3.4. - *Remarques*

- L'opération définie ci-dessus revient à convertir le système de contraintes de P_1 en un système équivalent, mais tel que le plus possible de contraintes du nouveau système soient vérifiées par P_2. Ce nouveau système ne dépend pas de la représentation initiale de P_1. On peut donc lui appliquer l'opérateur défini au § 5.3.2. et le résultat de $P_1 \nabla P_2$ ne dépend pas de la représentation de P_1.

- L'ensemble M_1 est l'ensemble des indices des contraintes de P_1 satisfaites par tous les éléments générateurs de P_2. L'ensemble M_2 est l'ensemble des indices des contraintes de P_2 qui, adjointes au système de contraintes de P_1, sont mutuellement redondantes avec une contrainte de P_1 (i.e. la contrainte de P_2 et la contrainte de P_1 sont saturées par le même ensemble d'éléments générateurs de P_1).

- Pour calculer un système générateur de $P_1 \nabla P_2$ (et pour simplifier son système de contraintes), la procédure du § 2.3.2 est appliquée, en profitant du fait que l'on connait un système générateur de P_1 (cf. 2.3.3.).

5.3.5. - *Exemple*

$P_1 = (\{y = 0,\ 0 \leq x \leq 1\},\ S_1 = \{(0,0),(1,0)\},\ R_1 = D_1 = \emptyset)$

$P_2 = (\{y \geq 0,\ x \geq y,\ x+y \leq 2\},\ S_2 = \{(0,0),\ (1,1),\ (2,0)\},\ R_2 = D_2 = \emptyset).$

Dans le tableau suivant, les lignes correspondent aux contraintes de P_1 et de P_2, les colonnes aux éléments générateurs de P_1. Une * à l'intersection d'une ligne et d'une colonne signifie que l'élément correspondant à la colonne sature la contrainte correspondant à la ligne.

		$\binom{0}{0}$	$\binom{1}{0}$
Contraintes de P_1	$y = 0$	*	*
	$0 \leq x$	*	
	$x \leq 1$		*
Contraintes de P_2	$y \geq 0$	*	
	$x \geq y$	*	
	$x+y \leq 2$		

La contrainte $x \geq 0$ de P_1 est satisfaite par P_2.
Les contraintes $y \geq 0$ et $x \geq y$ de P_2 sont mutuellement redondantes avec la contrainte $x \geq 0$ de P_1.

$$P_1 \nabla P_2 = \{(x,y) \mid 0 \leq x,\ 0 \leq y \leq x\}$$

$x \geq 0$ est redondante dans ce système.

5.3.6. - *Proposition*

L'opérateur ∇ satisfait les hypothèses de l'élargissement (cf. 5.2).

Démonstration

- Toute contrainte de $P_1 \nabla P_2$ est satisfaite par P_1 et P_2, donc $P_1 \nabla P_2$ est un polyèdre convexe contenant P_1 et P_2. Donc $P_1 \nabla P_2$ contient le plus petit polyèdre convexe contenant P_1 et P_2, qui est $env(P_1,P_2)$.

- Soit $(P_o,P_1,\ldots P_i,\ldots)$ une suite infinie de polyèdres convexes.
  Soit $(Q_o,Q_1,\ldots Q_i,\ldots)$ la séquence définie par $Q_o = P_o$, et $\forall i \geq 1$, $Q_i = Q_{i-1} \nabla P_i$.
  Soit q le plus petit entier tel que $\forall i \geq 0$, $\dim(Q_i) \leq q$. Alors $\exists\, i_o \geq 0$ tel que $\forall i \geq i_o$, $\dim(Q_i) = q$, puisque les Q_i forment une séquence croissante. Donc, $\forall i \geq i_o$, Q_i admet un seul système de contraintes dans $\mathbb{R}^q$, et $\forall i \geq i_o$, le système de contraintes de Q_i dans $\mathbb{R}^q$ est obtenu à partir de celui de Q_{i_o} par suppression de certaines contraintes. Q_{i_o} ayant un nombre fini de contraintes, on ne peut obtenir qu'un nombre fini de polyèdresQ_i différents à partir de Q_{i_o}. Donc $\exists\, i_1$ tel que $\forall i \geq i_1$, $Q_i = Q_{i_1}$ et ∇ satisfait à la condition de chaîne.

5.3.7. - Obtention de résultats plus précis

Disposant maintenant d'un opérateur d'élargissement qui nous assure de la convergence de la séquence croissante d'approximations en un nombre fini d'itérations, on peut choisir de n'appliquer cet opérateur qu'à partir de la k_o-ième étape de la séquence. Celle-ci convergera toujours, d'autant plus lentement que k_o sera choisi grand, mais vers un résultat d'autant plus précis ([Cousot 78b], 4.1.2.0.7.). En fait, il apparait que si l'on ne fait aucune approximation aux noeuds d'élargissement, le temps d'exécution de l'analyse croit exponentiellement avec k_o (intuitivement, il existe p^{k_o} chemins pour parcourir k_o fois une boucle simple présentant p chemins). On peut aussi, jusqu'à l'étape k_o appliquer un opérateur d'approximation évitant cette "explosion" exponentielle, mais ne vérifiant pas nécessairement la condition de chaîne. A titre d'exemple, nous avons expérimenté un opérateur ∇_λ , tiré de ∇ par affaiblissement du concept de redondance mutuelle, en posant :

$j \in M_2 \iff \exists\, i \in \{1...m_1\}$ tel que le nombre d'éléments générateurs de P_1 saturant $A_i X \leq B_i$ mais ne saturant pas $A'_j X \leq B'_j$ est au plus égal à λ .

Il est clair que $V_o = V$.
On peut imaginer de laisser l'utilisateur du système jouer sur les paramètres k_o et λ selon la précision qu'il désire et le prix qu'il est prêt à payer.
Dans tous les exemples que nous donnerons, on prendra $k_o = \lambda = 0$.

5.4. - Calcul de la séquence croissante d'approximations

Il nous reste à déterminer la suite $k_1 \ldots k_s \ldots$ dans l'ordre de laquelle les contextes vont être calculés.

5.4.1. - Ordre d'évaluation des contextes

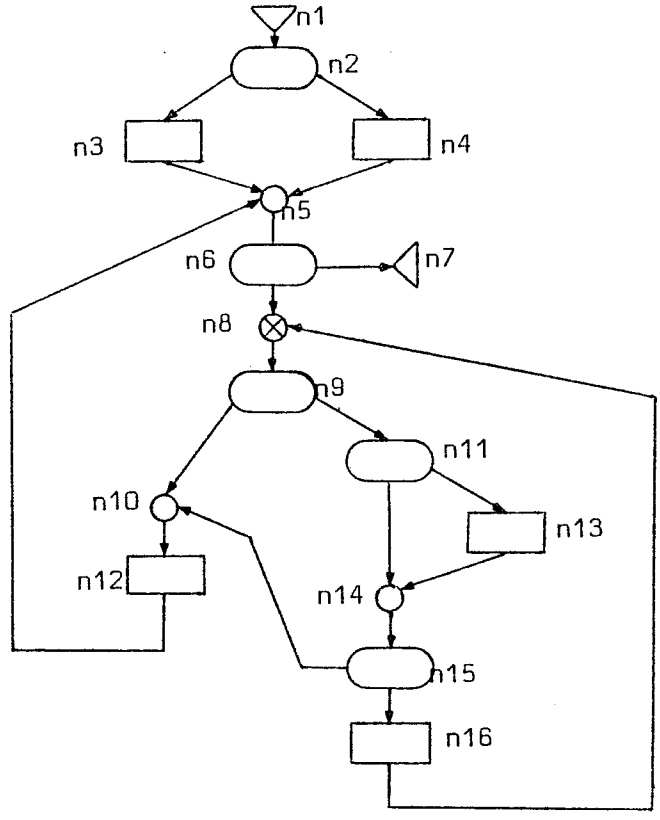
L'ordre d'évaluation que nous avons adopté ne prétend pas être optimal.
Il est cependant fondé sur les remarques suivantes :

5.4.1.1. - On a intérêt à attendre pour calculer un contexte P_k, qu'un nombre maximum des contextes dont dépend P_k aient été recalculés depuis le dernier calcul de P_k. En particulier, si aucun de ces contextes n'a changé depuis le dernier calcul de P_k, il est parfaitement inutile de recalculer P_k.

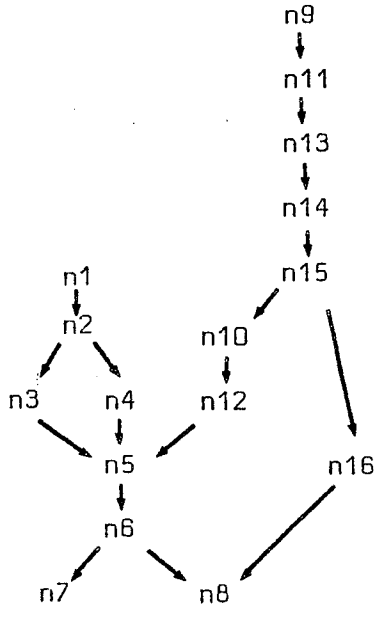
5.4.1.2. - S'il existe un chemin du noeud n_1 au noeud n_2 et s'il n'en existe pas de n_2 à n_1, alors on doit attendre pour traiter n_2, d'avoir obtenu l'information définitive sur les arcs issus de n_1.

5.4.1.3. - L'élargissement, qui fait perdre beaucoup d'information, doit être effectué le moins souvent possible.

Si n_1 et n_2 sont des noeuds du graphe, on note $n_1 \; A \; n_2$, s'il existe un chemin de n_1 à n_2, et $n_1 \; R \; n_2$ s'il existe un chemin de n_1 à n_2 ne contenant pas de noeud d'élargissement, sauf éventuellement n_2. Puisque toute boucle du graphe contient au moins un noeud d'élargissement, la relation R est antisymétrique. Il en résulte que R est un ordre partiel sur les noeuds du graphe.


n1
n2
n3
n4
n5
n6
n7
n8
n9
n11
n10
n13
n12
n14
n15
n16

a


n9
n11
n13
n14
n15
n1
n2
n10
n3
n4
n12
n5
n16
n6
n7
n8

b

Figure 5.1

Alors, la remarque 5.4.1.1. implique qu'après le calcul d'un terme de la séquence il n'est intéressant de calculer qu'un sous-ensemble des contextes. 5.4.1.1. et 5.4.1.3. suggèrent de calculer un contexte de ce sous-ensemble qui soit minimal pour la relation R . Enfin, il résulte de 5.4.1.2. que l'on peut traiter les composantes fortement connexe du graphe une à une, selon l'ordre partiel induit sur ces composantes par la relation A (rappelons qu'on appelle *composante fortement connexe* d'un graphe G un ensemble C de noeuds de G tels que $(n_1 \in C) \Rightarrow (\forall n_2 \in G.\ (n_1 A n_2 \text{ et } n_2 A n_1) \Leftrightarrow (n_2 \in C))$.
On construit alors un ordre total $\leq$ sur les noeuds du graphe, par un tri topologique :

$$n_1 \ R \ n_2 \Rightarrow n_1 \leq n_2$$

et

$$(n_1 \ A \ n_2 \text{ et } n_2 \ A \ n_1) \Rightarrow n_1 \leq n_2$$

Exemple
On considère le graphe de programme de la figure 5.1.a. Le noeud d'élargissement est n_8. L'ordre partiel R est décrit par la figure 5.1.b., les noeuds minimaux étant n_1 et n_9, les maximaux n_7 et n_8. Les noeuds sont totalement ordonnés comme suit :

$$n_1 \leq n_2 \leq n_3 \leq n_4 \leq n_9 \leq n_{11} \leq n_{13} \leq n_{14} \leq n_{15} \leq n_{10} \leq n_{12} \leq n_5 \leq n_6 \leq n_{16} \leq n_8 \leq n_7$$

5.4.2. - Algorithme d'analyse

Pour initialiser le calcul, on associe à tous les arcs du graphe le contexte vide, et on met le noeud d'entrée en attente de traitement.
Un pas de l'algorithme consiste à choisir un noeud en attente de traitement et à traiter ce noeud.S'il n'y a pas de noeud en attente, l'analyse est terminée, sinon on choisit toujours *le noeud le plus petit*, pour l'ordre total $\leq$, parmi les noeuds en attente.
Traiter un noeud n consiste pour chaque arc a issu de n, à calculer le contexte associé à a en fonction des contextes des arcs incidents à n, puis, *si le nouveau contexte est strictement plus grand* que le contexte précédemment associé à a, à mettre en attente le noeud extrémité de a.

Exemple

Donnons l'ordre de traitement des noeuds du graphe de programme de l'exemple précédent (fig 5.1) sans tenir compte d'une éventuelle convergence de la séquence :

pas	noeud traité	noeuds en attente
initialisation		n_1
1	n_1	n_2
2	n_2	n_3, n_4
3	n_3	n_4, n_5
4	n_4	n_5
5	n_5	n_6
6	n_6	n_7, n_8
7	n_8	n_7, n_9
8	n_9	n_7, n_{10}, n_{11}
9	n_{11}	$n_7, n_{10}, n_{13}, n_{14}$
10	n_{13}	n_7, n_{10}, n_{14}
11	n_{14}	n_7, n_{10}, n_{15}
12	n_{15}	n_7, n_{10}, n_{16}
13	n_{10}	n_7, n_{16}, n_{12}
14	n_{12}	n_7, n_{16}, n_5
15	n_5	n_7, n_{16}, n_8
16	n_{16}	n_7, n_8

5.5 - Séquence Décroissante d'Approximations Supérieures

La solution $\tilde{P}$ obtenue comme limite de la séquence croissante d'approximations est telle que

$$\forall k = 1...\alpha \ , \ \tilde{P}_k \geq F_k(\tilde{P}_1...\tilde{P}_\alpha)$$

et

$$\tilde{P}_k \geq \bar{P}_k$$

De l'isotonie des F_k il résulte que

$$F_k(\tilde{P}_1...\tilde{P}_\alpha) \geq \bar{P}_k$$

Donc $F_k(\tilde{P}_1...\tilde{P}_\alpha)$ est une approximation correcte de la plus petite solution $\bar{P}_k$. Il en résulte que si $\exists\ k = 1...\alpha$ tel que $\tilde{P}_k \neq F_k(\tilde{P}_1...\tilde{P}_\alpha)$, on améliore la précision des résultats en continuant le calcul de la séquence en appliquant F à la place de $\tilde{F}$ (c'est-à-dire qu'on n'effectue plus d'élargissement). Contrairement à la séquence croissante, tous les termes de cette séquence décroissante sont des solutions correctes. Pour éviter une itération infinie [Cousot 76] propose d'accélérer sa convergence de manière analogue à l'élargissement dans la séquence croissante. Nous nous bornons à limiter la longueur de la séquence à k_1 étapes, k_1 étant un nouveau paramètre.
Dans la pratique, cela signifie que lorsque la séquence croissante de contextes, relative à une composante fortement connexe du graphe, a convergé, on effectue encore k_1 itérations en considérant les noeuds d'élargissement comme des noeuds de jonction ordinaires. Soit la séquence converge avant la k_1-ième itération (vers une solution qui n'est pas forcément la plus petite), soit on se contente d'une solution approchée.

5.6. - Exemples d'Analyse en Avant

Tous les exemples que nous allons présenter ont été traités automatiquement par notre analyseur. Pour chaque exemple nous donnerons les indications relatives à la vitesse de convergence (nombre d'étapes de la séquence), et le temps d'exécution de l'analyse automatique.

5.6.1. - Séquence Croissante

Au programme représenté par la figure 5.2, correspond le système d'équations suivant :

$P_1 = \mathbb{R}^2$
$P_2 = affect\ (P_1,\ I:=0)$
$P_3 = affect\ (P_2,\ J:=0)$
$P_4 = env\ (P_3, P_{10})$
$P_5 = P_4$
$P_6 = P_4$ } test non linéaire
$P_7 = affect\ (P_5,\ I:=I+2)$
$P_8 = affect\ (P_7,\ J:=J+1)$
$P_9 = affect\ (P_6,\ I:=I+4)$
$P_{10} = env(P_8, P_9)$

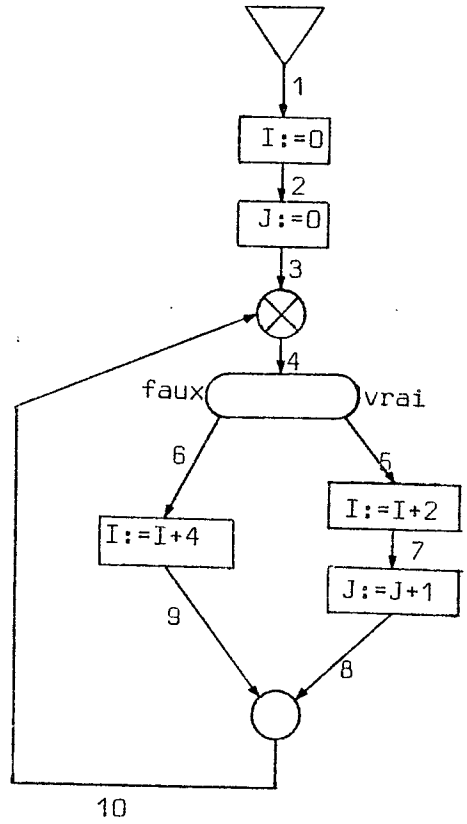


Figure 5.2

A l'initialisation, on pose $P_i^0 = \emptyset$, $\forall i=1...10$

1ère Etape

$P_1^1 = \mathbb{R}^n$
$P_2^1 = affect\ (P_1^1,\ I:=0) = \{I=0\}$
$P_3^1 = affect\ (P_2^1,\ J:=0) = \{I=0,\ J=0\}$
$P_4^1 = P_4^0 \nabla env(P_3^1, P_{10}^0) = P_3^1 = \{I=0,\ J=0\}$
$P_5^1 = P_6^1 = P_4^1$
$P_7^1 = affect\ (P_5^1,\ I:=I+2) = \{I=2,\ J=0\}$
$P_8^1 = affect\ (P_7^1,\ J:=J+1) = \{I=2,\ J=1\}$
$P_9^1 = affect\ (P_6^1,\ I:=I+4) = \{I=4,\ J=0\}$
$P_{10}^1 = env\ (P_8^1, P_9^1) = \{I+2J=4\ ,\ 2 \leq I \leq 4\}$ (fig. 5.3.a.)

2ème étape

$P_4^2 = P_1^1 \nabla env(P_3^1, P_{10}^1)$

$env\ (P_3^1, P_{10}^1) = \{I+2J \leq 4,\ I-2J \geq 0,\ J \geq 0\}$ (fig. 5.3.b)

$P_4^2 = \{J \geq 0,\ I-2J \geq 0\}$ (fig. 5.3.c)

$P_8^2 = affect\ (affect(P_4^2,\ I:=I+2),\ J:=J+1)$

$= \{J \geq 1,\ I-2J \geq 0\}$ (fig. 5.3.d)

$P_9^2 = affect\ (P_4^2,\ I:=I+4)$

$= \{J \geq 0,\ I-2J \geq 4\}$ (fig. 5.3.e)

$P_{10}^2 = env\ (P_8^2, P_9^2)$

$= \{J \geq 0,\ I-2J \geq 0,\ I+2J \geq 4\}$ (fig. 5.3.f)

$env\ (P_3^1, P_{10}^2) = \{J \geq 0,\ I-2J \geq 0\}$

$= P_4^2$

La séquence croissante a convergé vers une solution exacte. L'analyse s'arrête avec les résultats suivants :

P_1 = aucune information

$P_2 = \{I=0\}$

$P_3 = \{I=0,\ J=0\}$

$P_4 = \{J \geq 0\ ,\ I-2J \geq 0\} = P_5 = P_6$

$P_7 = \{J \geq 0,\ I-2J \geq 2\}$

$P_8 = \{J \geq 1,\ I-2J \geq 0\}$

$P_9 = \{J \geq 0,\ I-2J \geq 4\}$

$P_{10} = \{J \geq 0,\ I-2J \geq 0,\ I+2J \geq 4\}$

La séquence croissante converge en 2 étapes vers une solution exacte

Temps d'éxécution : 0,22 secondes.

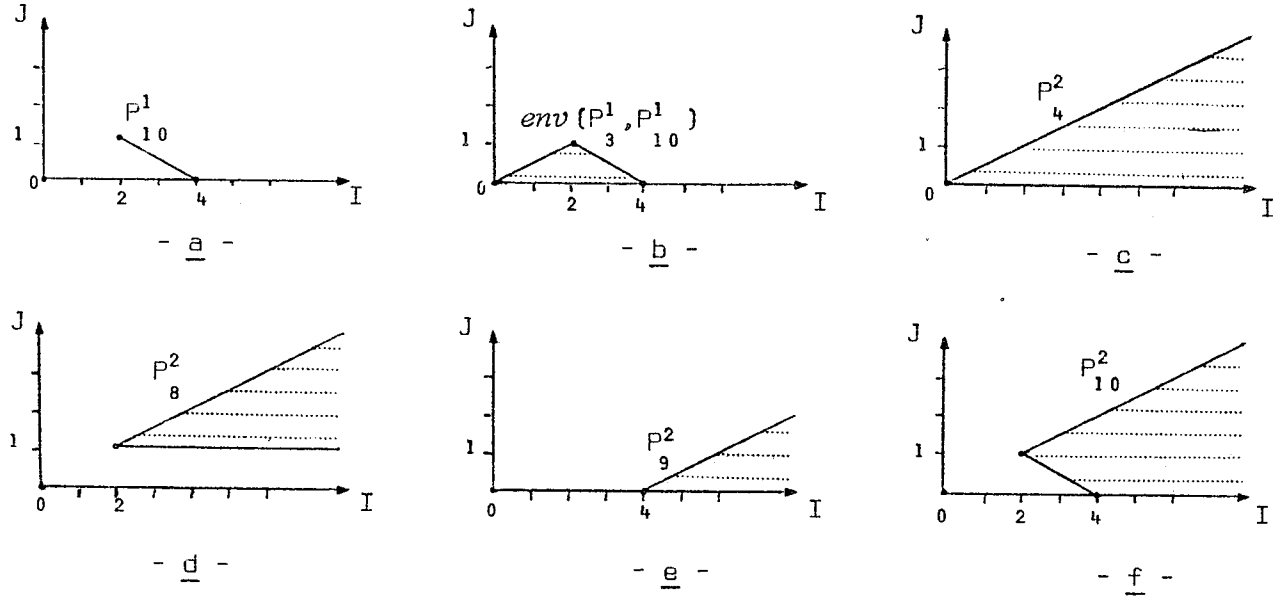


figure 5.3.

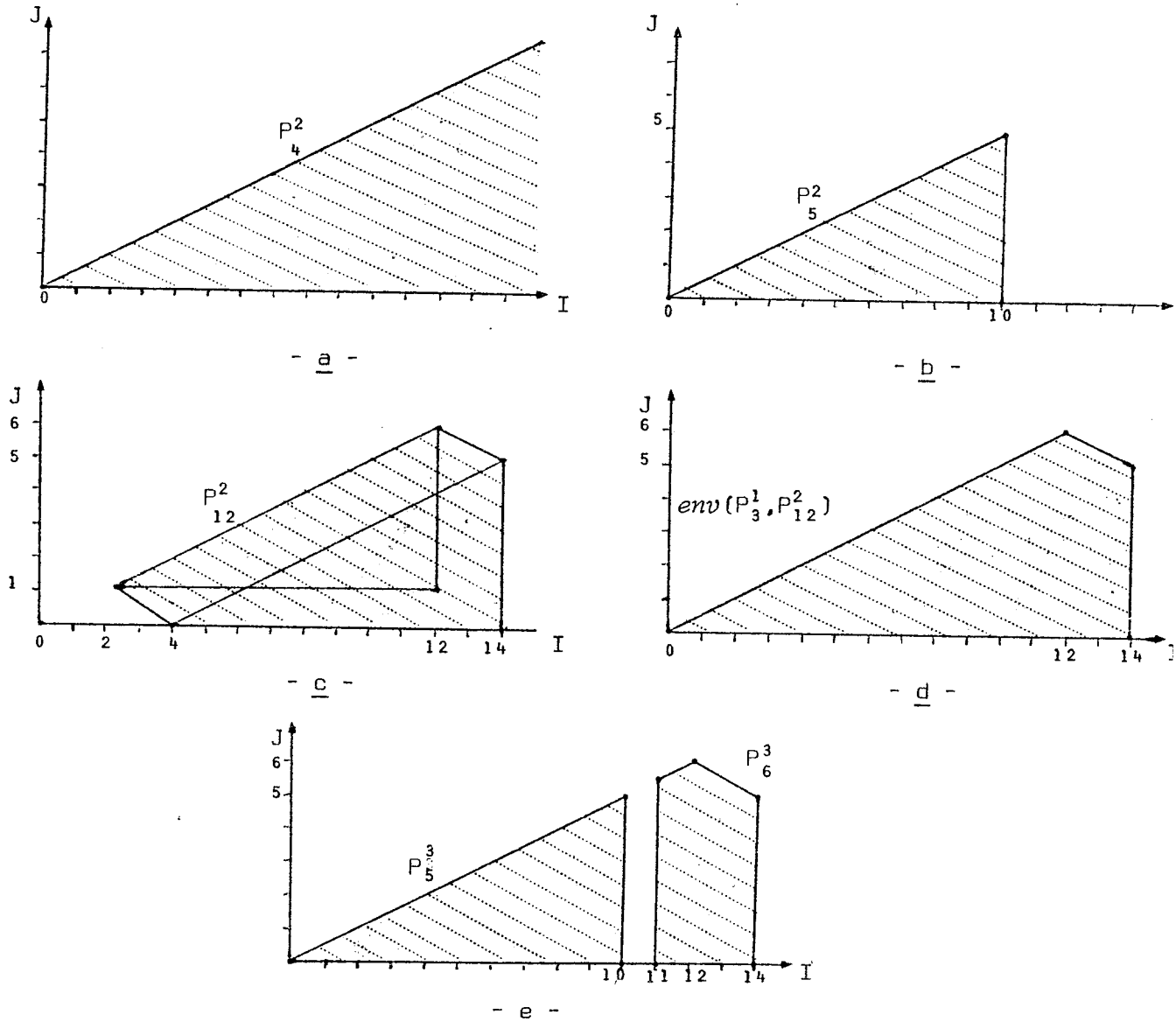


figure 5.4.

5.6.2. - Séquences Croissante et Décroissante

L'exemple précédent est enrichi d'un test de sortie (Fig. 5.4.). Le nouveau système d'équations s'écrit, après simplifications:

P_3 = affect(affect(R^n, I:=0), J:=0)
P_4 = P_4 ∇ *env*(P_3,P_{12})
P_5 = P_4 $\wedge$ {I≤10}
P_6 = P_4 $\wedge$ {I≥11}
P_{10}= *affect* (*affect* (P_5, I:=I+2), J:=J+1)
P_{11}= *affect* (P_5, I:=I+4)
P_{12}= *env* (P_{10},P_{11})

1ère étape

P_3^1 = P_4^1 = {I=0, J=0}
P_5^1 = P_4^1
P_6^1 = ∅
P_{10}^1= {I=2,J=1}
P_{11}^1= {I=4,J=0}
P_{12}^1= {I+2J=4, 2≤I≤4}

2ème étape

P_4^2 = P_4^1 ∇ env(P_3^1,P_{12}^1) = {J≥0, I-2J≥0} (fig. 5.4.a)
P_5^2 = P_4^2 $\wedge$ I≤10 = {J≥0, I-2J≥0, I≤10} (fig. 5.4.b)
P_6^2 = P_4^2 $\wedge$ I≥11 = {J≥0, I-2J≥0, I≥11}
P_{10}^2= *affect*(*affect*(P_5^2, I:=I+2), J:=J+1)
= {J≥1, I-2J≥0, I≤12}
P_{11}^2= *affect*(P_5^2, I:=I+4) = {J≥0, I-2J≥4, I≤14}
P_{12}^2= *env*(P_{10}^2,P_{11}^2) = {J≥0, I-2J≥0, I≤14, 4≤I+2J≤24} (fig. 5.4.c)
env(P_3^1,P_{12}^2) = {J≥0, I-2J≥0, I≤14, I+2J≤24} (fig. 5.4.d)
env(P_3^1,P_{12}^2) ⊆ P_4^2

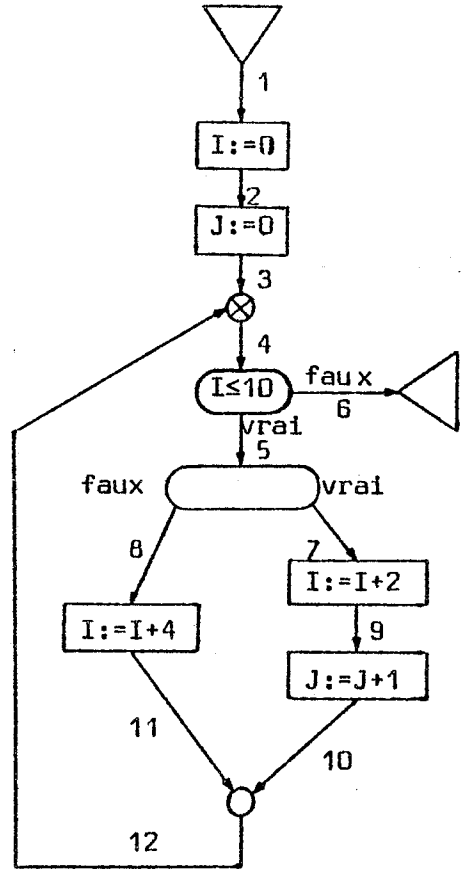


Figure 5.5.

La séquence croissante a convergé. On amorce alors une séquence décroissante à partir de la solution atteinte :

3ème étape

$P_4^3 = env(P_3^1, P_{12}^2) = \{J \geq 0,\ I-2J \geq 0,\ I \leq 14,\ I+2J \leq 24\}$

$P_5^3 = P_4^3 \wedge I \leq 10 = \{J \geq 0,\ I-2J \geq 0,\ I \leq 14,\ I+2J \leq 24,\ I \leq 10\}$

$= \{J \geq 0,\ I-2J \geq 0,\ I \leq 10\}$ (fig. 5.4.e)

$= P_5^2$

$P_6^3 = P_4^3 \wedge I \geq 11 = \{J \geq 0,\ I-2J \geq 0,\ 11 \leq I \leq 14,\ I+2J \leq 24\}$ (fig. 5.4.e)

La séquence décroissante a convergé vers une solution exacte.

- La séquence croissante converge en deux étapes.
- La séquence décroissante atteint une solution exacte en une étape
- temps d'exécution : 0,37 secondes.

Les deux exemples précédents sont assez simples pour pouvoir être analysés à la main. Notons que les résultats obtenus ne sont pas triviaux, et n'apparaissent qu'après un examen attentif des programmes.

Pour des programmes plus importants, il est impossible de donner ici le détail des calculs et de représenter graphiquement les contextes (espace à plusieurs dimensions). Nous ne donnons, pour les exemples suivants, que les résultats obtenus.

5.6.3. - Recherche Dichotomique

L'exemple suivant est la recherche dichotomique d'un élément R dans un tableau trié T à N éléments réels ($N \geq 1$). La première version de cet algorithme est celle de [Knuth 73]. On n'analyse que les relations entre les variables entières du programme (fig. 5.6.) :

Le système d'équations est le suivant :

P_1 = (N≥1)

P_3 = *affect*(*affect*(P_1, L:=1), U:=N)

P_4 = P_4 ∇ *env*(P_3,P_{10},P_{13})

P_5 = P_4 ∧ (L≤U)

P_6 = P_4 ∧ (L≥U+1)

P_7 = *affect*(P_5, I:=(L+U)div2)

P_{10} = *affect*(P_7, U:=I-1)

P_{11} = *affect*(P_7, L:=I+1)

P_{12} = P_7

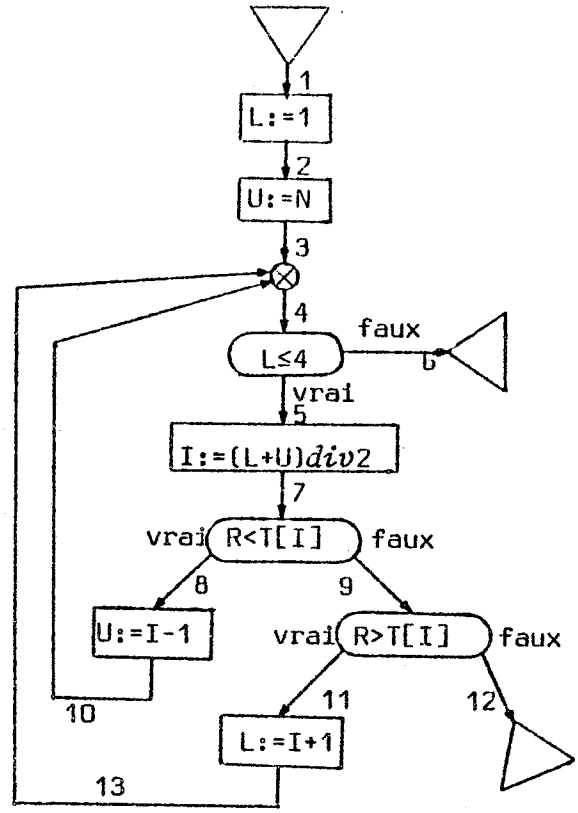


Figure 5.6

Remarques :

- On exploite la spécification d'entrée, qui est que la borne supérieure du tableau est supérieure à sa borne inférieure (cf. 4.2.1.).
- On connait des expressions linéaires approchant l'expression (L+U) *div* 2 (cf. 4.2.2.1.1.) : L+U-1 ≤ 2 [(L+U)div2] ≤ L+U
- Les tests R < T[I] et R > T[I] sont non linéaires et n'apportent pas d'information.

Les résultats sont les suivants :

P_3 = {L=1, U=N, N≥1}

P_4 = {N≥1, L≥1, 2L≤2U+3, U≤N, 3L≤2U+N+3}

P_6 = {N≥1, L≥1, 2L≤2U+3, 3L≤2U+N+3,U+1≤L}

P_7 = P_8 = P_9 = P_{11} = P_{12} = {1≤L≤U≤N, 2I≤L+U≤2I+1}

P_{10} = {1≤L≤N, 2L-1≤2I≤L+N, I=U+1}

P_{13} = {1≤U≤N, U≤2I≤2U, L=I+1}

La séquence croissante converge en deux étapes.
La séquence décroissante converge vers une solution en une étape
Temps d'exécution : 1,69 secondes.

A partir du résultat de P_7, il est facile de montrer que les accès à l'élément T[I] sont corrects, c'est-à-dire que 1≤I≤N. Nous reviendrons sur l'utilisation des résultats au chapitre 7.

Une autre version de l'algorithme de recherche dichotomique a été analysée : c'est celle qui est utilisée dans le compilateur PL/0 ([Wirth 76b]).
Voici les résultats de l'analyse :

```
I:=1 ; J:=N ; {I=1, J=N, N≥1}
faire {1≤I≤J≤N}
    K:=(I+J)div 2 ; {1≤I≤J≤N, 2K≤I+J≤2K+1}
    si R≤T[K] alors J:=K-1 {1≤I≤N, 2I-1≤2K≤I+N, K=J+1}
    finsi ; {1≤I≤N, 2I-1≤2K≤I+N, K-1≤J≤N, I+J≤2K+1}
    si R≥T[K] alors I:=K+1 {N≥1, K-1≤J≤2K, 1≤2K, K≤N, J≤N, I=K+1}
    finsi ; {1≤I, 1≤N, J≤N, 1≤2K≤2N, 2I≤2K≤I+N, J≤2K≤2J+2}
jusqua I>J ; {1≤I, I-1≤K≤J+1≤I, 1≤2K≤2N, N≥1}
```

La séquence croissante converge en deux étapes vers une solution exacte
Temps d'exécution : 2,8 secondes.

5.6.4. - Algorithmes de Tri [Knuth 73]

5.6.4.1. - Tri par insertion

```
pour J:=2 à N faire {2≤J≤N}
   I:=J-1 ; K:=T[J];
   tantque(I≥1) et (K≤T[I]) faire {1≤I, I+1≤J≤N}
      T[I+1] := T[I] ; I:=I-1 ;
   fait ; {0≤I, I+1≤J, 2≤J≤N}
   T[I+1] := K ;
fait ; {2≤J, N≤J}
```

La séquence croissante converge en deux étapes
La séquence décroissante converge en une étape vers une solution exacte
Temps d'exécution : 0,75 secondes.

5.6.4.2. - Tri par Sélection

```
pour J:=N pas-1 à 2 faire {2≤J≤N}
   I:=J ;
   pour K:=J-1 pas-1 à 1 faire {1≤K≤I-1, I≤J≤N}
        si T[I] < T[K] alors I:=K finsi ; {1≤K≤I≤J≤N, K+1≤J}
   fait ; {K=0, 1≤I≤J≤N, J≥2}
   R:=T[I] ; T[I] := T[J] ; T[J]:=R ;
fait ; {J≤1, J≤N}
```

La séquence croissante converge en trois étapes.
La séquence décroissante atteint une solution en une étape
Temps d'exécution : 1,86 secondes.

5.6.4.3. - Tri par "remontées des bulles"

```
K:=N ;
tantque K≥1 faire {1≤K≤N}
   I:=0 ;
   pour J:=1 a K-1 faire {0≤I, I+1≤J, J+1≤K≤N}
      si T[J]>T[J+1] alors
         W:=T[J] ; T[J] :=T[J+1]; T[J+1]:=W ;
         I:=J {1≤J, J=I, J+1≤K≤N}
      finsi ; {0≤I≤J, 1≤J, J+1≤K≤N}
   fait ;  {0≤I≤J-1, J=K≤N}
   K:=I ;  {0≤I = K≤J-1, J≤N}
fait  {K≤N, K≤0}
```

La séquence croissante converge en deux étapes vers une solution.
Temps d'exécution : 0,89 secondes.

5.6.4.4. - "Heapsort"

```
procedure HEAPSORT (entier valeur N ; réel tableau [1..N]T) ;
début entier L,R,I,J ; réel K ; {N≥1}
    L:=(N div 2)+1 ; R:=N ;  {N≥1, N+1≤2L≤N+2, R=N}
    si L≥2 alors L:=L-1 ; K:=T[L] {2L≤N≤2L+1, R=N, L≥1}
       sinon K:=T[R]; T[R]:= T[1] ; R:=R-1 {L=N=R+1=1}
    finsi ;  {L≥1, R≤N, 2L+R≤2N, 3N≤2L+2R+1}
    tantque R≥2 faire {L≥1, 2≤R≤N, 2L+R≤2N}
       I:=L ; J:=2*I ;
       tantque J≤R faire  {R≤N, 1≤L≤I, 2I≤R, J=2I}
          si J≤R-1 alors si T[J]<T[J+1] alors J:=J+1 finsi finsi ;
         {1≤L≤I, 2I≤J≤2I+1, J≤R≤N}
          si K≥T[J] alors exit finsi ;
          T[I]:=T[J] ; I:=J ; J:=2*J ; {L≥1, 2L≤I≤R≤N, 2I=J}
       fait ; {1≤L≤I, 2I≤J≤2I+1, R≤N, J+2L+R≤2I+2N,
               J+2≤2I+R, 2J+1≤3I+L+R, 2J≤3I+N}
       T[I]:= K ;
       si L≥2 alors L:=L-1 ; K:=T[L]  {2≤R≤N, L≥1, 2L+R+2≤2N}
          sinon K:=T[R]; T[R]:=T[1] ; R:=R-1 {L=1, 1≤R≤N-1}
       finsi ;
       {1≤R≤N, L≥1, 2L+2R+2≤3N, 2L+R+1≤2N}
    fait ;
    T[1]:= K ; {L≥1, 0≤R≤1, 2L+R≤2N}
fin ;
```

La séquence croissante converge en deux étapes.

La séquence décroissante atteint une solution en une étape.

Temps d'exécution : 5,8 secondes.

5.7 - Conclusions sur l'analyse en avant

Les exemples proposés illustrent la richesse des résultats de l'analyse en avant. En effet, tous les accès aux éléments de tableaux dans les programmes de tri présentés peuvent être validés à partir des contextes calculés.
D'autre part, ces exemples justifient les mécanismes d'approximation adoptés : d'une part la séquence croissante converge vite (deux à trois étapes par composante fortement connexe) et d'autre part elle converge vers une solution suffisamment précise pour qu'une étape de la séquence décroissante suffise en général à atteindre une solution qui ne peut plus être améliorée par itération.

6 - ANALYSE APPROCHEE EN ARRIERE DES PROGRAMMES

6.1 - Introduction [Cousot 78b, 3.5 & 5.6]

Nous allons chercher maintenant un autre type de résultats : étant donné un programme π et une spécification de sortie ψ, nous voulons caractériser l'ensemble des états à partir desquels il est possible d'atteindre ψ . En d'autres termes le contexte associé à l'arc a sera déduit des contextes associés aux arcs issus du noeud extrémité de a, conformément à la sémantique du langage qui sera cette fois exprimée par les préconditions de [Hoare 69] :
Si a est un arc incident au noeud N et si $Q_1 \ldots Q_k$ sont les contextes associés aux arcs issus de N par l'analyse sémantique en arrière, alors le contexte P associé à a doit être tel que :

$$P \supseteq \alpha(pré(\gamma(Q_1)\ldots\gamma(Q_k),N,a))$$

où $pré(P_1 \ldots P_k,N,a)$ dénote la plus faible précondition sur a impliquant $P_1 \ldots P_k$ sur les arcs successeurs de N.

6.2 - Interprétation en arrière des constructions élémentaires du langage

6.2.1. - Noeud de Sortie

Si l'on connait un polyèdre P de $\mathbb{R}^n$ tel que $\psi(X) \Rightarrow X \in P$, où ψ est la spécification de sortie, alors P sera le contexte associé aux arcs de sortie du programme. Sinon on prendra simplement $P = \mathbb{R}^n$.

6.2.2. - Noeud d'Affectation

Soit Q le contexte associé à l'arc issu d'un noeud d'affectation $X_i := E(X)$. On notera *ar-affect* $(Q, X_i := E(X))$ le polyèdre associé à l'arc incident. La condition de validité s'écrit (en négligeant les conditions de définition de l'expression E(X)) :

$$ar\text{-}affect\ (Q, X_i := E(X)) \supseteq \alpha((X_1 \ldots X_{i-1}, E(X_1 \ldots X_n), \ldots X_n) \in \gamma(Q))$$

6.2.2.1. -Si E est une expression non linéaire, et si l'on connait un système de contraintes P(E) liant $X_1 \ldots X_n$ et $E(X_1 \ldots X_n)$, alors
$ar\text{-}affect\ (Q, X_i := E(X)) = proj(\{(X_1 \ldots X_{i-1}, X_o \ldots X_n) \in Q \wedge (X_o X_1 \ldots X_n) \in P(E), X_o)$
est une interprétation correcte.

Démonstration :

$$\{(X_1 \ldots X_n) \mid (X_1 \ldots X_{i-1}\ E(X_1 \ldots X_n) \ldots X_n) \in \gamma(Q)\} \subseteq$$
$$\{(X_1 \ldots X_n) \mid (X_1 \ldots X_{i-1} X_o \ldots X_n) \in Q \wedge (X_o X_1 \ldots X_n) \in P(E)\} =$$
$$proj\ (\{(X_o \ldots X_n) \mid (X_1 \ldots X_o \ldots X_n) \in Q \wedge (X_o \ldots X_n) \in P(E)\}, X_o).\ \square$$

Il s'agit d'effectuer l'intersection de deux polyèdres dans $\mathbb{R}^{n+1}$, puis d'éliminer X_o du système obtenu. Si le seul polyèdre P(E) connu est égal à $\mathbb{R}^{n+1}$, l'expression ci-dessus se simplifie :
$ar\text{-}affect\ (Q, X_i := E(X)) = proj(Q, X_i)$

6.2.2.2. - Si $E(X) = aX+b$, avec $a_i = 0$ (affectation linéaire non inversible) et si $H = \{X \in \mathbb{R}^n \mid X_i = aX+b\}$
$ar\text{-}affect\ (Q, X_i := E(X)) \supseteq proj(Q \cap H, X_i)$

Démonstration :
Résulte de 6.2.2.1., avec H = P(E).□
On sait calculer les deux représentations de $proj(Q \cap H, X_i)$, aussi prendra-t-on $ar\text{-}affect\ (Q, X_i := E(X)) = proj(Q \cap H, X_i)$.

6.2.2.3. - Si $E(X) = aX+b$ avec $a_i \neq 0$, alors il est correct de prendre :

$$ar\text{-}affect(Q, X_i := E(X)) = affect(Q, X_i := (X_i - \Sigma_{j \neq i}\ a_j\ X_j - b)/a_i$$

Démonstration

Soit $AX \leq B$ le système de contraintes de Q. La condition de validité s'écrit :

$ar\text{-}affect\ (Q, X_i := E(X)) \supseteq$

$\{(X_1 \ldots X_n) \mid \Lambda_{k=1}^{m} A_k^1 X_1 + \ldots + A_k^{i-1} X_{i-1} + A_k^i (\Sigma_{j=1}^{n}\ a_j X_j + b) + \ldots + A_k^n X_n \leq B_k\}$

$= \{\Lambda_{k=1}^{m}\ \Sigma_{j \neq i} (A_k^j + A_k^i\ a_j) X_j + A_k^i\ a_i X_i \leq B_k - A_k^i b\}$

$= \{A'X \leq B'\}$ avec

- $B' = B - b.A^i$
- $\forall j \neq i,\ A^j = A^j + a_j.A^i$
- $A^i = a_i A^i$

On vérifie que $\{A'X \leq B'\} = affect(\{AX \leq B\},\ X_i := a'X + b')$ avec
$b' = -b/a_i$, $a'_i = 1/a_i$, $\forall j \neq i$, $a'_j = -a_j/a_i$.
On sait donc calculer les deux représentations de $ar\text{-}affect(Q, X_i := E(X))$
(cf. 4.2.2.2.1. & 4.2.2.2.3.).

6.2.2.4. - *Remarques*

- Lorsque l'affectation est linéaire, l'interprétation définie ci-dessus ne perd pas d'information.
- La fonction $\lambda Q.ar\text{-}affect\ (Q, X_i := E(X))$ est isotone quelle que soit l'instruction d'affectation.

6.2.3. - Noeud de Test

Soient Q_V et Q_F les contextes associés respectivement aux arcs "vrai" et "faux" d'un noeud de test portant sur la condition C.
Alors $pré(C, Q_V, Q_F) = (Q_V \wedge C) \vee (Q_F \wedge \neg C)$
(on néglige les conditions de définition de l'expression C). Il en résulte la proposition suivante :

6.2.3.1. - *Proposition*

Si P_V et P_F sont les plus petits ensembles convexes tels que $P_V \supseteq \{X \in Q_V | C(X)\}$ et $P_F \supseteq \{X \in Q_F \mid \neg C(X)\}$ alors le plus petit ensemble convexe qu'on puisse associer à l'arc k incident au noeud de test est : $env(P_V, P_F)$.

6.2.3.2. - *Corollaires*

. Si C est une condition non linéaire, $env\ (Q_V, Q_F)$ est un contexte correct pour l'arc k (des études spécifiques selon la forme de C peuvent permettre de trouver un contexte plus précis).

. Si C est une condition linéaire de la forme $aX=b$, alors
$P_V = \{X \in Q_V \wedge aX=b\}$, $P_F = (si\ Q_F \subseteq \{X|aX=b\}\ alors\ \emptyset\ sinon\ Q_F)$.
sont des polyèdres et $env(P_V, P_F)$ est un contexte correct pour k.

. Si C est une condition linéaire de la forme $aX \leq b$, alors,
$P_V = \{X \in Q_V \wedge aX \leq b\}$, $P_F = (si\ Q_F \subseteq \{X|aX \leq b\}\ alors\ \emptyset\ sinon\ \{X \in Q_F \wedge aX \geq b\})$,
sont des polyèdres et $env(P_V, P_F)$ est un contexte correct pour k.

. Si C est une condition linéaire ne portant que sur des variables de types discrets (entier), si tous les coefficients a_i et b sont entiers et si l'on pose $a_o = \min_{i=1..n} (|a_i|)$, $H = \{X|aX=b\}$, $E_1 = \{X|aX \leq b\}$, $E_2 = \{X|aX \not\leq b\}$,
$E'_1 = \{X|aX \leq b-a_o\}$ $E'_2 = \{X|aX \geq b+a_o\}$, alors

1°- si $C = (aX=b)$, $P_V = Q_V \cap H$, $P_F = (si\ Q_F \cap E'_1 = \emptyset$ alors $Q_F \cap E'_2$ sinon si $Q_F \cap E'_2 = \emptyset$ alors $Q_F \cap E'_1$ sinon $Q_F)$;

2°- Si $C = (aX \leq b)$, $P_V = Q_V \cap E_1$, $P_F = Q_F \cap E'_2$;

3°- Dans tous les cas P_V et P_F sont des polyèdres et $env(P_V, P_F)$ est un contexte correct pour k.

6.2.4. - Noeud de Jonction

Soient $k_1 \ldots k_p$ les arcs incidents à un noeud de jonction et Q le contexte associé à l'arc issu de ce noeud. Alors la plus faible précondition sur l'arc k_i, impliquant $\gamma(Q)$ est :

$$pré(\gamma(Q), k_i) = \gamma(Q)$$

Il en résulte que $\forall i=1 \ldots p$, le contexte Q_{k_i} associé à l'arc k_i est :

$$Q_{k_i} = \alpha(\gamma(Q)) = Q.$$

6.2.5. - Propagation en arrière des conditions de cohérence

Si l'on tient compte dans les contextes précédant l'évaluation d'une expression E, des conditions nécessaires à la bonne définition de E (par exemple : limites de sur- et sous- débordement des nombres, diviseurs différents de zéro, paramètres d'une racine carrée ou d'un logarithme positifs, indices de tableaux entre les bornes...), les contextes obtenus au terme de l'analyse en arrière sont alors des propriétés nécessaires pour qu'aucune de ces conditions de cohérence ne soit violée. Notons qu'en affectant un contexte non vide aux arcs de sortie (cf. 6.2.1.) on propage implicitement une condition de terminaison du programme. Les contextes obtenus sont donc tels que si au cours d'une exécution du programme, il advient que les valeurs des variables se trouvent à l'extérieur du contexte en arrière, alors on est sûr, à l'avance, que soit une condition de cohérence sera violée, soit la spécification de sortie ne sera pas satisfaite, soit le programme ne terminera pas.

6.3. - Evaluation en Arrière

Comme pour l'évaluation en avant, l'analyse en arrière se compose d'une séquence croissante, où sur tout arc issu d'un noeud d'élargissement, le contexte est calculé en élargissant la valeur précédemment trouvée par la nouvelle valeur, suivie éventuellement d'une séquence décroissante sans élargissement.
L'ordre d'évaluation des contextes est strictement inverse de celui utilisé lors de l'évaluation en avant.

6.4 - *Exemple*

Considérons la procédure de tri d'un tableau T[1...N] par "remontée des bulles", telle que la donne [Manna 74] , et donnons les deux systèmes d'équations, sans tenir compte des conditions de cohérence :

```
    procédure TRI(réel tableau T[1..N], entier N) ;
    début entier I,J ; réel  S ;
1.      I:=N ;
2.      tantque I≠0 faire
3.         J:=0 ;
4.         tantque J≠I faire
5.             si T[J]>T[J+1] alors
6.                S:=T[J] ; T[J]:=T[J+1]; T[J+1]:=S ;
7.             finsi ;
8.             J:=J+1 ;
9.         refaire ;
10.        I:=I-1 ;
11.     refaire ;
12. fin ;
```

Système en Avant

$P_1 = \mathbb{R}^3$
$P_2 = affect\ (P_1,\ I:=N)$
$P_3 = faux\ (env\ (P_2, P_{11}),\ I=0)$
$P_4 = affect\ (P_3,\ J:=0)$
$P_5 = faux\ (env\ (P_4, P_9),\ I=J)$
$P_9 = affect\ (P_5,\ J:=J+1)$
$P_{10} = vrai\ (env\ (P_4, P_9),\ I=J)$
$P_{11} = affect\ (P_{10},\ I:=I-1)$
$P_{12} = vrai\ (env\ (P_2, P_{11}),\ I=0)$

Système en arrière

$Q_1 = ar\text{-}affect\ (Q_2,\ I:=N)$
$Q_2 = env\ (Q_3 \wedge I\neq 0,\ Q_{12} \wedge I=0)$
$Q_3 = ar\text{-}affect\ (Q_4,\ J:=0)$
$Q_4 = env\ (Q_5 \wedge I\neq J,\ Q_{10} \wedge I=J)$
$Q_5 = ar\text{-}affect\ (Q_9,\ J:=J+1)$
$Q_9 = env\ (Q_5 \wedge I\neq J,\ Q_{10} \wedge I=J)$
$Q_{10} = ar\text{-}affect\ (Q_{11},\ I:=I-1)$
$Q_{11} = env\ (Q_3 \wedge I\neq 0,\ Q_{12} \wedge I=0)$
$Q_{12} = \mathbb{R}^3$

Résolution du système en arrière

Initialisation : $Q_i^0 = \emptyset$, i=1..12

1ère étape

$Q_{12}^1 = R^3$

$Q_{11}^1 = Q_2^1 = \{I=0\}$

$Q_{10}^1 = \{I=1\}$

$Q_9^1 = Q_4^1 = \{I=J=1\}$

$Q_5^1 = \{I=1, J=0\}$

$Q_3^1 = proj(\{I=J=1,\ J=0\},\ J)$

$= proj(\emptyset, J) = \emptyset$

2ème étape

$Q_9^2 = Q_4^2 = Q_9^1 \ \nabla\ env(Q_5^1 \wedge I \neq J,\ Q_{10}^1 \wedge I=J)$

$= Q_9^1 \nabla \{I=1,\ 0 \leq J \leq 1\}$

$= \{I=1,\ J \leq 1\}$

$Q_5^2 = \{I=1,\ J \leq 0\}$

$Q_3^2 = \{I=1\}$

$Q_{11}^2 = \{0 \leq I \leq 1\}$

$Q_{10}^2 = \{1 \leq I \leq 2\}$

3ème étape

$Q_9^3 = Q_4^3 = Q_9^2 \nabla \{1 \leq I \leq 2,\ J \leq I\}$

$= \{I \geq 1,\ J \leq I\}$

$Q_5^3 = \{I \geq 1,\ J \leq I-1\}$

$Q_3^3 = \{I \geq 1\}$

$Q_{11}^3 = \{I \geq 0\}$

$Q_{10}^3 = \{I \geq 1\}$

$env\ (Q_5^3 \wedge I \neq J,\ Q_{10}^3 \wedge I=J) = Q_9^3$

La séquence a convergé vers une solution exacte

$Q_2^3 = Q_{11}^3$

$Q_1^3 = \{N \geq 0\}$.

Le contexte Q_1 associé à l'arc d'entrée n'est pas égal à $\mathbb{R}^3$: on a détecté que si N < 0, le programme ne termine pas.
Nous donnons ci-dessous les résultats de l'analyse en avant et de l'analyse en arrière :

$P_1 = \mathbb{R}^3$
$P_2 = \{I=N\}$
$P_3 = \{I \leq N\}$
$P_4 = \{I \leq N,\ J=0\}$
$P_5 = \{I \leq N,\ J \geq 0\}$
$P_9 = \{I \leq N,\ J \geq 1\}$
$P_{10} = \{0 \leq I \leq N,\ J=I\}$
$P_{11} = \{-1 \leq I \leq N-1,\ J=I+1\}$
$P_{12} = \{I=0,\ I \leq N\}$

$Q_1 = \{N \geq 0\}$
$Q_2 = \{0 \leq I \leq 1\}$
$Q_3 = \{I \geq 1\}$
$Q_4 = \{I \geq 1,\ I \geq J\}$
$Q_5 = \{I \geq 1,\ I \geq J+1\}$
$Q_9 = \{I \geq 1,\ I \geq J\}$
$Q_{10} = \{I \geq 1\}$
$Q_{11} = \{I \geq 0\}$
$Q_{12} = \mathbb{R}^3$

6.5 - Combinaison des deux interprétations [Cousot 78b, 5.6.2.3. & 5.7.5.].

Nous nous proposons maintenant de caractériser l'ensemble des états de calcul qui sont simultanément descendants d'un état d'entrée satisfaisant une spécification d'entrée ϕ et ascendants d'un état de sortie satisfaisant une spécification de sortie ψ.

6.5.1. - Soient $\{P_k = F_k(P_1 \ldots P_\alpha) \mid k=1 \ldots \alpha\}$ et $\{Q_k = G_k(Q_1 \ldots Q_\alpha) \mid k=1 \ldots \alpha\}$ les systèmes d'équations respectivement en avant et en arrière associés au programme π, muni des spécifications d'entrées ϕ et de sortie ψ. Alors l'ensemble des états descendant de ϕ et ascendant de ψ est caractérisé par le système d'équations :

$$\{R_k = F_k(R_1 \ldots R_\alpha) \cap G_k(R_1 \ldots R_\alpha)\}$$

6.5.2. - Une première approximation des R_k consiste à associer à chaque arc a_k le contexte $P_k \cap Q_k$.

6.5.3. - Une meilleure solution consiste à construire une séquence analogue à celle de l'analyse en arrière, mais en posant cette fois : $R'_k = P_k \cap G_k(R'_1 \ldots R'_\alpha)$. Cette solution est malheureusement difficile à mettre en oeuvre, en raison du coût élevé du calcul de l'intersection des polyèdres. Notons cependant quelques simplifications possibles :

. Noeud de test : si a_i, a_V, a_F sont respectivement les arcs incident, de sortie "vrai" et "faux" d'un noeud de test portant sur la condition C, alors :

$$R'_i = P_i \cap env(R'_V \wedge C, R'_F \wedge \neg C) = env\ (R'_V, R'_F)$$

. Affectation linéaire inversible : si l'expression E(X) est linéaire inversible alors $\forall P,Q \in K(\mathbb{R}^n)$:

$$ar\text{-}affect\ (affect\ (P, X_i := E(X)) \cap Q, X_i := E(X)) \subseteq P.$$

6.5.4. - On peut encore combiner les deux interprétations, en supposant que le compilateur place en certains points du programme un test dynamique pour vérifier que les variables appartiennent au contexte en arrière. L'analyse en avant du programme ainsi modifié fournit alors les propriétés impliquées par les tests.

Exemple

Dans le programme du § 6.4., l'analyse en arrière a permis de découvrir une condition d'entrée (N≥0) nécessaire pour que le programme termine. Appelons P'_i, i = 1...12, les contextes résultats de l'analyse en avant à partir de cette spécification d'entrée. Nous donnons ci-dessous ces résultats, simultanément avec les résultats de l'analyse en avant sans spécification d'entrée :

$P_1 = \mathbb{R}^3$
$P_2 = \{I=N\}$
$P_3 = \{I\leq N\}$
$P_4 = \{I\leq N,\ J=0\}$
$P_5 = \{I\leq N,\ J\geq 0\}$
$P_9 = \{I\leq N,\ J\geq 1\}$
$P_{10} = \{0\leq I\leq N,\ I=J\}$
$P_{11} = \{-1\leq I\leq N-1,\ I=J-1\}$
$P_{12} = \{I=0,\ I\leq N\}$

$P'_1 = \{N\geq 0\}$
$P'_2 = \{I=N,\ N\geq 0\}$
$P'_3 = \{1\leq I\leq N\}$
$P'_4 = \{1\leq I\leq N,\ J=0\}$
$P'_5 = \{1\leq I\leq N,\ J\geq 0\}$
$P'_9 = \{1\leq I\leq N,\ J\geq 1\}$
$P'_{10} = \{1\leq I\leq N,\ I=J\}$
$P'_{11} = \{0\leq I\leq N-1,\ I=J-1\}$
$P'_{12} = \{N\geq 0,\ I=0\}$

7 - PRIMITIVES EVOLUEES

Nous allons montrer maintenant comment la méthode peut être appliquée à des programmes écrits en langage plus évolué. Nous étudierons successivement la structure de bloc, les procédures, éventuellement récursives, et nous montrerons enfin comment mettre en oeuvre l'analyse des propriétés des objets décrits par un mécanisme de types abstraits. Nous ne parlerons ici, que de l'analyse en avant. Les éléments présentés dans ce chapitre n'ont pas été implémentés.

7.1 - Structure de bloc

Pour prendre en compte la structure de bloc, nous devons associer à chaque point d'un programme son environnement, c'est-à-dire la liste ordonnée $X = \{X_1 \ldots X_n\}$ des identificateurs des variables numériques visibles en ce point. Si $Y = \{Y_1 \ldots Y_p\}$ est une liste d'identificateurs visibles au point a_i, nous noterons $\omega(Y,a_i)$ la permutation de Y conforme à l'ordre des identificateurs dans l'environnement de a_i. Pour simplifier, nous supposerons qu'un identificateur I déclaré dans un bloc B, ne peut être redéclaré dans un bloc intérieur à B.
Par abus de notations, puisque la dimension de l'espace de référence varie maintenant selon l'environnement, nous noterons :

$$proj(P,Y) = \{X \mid \exists\ Y \text{ tel que } (X,Y) \in P\}$$

Alors, les opérations d'entrée et de sortie de bloc sont interprétées comme suit, en admettant que les identificateurs $X_1 \ldots X_n$, $Y_1 \ldots Y_p$ sont tous syntaxiquement différents :

$\{(X_1 \ldots X_n) \in P\}$
début
  réel $Y_1 \ldots Y_p$;
  $\{(X_1 \ldots X_n, Y_1 \ldots Y_p) \in \mathbb{R}^{n+p} \mid (X_1 \ldots X_n) \in \hat{P}\}$

  . . .

  $\{(X_1 \ldots X_n, Y_1 \ldots Y_p) \in Q\}$
fin ;
$\{(X_1 \ldots X_n) \in proj(Q, Y_1 \ldots Y_p)\}$

Les branchements à une étiquette extérieure au bloc courant sont traités de la même manière [Cousot 78b, § 6.1], à condition que les étiquettes soient statiquement déterminées.

7.2 - Procédures Récursives

7.2.1. - Dans [Cousot 77b, Cousot 78b] l'analyse d'une procédure récursive associe à tout point a_i de la procédure un transformateur de prédicats Φ_i, tel que l'ensemble des valeurs possibles des variables X, environnement de la procédure, au point a_i soit caractérisé par le prédicat $\Phi_i(\phi)$, quelle que soit l'exécution de la procédure à partirde paramètres d'entrée vérifiant ϕ . L'application que nous traitons ici permet une formulation plus simple de l'analyse des procédures, parce que les propriétés analysées expriment des relations entre les variables.
En effet, soit $V = (v_1 \ldots v_p)$ et $R = (r_1 \ldots r_q)$ les paramètres d'une procédure f, passés respectivement par valeur et par résultat, et soit $X = (X_1 \ldots X_n)$ l'environnement du corps de f. Alors nous associerons à chaque point a_i de f un polyèdre $P_i \in \mathbb{R}^{n+p}$ de telle sorte que le transformateur de polyèdres Φ_i défini plus haut s'exprime de la manière suivante :

$$\forall P \in \mathbb{R}^{p+q},\ \Phi_i(P) = \{X \in \mathbb{R}^n \mid \exists\ (V_o, R_o) \in \mathbb{R}^{p+q} \wedge (V_o, R_o) \in P \wedge (X, V_o) \in P_i\}$$

En d'autres termes, en chaque point de la procédure nous chercherons les contraintes liant les valeurs courantes des variables et les valeurs initiales des paramètres par valeur. Ces contraintes seront établies indépendamment de tout contexte d'appel, mais en adjoignant à ce système les contraintes sur les valeurs initiales,

connues lors d'un appel spécifique, et en éliminant de ce nouveau système les variables représentant leurs valeurs initiales, on obtiendra les contraintes sur les valeurs courantes des variables.

7.2.2. - Les règles de déduction sur les instructions du corps de la procédure (affectation, test, jonction) sont donc inchangées. Il reste à définir les règles correspondant à l'entrée, à la sortie et à l'appel de procédure.
Tout d'abord, nous introduirons quelques hypothèses :

Restrictions Syntaxiques (prises pour simplifier la présentation) :

- les paramètres effectifs passés par valeur ou par valeur-résultat sont toujours des variables.
- tous les paramètres effectifs sont des variables syntaxiquement distinctes.
- nous considérons qu'aucune variable globale n'est visible à l'intérieur d'une procédure
- enfin nous omettons le cas des fonctions, qui ne sont qu'une facilité d'écriture.

Restrictions Sémantiques

- Nous ne traiterons pas les passages de paramètres par nom et par référence, qui posent de difficiles problèmes d'"alias".
- Enfin nous supposerons qu'on peut toujours associer statiquement un corps de procédure à un appel de procédure.

L'interprétation d'une procédure se définit comme suit (nous supposons que toutes les variables sont syntaxiquement distinctes) :

procédure f (*réel valeur* $v_1 \ldots v_p$; *réel résultat* $r_1 \ldots r_q$;
réel valeur résultat $u_1 \ldots u_s$)
entrée (V,R,U) = $\{(V,R,U,V_o,U_o) \in R^{2p+q+2s} \mid \forall i=1..p,\ V_i=V_{oi}$ et $\forall j=1..s,\ U_j=U_{oj}\}$
. . .

$\{(V,R,U,V_o,U_o) \in$ *sortie*$\}$
finproc ;

Le polyèdre *sortie* résulte de l'analyse du corps de f.
Soit (X,V,R,U) l'environnement d'un appel f(V,R,U) avec $X = (x_1 \ldots x_n)$, $V = (v_1 \ldots v_p)$, $R = (r_1 \ldots r_q)$, $U = (u_1 \ldots u_s)$ représentant des variables syntaxiquement distinctes, et soit $P \subseteq \mathbb{R}^{n+p+q+s}$ le polyèdre associé au point a précédant l'appel. Alors nous noterons *appel* (f,V,R,U,P) le polyèdre associé au point suivant l'appel, et défini comme suit :

$$\begin{aligned} \textit{appel}\ (f,V,R,U,P) = \{(X,V,R,U) \in \mathbb{R}^{n+p+q+s} \mid \exists\, V' \in \mathbb{R}^p,\ \exists\, R_o \in \mathbb{R}^q,\ \exists\, U_o \in \mathbb{R}^s \\ \text{tels que } \omega(X,V,R_o,U_o,a) \in P \text{ et } (V',R,U,V,U_o) \in \textit{sortie}\} \ ) \\ = \textit{proj}\ (\{(X,V,R,U,V',R_o,U_o) \mid \\ \omega(X,V,R_o,U_o,a) \in P \text{ et } (V',R,U,V,U_o) \in \textit{sortie}\},\ V',R_o,U_o) \end{aligned}$$

7.2.3. - Calcul de la séquence d'approximations

Dans le cas des procédures non récursives, la séquence est calculée comme précédemment (chapitre 5). Par contre la séquence des polyèdres correspondant à une procédure récursive peut être infinie. Il convient donc d'accélérer la convergence par élargissement, ce que nous conviendrons de faire au niveau du polyèdre de sortie de la procédure.

7.2.4. - Exemples

7.2.4.1. - Fonction 91 de Mac Carthy

```
procédure f (entier valeur x ; entier résultat y) = {P1}
   si x>100 alors y:=x-10                          {P2}
   sinon
        début entier z,t ;
           z:=x+11 ;                               {P3}
           f(z,t)                                  {P4}
           f(t,y) ;                                {P5}
        fin ;                                      {P6}
   finsi ;                                         {P7}
finproc ;
```

Système d'équations approchées associé à f :

$$P_1 = \{(x,y,x_0) \mid x=x_0\}$$

$$P_2 = \{(x,y,x_0) \in aff(P_1 \wedge x\geq 101), y := x-10)\}$$

$$P_3 = \{(x,y,x_0,z,t) \in aff(\{(x,y,x_0,z,t) \mid (x,y,x_0) \in (P_1 \wedge x\leq 100),\ z:=x+11\})$$

$$P_4 = \{(x,y,x_0,z,t) \in appel(f,z,t,P_3)\}$$

$$= \{(x,y,x_0,z,t) \in proj(\{(x,y,x_0,z,t,z',t_0) \mid (x,y,x_0,z,t_0) \in P_3 \text{ et } (z',t,z) \in P_7\},\ z',t_0)\}$$

$$P_5 = \{(x,y,x_0,z,t) \in appel(f,t,y,P_4)\}$$

$$= \{(x,y,x_0,z,t) \in proj(\{x,y,x_0,z,t,t',y_0) \mid (x,y_0,x_0,z,t) \in P_4 \text{ et } (t',y,t) \in P_7\},\ t',y_0)\}$$

$$P_6 = \{(x,y,x_0) \in proj\ (P_5,z,t)\}$$

$$P_7 = \{(x,y,x_0) \in env(P_2,P_6)\}$$

Résolution approchée du système

Initialisation : $P_i^0 = \emptyset$, $i = 1..7$

1ère étape :

$$P_1^1 = \{x=x_0\}$$
$$P_2^1 = \{x=x_0,\ x\geq 101,\ y=x-10\}$$
$$P_3^1 = \{x=x_0,\ x\leq 100,\ z=x+11\}$$
$$P_4^1 = P_5^1 = P_6^1 = \emptyset$$
$$P_7^1 = P_2^1$$

2ème étape :

$$P_4^2 = \{x=x_0,\ x\leq 100,\ z=x+11,\ z\geq 101,\ t=z-10\}$$
$$P_5^2 = \{x=x_0,\ x\leq 100,\ z=x+11,\ z\geq 101,\ t=z-10,\ t\geq 101,\ y=t-10\}$$
$$= \{x=x_0=100,\ z=111,\ t=101,\ y=91\}$$
$$P_6^2 = \{x=x_0=100,\ y=91\}$$
$$P_7^2 = P_7^1 \nabla env(P_2^1,P_6^2)$$
$$= P_7^1 \nabla \{x=x_0,\ y\geq 91,\ x-10\leq y\leq x-9\}$$
$$= \{x=x_0,\ x-10\leq y,\ y\geq 91\}$$

3ème étape

P_4^3 = {x=x_o, x≤100, z=x+11, z-10≤t, t≥91}
P_5^3 = {x=x_o, x≤100, z=x+11, z-10≤t, t≥91, t-10≤y, y≥91}
P_6^3 = {x=x_o, x≤100, x-9≤y, y≥91}
env(P_2^1,P_6^3) = P_7^2

La séquence a convergé.

La solution obtenue pour P_7 est égale à l'enveloppe convexe du domaine réel qui est

{x_o≤100, y=91} ∪ {x_o≥101, y=x-10}

7.2.4.2. - Tours de Hanoï

```
procédure Hanoï (entier valeur n ; entier valeur-résultat a,b,c ;
                 entier tableau [*] TA,TB,TC) =                {P1}
   si n=1 alors
      b:=b+1 ; TB[b]:=TA[a]; a:=a-1                            {P2}
   sinon
      début entier m ; m:=n-1 ;                                {P3}
           Hanoï (m,a,c,b,TA,TC,TB) ;                          {P4}
           b:=b+1 ; TB[b]:=TA[a]; a:=a-1 ;                     {P5}
           Hanoï(m,c,b,a,TC,TB,TA) ;                           {P6}
      fin ;                                                    {P7}
   finsi ;                                                     {P8}
finproc ;
```

Système d'équations approchées

$$P_1 = \{(n,a,b,c,n_0,a_0,b_0,c_0) \mid n=n_0,\ a=a_0,\ b=b_0,\ c=c_0\}$$

$$P_2 = \{(n,a,b,c,n_0,a_0,b_0,c_0) \in \mathit{aff}(\mathit{aff}((P_1 \wedge n=1),\ b:=b+1),\ a:=a-1)\}$$

$$P_3 = \{(n,a,b,c,n_0,a_0,b_0,c_0,m) \in \mathit{aff}(\{(n,a,b,c,n_0,a_0,b_0,c_0,m) \mid (n,a,b,c,n_0,a_0,b_0,c_0) \in P_1\},\ m:=n-1)\}$$

$$P_4 = \{(n,a,b,c,n_0,a_0,b_0,c_0,m) \in \mathit{proj}(\{(n,a,b,c,n_0,a_0,b_0,c_0,m,a_1,b_1,c_1,m') \mid (n,a_1,b_1,c_1,n_0,a_0,b_0,c_0,m) \in P_3 \wedge (m',a,c,b,m,a_1,c_1,b_1) \in P_8\},\ m',a_1,b_1,c_1)\}$$

$$P_5 = \{(n,a,b,c,n_0,a_0,b_0,c_0,m) \in \mathit{aff}(\mathit{aff}(P_4,\ b:=b+1),\ a:=a-1)\}$$

$$P_6 = \{(n,a,b,c,n_0,a_0,b_0,c_0,m) \in \mathit{proj}(\{(n,a,b,c,n_0,a_0,b_0,c_0,m,a_1,b_1,c_1,m') \mid (n,a_1,b_1,c_1,n_0,a_0,b_0,c_0,m) \in P_5 \wedge (m',c,b,a,m,c_1,b_1,a_1) \in P_8\},\ m',a_1,b_1,c_1)\}$$

$$P_7 = \{(n,a,b,c,n_0,a_0,b_0,c_0) \in \mathit{proj}(P_6,m)\}$$

$$P_8 = \{(n,a,b,c,n_0,a_0,b_0,c_0) \in \mathit{env}(P_2,P_7)\}$$

Résultats

```
procédure Hanoï (entier valeur n ; entier valeur-résultat a,b,c ;
                 entier tableau [*] TA,TB, TC) =
   {n=n_o, a=a_o, b=b_o, c=c_o}
   si n=1 alors
      b:=b+1 ; TB[b]:=TA[a]; a:=a-1
      {n=n_o=1, a=a_o-1, b=b_o+1, c=c_o}
   sinon
      début entier m ; m:=n-1 ;
          {n=n_o=m+1, a=a_o, b=b_o, c=c_o}
          Hanoï (m,a,c,b,TA,TC,TB) ;
          {n=n_o=m+1, m≥1, a=a_o-m, b=b_o, c=c_o+m}
          b:=b+1 ; TB[b]:=TA[a]; a:=a-1 ;
          {n=n_o=m+1, m≥1, a=a_o-m-1, b=b_o+1, c=c_o+m}
          Hanoï (m,c,b,a,TC,TB,TA) ;
          {n=n_o=m+1, m≥1, a=a_o-m-1, b=b_o+m+1, c=c_o}
      fin ;
      {n=n_o, n_o≥2, a=a_o-n_o, b=b_o+n_o, c=c_o}
   finsi ;
   {n=n_o, n_o≥1, a=a_o-n_o, b=b_o+n_o, c=c_o}
finproc ;
```

Remarques :

- Le prédicat de sortie contient une condition sur n_o ($n_o \geq 1$) qui est une condition nécessaire de terminaison de la procédure.
- Les résultats obtenus rendent compte de manière précise du comportement de la procédure : en effet, Hanoï étant appelée initialement avec les paramètres $a_o=n_o$, $b_o=c_o=0$, d'après le prédicat de sortie, le résultat de l'appel sera $a=0$, $b=n_o$, $c=0$, ce qui correspond bien aux spécifications du problème des Tours de Hanoï.

7.3 - Types Abstraits

Dans ce paragraphe, nous montrons comment notre méthode pourrait être étendue à l'analyse de programmes manipulant des objets décrits par une spécification abstraite. Quoique le principe de cette extension soit simple, les problèmes concrets posés par son application ne sont pas tous résolus.

7.3.1. - Généralités

L'idée est d'associer à un objet d'un type abstrait un ensemble de valeurs numériques telles que certaines conditions de validation des opérations sur les objets du type abstrait puissent s'exprimer sous forme de relations linéaires sur ces valeurs numériques.

Exemple

Si l'on associe à tout objet du type *pile* la hauteur h de la pile, la précondition de l'opération *dépiler* s'écrit $h \geq 1$. □

Autrement dit, à un type abstrait T, nous voulons associer deux nouvelles fonctions d'abstraction et de concrétisation α' et γ' (ou un ensemble de couples $(\alpha'_1, \gamma'_1) .. (\alpha'_p, \gamma'_p)$) telles que

$\alpha' : T \rightarrow \mathbb{E} \quad \gamma' : \mathbb{E} \rightarrow P(T)$

$\forall u \in T,\ u \in \gamma'(\alpha'(u))$

$\forall y \in P(\mathbb{E}),\ y = \alpha'(\gamma'(y))$

où $\mathbb{E}$ est un domaine numérique (réels, entiers, intervalle...).

Par extension, si $U \in P(T)$, on notera $\alpha'(u)$ l'ensemble $\{\alpha'(u) \mid u \in U\}$.
D'autre part, on aura toujours $\gamma'(x) = \{u \in T \mid \alpha'(u)=x\}$.
Alors $\gamma' \circ \gamma \circ \alpha \circ \alpha'$ est une fermeture sur $P(T)$, et d'après [Cousot 78b, ch.4] ceci est suffisant pour définir une interprétation approchée des objets de type T.

7.3.2. - Spécification de l'interprétation numérique d'un type abstrait

7.3.2.1. - Dans la mesure où la valeur numérique $\alpha'(t)$ que nous associons à un objet $t \in T$ sera souvent effectivement concrétisée dans l'implémentation de t (par exemple si une pile est implémentée sous forme de tableau, sa hauteur sera concrétisée par l'indice du sommet de la pile), il pourrait suffire de donner la correspondance entre $\alpha'(t)$ et sa concrétisation, et de laisser le système déduire les propriétés de $\alpha'(t)$ à partir de l'analyse de l'implémentation de T. Si l'on veut séparer nettement la spécification du type, de son implémentation, il faut que le programmeur spécifie lui-même le comportement de $\alpha'(t)$ quitte ensuite à vérifier la cohérence de cette spécification avec une implémentation.

7.3.2.2. - Tout objet de type T est obtenu à l'issue d'une suite d'opérations à partir de constantes prédéfinies du type. L'analyseur pourra considérer les variables $\alpha'(u)$, $u \in T$, comme des variables numériques ordinaires à condition de connaître $\alpha'(c)$ pour toute constante prédéfinie c de T, et de savoir interpréter les opérations sur les objets de type T.
Nous utilisons pour définir les types abstraits, un langage semblable à Alphard ([Wulf 76]), à cause de la claire séparation qui est faite dans ce langage entre la spécification et l'implémentation des types, et de la possibilité qu'il offre de préciser les pré et post-conditions des procédures sur les objets.
L'écriture de la partie *"spécifications"* d'une *"forme"* s'accompagne de la définition d'une ou plusieurs interprétations numériques $\alpha'_1 \ldots \alpha'_p$ comme suit :

- D'abord le nom du type est déclaré, ainsi que les paramètres d'instantiation. A ce niveau sont déclarées les interprétations numériques éventuelles. Si $(x_1 \ldots x_n)$ sont les paramètres numériques, le programmeur définira dans le reste de la spécification, le comportement de $\alpha(x_1 \ldots x_n, \alpha'_1(u) \ldots \alpha'_p(u))$ où u est l'objet spécifié.

- Les propriétés invariantes des objets du type peuvent induire des propriétés des valeurs des fonctions α'_i, propriétés qui seront utilisées par l'analyseur.

. Dans la partie d'initialisation, le programmeur donnera les valeurs des fonctions d'interprétation correspondant à un objet du type immédiatement après sa création.

. Pour chaque procédure $f : T^q \times R^r \to T^s \times R^t$, on donnera :

$$\tilde{pré}(f) = \alpha(\{(X,\alpha'_1(U)\ldots\alpha'_p(U)) \mid X \in R^r,\ U \in T^q,\ \text{et } (X,U) \in pré(f)\})$$

$$\tilde{post}\ (f) = \alpha(\{(X,Y,\alpha'_1(U)\ldots\alpha'_p(U),\alpha'_1(V)\ldots\alpha'_p(V) \mid X \in R^r,\ Y \in R^t,\ U \in T^q,\ V \in T^s \text{ et } (X,Y,U,V) \in post\ (f)\})$$

. Si la fonction f est à résultat booléen, le programmeur donnera :

$$\tilde{vrai}\ (f) = \alpha(\{X,\alpha'_1(U)\ldots\alpha'_p(U) \mid X \in R^r,\ U \in T^q \text{ et } f(X,U) = vrai\})$$

$$\tilde{faux}\ (f) = \alpha(\{X,\alpha'_1(U)\ldots\alpha'_p(U) \mid X \in R^r,\ U \in T^q \text{ et } f(X,U) = faux\})$$

7.3.2.3. exemple

forme pile (entier n ; type t) =	{h(pile) : entier}
spécifications	
requiert n>0 ;	{$\tilde{req}$ n≥1}
soit pile = <...x_i...>*où* x_i *est* t ;	
invariant 0 ≤ *longueur* (pile)≤n ;	{$\tilde{inv}$:0≤h(pile)≤n}
initialement pile = *sequence nulle*	{$\tilde{init}$: h(pile)=0}
procédures	
empiler (pile *valeur-résultat* p ; t *valeur* x);	
pré : 0≤*longueur*(p)<n	{$\tilde{pré}$ (empiler): 0≤h(p)<n}
post : p=p_o ∿ x	{$\tilde{post}$ (empiler): h(p)=h(p_o)+1}
dépiler (pile *valeur-résultat* p) ;	
pré : 0<*longueur* (p)≤n	{$\tilde{pré}$ (dépiler): 0<h(p)≤n}
post : p=*début* (p)	{$\tilde{post}$ (dépiler): h(p)=h(p_o)-1}
sommet (pile *valeur* p ; t *résultat* x);	
pré : 0<longueur (p)≤n	{$\tilde{pré}$ (sommet): 0≤h(p)≤n}
post : x=dernier (p)	{$\tilde{post}$ (sommet): h(p)=h(p_o)}
vide (pile valeur p) retourne b:booléen ;	
post : b=(p=*séquence nulle*)	{$\tilde{vrai}$ (vide): h(p)=h(p_o)=0}
	{$\tilde{faux}$ (vide): h(p)=h(p_o)≥1}

7.3.2.4. - *Remarques*

. Dans l'exemple ci-dessus on a pu décrire exactement l'influence des fonctions sur l'interprétation numérique : toutes les post-conditions sont des équations définissant h(p). Il n'en est pas toujours de même : ainsi si la cardinalité c est une interprétation du type "ensemble", alors la post-condition de la fonction qui inserre un élément dans un ensemble e est : $c(e') \leq c(e) \leq c(e'+1)$.

. D'autres exemples de spécifications seraient plus difficiles à traiter : ainsi si l'on associe au type "noeud d'un arbre" l'interprétation décrivant la hauteur d'un noeud, alors la fonction qui inserre une feuille dans l'arbre modifie les hauteurs d'un ensemble de noeuds qui ne lui sont pas donnés en paramètres, et peuvent même ne pas être visibles au point d'appel de la fonction. Ce type de problèmes reste à étudier.

7.3.3. - Analyse des Programmes comportant des Spécifications Abstraites

Une interprétation numérique α' du type T étant fournie, le système est alors capable d'associer à tout programme utilisant T un système d'équations approchées rendant compte du comportement des variables "virtuelles" $\alpha'(t)$, $t \in T$, au même titre que des variables de types numériques du programme.

7.3.3.1. - Déclaration

Toute déclaration d'une variable t de type T revient à la déclaration d'une variable $\alpha'(t)$ de type E. La création de la variable t n'étant pas une opération séparée, en Alphard, le contexte successeur de la déclaration contiendra les contraintes sur les paramètres du type (contraintes $\tilde{req}$), les contraintes sur $\alpha'(t)$ à l'initialisation (contraintes $\tilde{init}$), et les contraintes résultant des invariants du type (contraintes $\tilde{inv}$).

7.3.3.2. - Appel de Procédure

L'appel d'une fonction f à partir du contexte Q, sera interprété comme l'appel d'une procédure ordinaire (ch. 7.2.2.) à partir du contexte $Q \cap \tilde{pré}(f)$, le contexte de sortie de la procédure étant $\tilde{post}(f)$.

7.3.3.3. - Test

Un test portant sur des variables de type T ne peut s'écrire qu'au moyen d'une fonction sur T à résultat booléen. Alors, avec les notations de 4.2.3. on aura :

$vrai\ (f,Q) = Q \cap \tilde{vrai}(f)$

$faux\ (f,Q) = Q \cap \tilde{faux}(f)$

7.3.3.4. - Exemple

Pour illustrer la construction d'un système d'équations, nous donnons ci-dessous l'exemple d'un programme transférant les valeurs de la pile P_1 dans la pile P_2, en inversant leur ordre. Les résultats de l'analyse d'un programme plus important seront donnés au § 7.3.4.

```
        procédure Inverse (pile (n, entier)valeur P1 ; pile(n,entier )résultat P2) =
{1}         entier x ;
{2}         tantque ( ¬vide(P1)) faire
{3}             sommet(P1,X) ;
{4}             empiler(P2,X) ;
{5}             dépiler(P1)
{6}         fait
        finproc ;
```

Système d'équations

$Q_1 = \{n \geq 1,\ h(P_1)=h_0,\ 0 \leq h_0 \leq n,\ h(P_2)=0\}$

$Q_2 = env(Q_1,Q_5) \cap \{h(P_1) \geq 1\}$

$Q_3 = Q_2$

$Q_4 = \{\exists\ h_1 \mid (n,h_0,h(P_1),h_1) \in Q_3 \wedge h(P_2)=h_1+1\} =$

$Q_5 = \{\exists\ h_1 \mid (n,h_0,h_1,h(P_2)) \in Q_4 \wedge h(P_1)=h_1-1\}$

$Q_6 = env(Q_1,Q_5) \cap \{h(P_1)=0\}$

7.3.4. - Exemple de programme analysé

Le programme ci-dessous construit dans la file de sortie la forme postfixée de l'expression infixée fournie dans la file d'entrée, conformément à la priorité (≤) des opérateurs, en utilisant une pile intermédiaire. L'interprétation du type file se déduit directement de celle du type pile. On n'a pas fait figurer les tailles maxima des files et de la pile. Pour simplifier l'écriture des résultats on note E,S,I respectivement les longueurs des files "entrée" et "sortie" et la hauteur de la pile "inter".

```
procédure Posftix (file (symbole) valeur entrée ; file (symbole) résultat sortie) =
   pile (symbole) inter ; symbole c, c1 ; {E=E0≥0, S=I=0}
   tantque ( ¬vide (entrée)) faire {I≥0, S≥0, E≥1, E+I+S≤E0}
      premier (entrée, c) ;
      si identificateur (c) alors % c est un identificateur %
         enfiler (sortie, c) {I≥0, S≥1, E≥1, E+I+S≤E0+1}
      sinon si opérateur (c) alors   % c est un opérateur %
         tantque ( ¬vide (inter)) faire {I≥1, S≥0, E≥1, E+I+S≤E0}
            sommet (inter,c1) ;
            si ( ¬opérateur (c1)) alors exit finsi ;
            si (c>c1) alors exit finsi ;
            enfiler (sortie, c1) ; dépiler (inter) ; {I≥0, S≥1, E≥1, E+I+S≤E0}
         fait ;
         empiler (inter,c) {I≥1, S≥0, E≥1, E+I+S≤E0+1}
      sinon si parenthèse-gauche-(c) alors % c est une ( %
         empiler (inter, c) {I≥1, S≥0, E≥1, E+I+S≤E0+1}
      sinon   % c est une parenthèse droite %
         tantque ( ¬vide (inter)) faire {I≥1, S≥0, E≥1, E+I+S≤E0}
            sommet (inter, c1) ;
            si ( ¬opérateur (c1)) alors exit finsi ;
            enfiler (sortie, c1) ; dépiler (inter) ; {I≥0, S≥1, E≥1, E+I+S≤E0}
         fait ;
         si vide (inter) alors erreur % trop de parenthèses droites %
         sinon dépiler (inter) ; {I≥0, S≥0, E≥1, E+I+S≤E0-1}
         finsi ;
      finsi ;
```

```
      défiler (entrée) ; {I≥0, S≥0, E≥0, E+I+S≤E0}
   fait ; {I≥0, S≥0, E=0, I+S≤E0}
   tantque ( ¬vide (inter)) faire {I≥1, S≥0, E=0, I+S≤E0 }
      sommet (inter, c1) ;
      si ( ¬opérateur (c1)) alors erreur % trop de parenthèses gauches %
      sinon enfiler (sortie, c1) ; dépiler (inter) ; {I≥0, S≥0, E=0, I+S≤E0}
      finsi ;
   fait ; {I=E=0, 0≤S≤E0}
finproc ;
```

7.3.5. - Conclusions sur les types abstraits

L'exemple précédent illustre la richesse des résultats que l'on peut obtenir : non seulement toutes les préconditions des appels de procédures ont été prouvées, mais on a de plus découvert que la somme des longueurs des trois listes était au plus égale à la longueur initiale de la file d'entrée +1. Cette information pourrait permettre à un compilateur d'optimiser l'implémentation des listes : on peut par exemple implanter les deux files bout à bout dans une zone de E_0+1 octets (si un symbole tient sur un octet) comme l'illustre la figure 7.1, ou même représenter les deux files et la pile sur une même liste circulaire de $2E_0+2$ octets (compte tenu du chaînage de la liste), moyennant des décalages de la pile, à mesure que la file d'entrée se vide et que la file de sortie croît (figure 7.2).

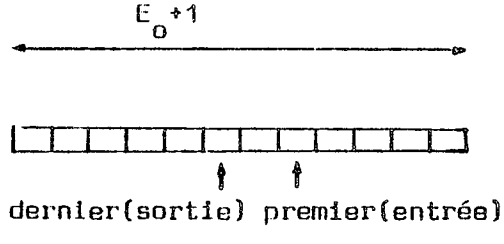


figure 7.1.

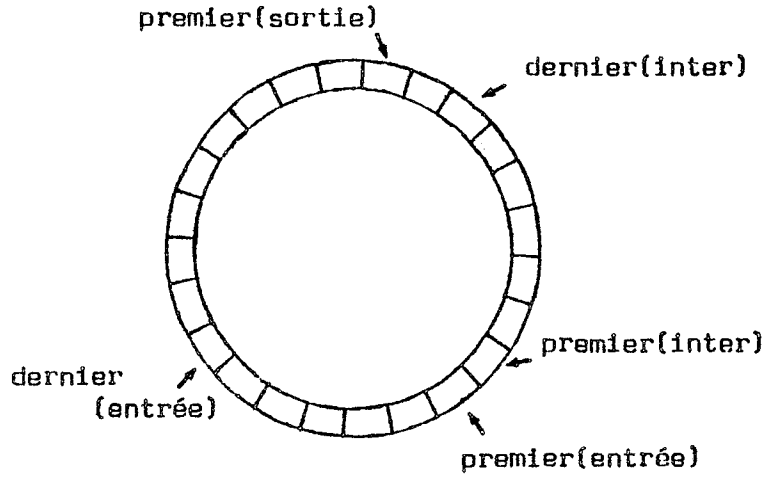


figure 7.2.

On peut ainsi envisager de laisser le compilateur choisir une représentation pour chaque objet d'un type abstrait, à partir des résultats d'une analyse sémantique. Il est bien évident cependant que de très gros problèmes restent à résoudre dans cette voie.

La difficulté essentielle soulevée par l'interprétation des types abstraits est que l'utilisateur définit lui-même les propriétés qu'il veut étudier, ainsi que la manière de les analyser. Citons une autre application possible de la méthode, qui ne présente pas ce problème, et qui est la détermination de la taille des objets dans un langage de très haut niveau comme SETL [Schwartz 75].

8 - APPLICATIONS DE LA METHODE

8.1 - Preuves de Propriétés

Soit (S,R,D) le système générateur du contexte associé à l'arc a_k à l'issue de l'analyse en avant. Pour prouver que, quelle que soit l'exécution du programme et quel que soit le moment où le contrôle parcourt a_k, les variables satisfont la contrainte $cx \leq b$, il suffit de vérifier que :

$\forall s \in S,\ cs \leq b,\ \forall r \in R,\ cr \leq 0$ et $\forall d \in D,\ cd = 0$

Cette condition n'est pas nécessaire, mais en progageant la contrainte au cours de l'analyse en arrière, on peut chercher des contre exemples, c'est-à-dire des valeurs d'entrée à partir desquelles l'exécution du programme conduit à un état violant la contrainte.

8.2 - Détermination de branches mortes, preuves de non terminaison

Si le contexte en evant associé à un arc reste vide après la convergence de la séquence d'approximations, alors quelles que soient les données satisfaisant la spécification d'entrée, le contrôle ne parcourra jamais cet arc. On détecte ainsi des chemins du graphe qui ne sont jamais exécutés, ce qui permet de supprimer les tests, origines de ces chemins. C'est ainsi que si, dans les programmes de tris analysés au § 5.6.4., on avait fait figurer les tests de cohérence des accès aux éléments de tableaux, ainsi qu'un noeud de sortie spécial correspondant aux erreurs d'indexation, tous les arcs menant à ce noeud d'erreur auraient été porteurs du contexte vide, ce qui permettrait de supprimer tous les tests correspondants.
D'autre part, si tous les contextes de sortie sont vides, le programme ne termine pas, quelles que soient les données satisfaisant à la spécification d'entrée.

8.3 - Preuves de Terminaison

On a vu comment l'analyse en arrière, et l'analyse en avant des procédures récursives, pouvaient fournir des conditions nécessaires de terminaison du programme.
Inversement, si à tout noeud d'élargissement on associe une nouvelle variable fictive, dont le rôle est de compter le nombre de passages du contrôle par le noeud d'élargissement, et si tous ces compteurs sont traités comme des variables ordinaires, il se peut que, dans les résultats de l'analyse en avant, certains de ces compteurs soient bornés supérieurement. Dans ce cas, on a prouvé que les boucles correspondantes se terminaient.

Exemple
Si l'on modifie localement le programme du § 5.6.2., en introduisant le compteur de boucle K, l'analyse en avant donne les résultats suivants :

```
I:=0 ; J:=0 ; K:=0 ; {I=0, J=0, K=0}
L: K:=K+1 ; {J≥0, I+2J+4=4K, I≤14, K≤7, 2K≤I+2}
si I≤10 alors  {J≥0, I+2J+4=4K, I≤10, 2K≤I+2}
   si...alors
      I:=I+2 ; J:=J+1 {J≥1, I+2J=4K, I≤12, 2K≤I}
   sinon I:=I+4  {J≥0, I+2J=4K, I≤14, 2K+2≤I}
   finsi ;  {J≥0, I+2J=4K, I≤14, 1≤K≤6, 2K≤I}
finsi ; {J≥0, I+2J+4=4K, 11≤I≤14, 2K≤I+2}
```

La séquence croissante converge en deux étapes.
La séquence décroissante atteint une solution exacte en une étape
Temps d'exécution : 0,49 secondes.

Par élimination de I et J dans le dernier contexte, on voit qu'à la sortie du programme, on a 4≤K≤7. On a donc montré que le test d'entrée dans la boucle est évalué au moins 4 fois et au plus 7 fois, c'est-à-dire que le corps de la boucle est exécuté de 3 à 6 fois.
Notons que cette technique peut servir à évaluer le temps d'exécution du programme.

8.4 - Optimisations du Code

La connaissance des contextes abstraits en avant permet à un compilateur d'optimiser le code généré du programme :

- On a déjà remarqué qu'un grand nombre de tests dynamiques pouvaient être supprimés (bornes de tableaux, formats de sortie, instruction *cas*, types intervalles, débordements arithmétiques, division par zéro,...).

- La détection des relations linéaires d'égalité invariantes en un point de programme permet dans certains cas un calcul à la compilation (lorsqu'une expression a été reconnue comme constante), et dans d'autre cas des simplifications dans les calculs, ainsi qu'une optimisation de l'allocation des registres.

- Lorsque les contraintes sur la mémoire sont fortes (micro-ordinateur) si l'on connait un domaine de valeurs restreint d'une variable, on peut ne lui allouer que la place strictement nécessaire à la représentation des valeurs de ce domaine. On a cité également au chapitre 7, comment l'allocation mémoire pouvait être optimisée de manière importante, dans les langages de très haut niveau comme SETL, lorsqu'on connait la taille des objets, même en fonction d'une constante symbolique dont la valeur n'est connue qu'à l'exécution.

9 - NOTES SUR L'IMPLEMENTATION ET LES PERFORMANCES

9.1 - Introduction

Le programme réalisant l'analyse en avant ou en arrière comprend environ 2500 lignes de langage PASCAL. Les temps d'exécution que nous avons cités sont ceux de l'ordinateur IRIS 80 du Centre Interuniversitaire de Calcul de Grenoble.

Cette implémentation a été précieuse dans la mesure où la quasi-totalité des programmes ne peuvent être analysés à la main. Nous avons pu, à partir de l'expérimentation, juger de la richesse des résultats, et ajuster les mécanismes d'approximation en fonction d'une étude des performances de l'analyse.

L'analyse combinée n'a pas été implémentée.

9.2 - Calculs en Nombres Réels

Des problèmes sont apparus, relatifs à la perte de précision dans les calculs en nombre réels. Théoriquement, ces problèmes peuvent être résolus au prix d'une augmentation du temps d'exécution, en considérant que les nombres manipulés sont des rationnels (ce sont mêmes des décimaux), et peuvent donc être traités, sans perte de précision, comme quotients d'entiers. En fait seuls les sommets des polyèdres ne sont pas des informations "à un facteur près". En effet, on peut multiplier les deux membres d'une contrainte, ou toutes les coordonnées d'un rayon ou d'une droite, par une même constante, sans en changer la valeur... On peut adopter la même représentation pour les systèmes générateurs que pour les systèmes de contraintes, c'est-à-dire associer à chaque sommet s un coefficient b tel que bs soit entier (ce coefficient correspond au terme constant d'une contrainte), et mémoriser bs et b. On peut alors faire en sorte que tous les coefficients, dans les systèmes ainsi représentés, restent entiers. Ce mode de calcul n'a pas encore été généralisé dans le système implémenté.

9.3 - Performances du Système

9.3.1. - Le tableau 9.1 donne, pour chacun des exemples d'analyse en avant du § 5.6., et pour chaque type de noeuds, le temps total T de traitement des noeuds de ce type, le nombre N de noeuds traités, et le temps moyen t = T/N de traitement d'un noeud. Il apparait clairement que le calcul de l'enveloppe convexe aux noeuds de jonction est de loin l'opération la plus coûteuse.

9.3.2. - La méthode est difficile à évaluer, pour plusieurs raisons :

. La complexité des algorithmes de manipulation des systèmes d'inéquations linéaires est mal connue, même en ce qui concerne les méthodes classiques de programmation linéaire. Le dénombrement des éléments extrémaux d'un polyèdre en fonction du nombre de ses contraintes à fait l'objet de nombreux travaux (par exemple [Saaty 55], [Klee 64], [Mac Mullen 70]), montrant une croissance potentiellement exponentielle. Notons que, d'après les résultats sur les polyèdres polaires (cf § 1.3.4.2.), il y a exactement autant de polyèdres dont le nombre d'éléments extrémaux est exponentiel, que des polyèdres dont le nombre d'éléments extrémaux est logarithmique, par rapport au nombre de leurs contraintes.
D'autre part, le nombre d'éléments extrémaux d'un polyèdre associé à un arc d'un graphe de programme est lié au nombre de chemins élémentaires menant à cet arc, à partir du noeud d'entrée dans l'analyse en avant, ou d'un noeud de sortie dans l'analyse en arrière. Ceci apparait clairement si l'on considère un contexte abstrait comme l'enveloppe convexe des résultats obtenus sur un arc par une exécution symbolique ([King 76]), simplifiée par élargissement au cours du parcours des boucles.

. Les critères d'évaluation sur les programmes analysés semblent difficiles à caractériser. En effet, le coût de l'analyse dépend plus de la complexité du graphe de programme (nombre de chemins, de boucles, degré d'imbrication ...) que de la longueur du programme. Rappelons que chaque composante fortement connexe du graphe peut être analysée séparément, et que donc le coût dépend linéairement du nombre de ces composantes.

Programme	Temps d'exécution	Nombre de variables	Affectation	Test	Jonction	Elargissement
§ 5.6.1.	0,22	2	N=8 T=0,033 t=0,004	N=0 T=0 t=-	N=5 T=0,121 t=0,024	N=1 T=0,027 t=0,027
§ 5.6.2.	0,37	2	N=8 T=0,029 t=0,003	N=3 T=0,039 t=0,013	N=5 T=0,216 t=0,043	N=1 T=0,031 t=0,031
Dichotomie 1ère version	1,69	4	N=6 T=0,218 t=0,036	N=9 T=0,252 t=0,028	N=5 T=0,978 t=0,195	N=1 T=0,113 t=0,113
Dichotomie 2ème version	2,8	4	N=6 T=0,23 t=0,038	N=8 T=0,473 t=0,059	N=7 T=1,911 t=0,273	N=1 T=0,115 t=0,115
Tri par insertion	0,75	4	N=9 T=0,087 t=0,009	N=6 T=0,115 t=0,019	N=10 T=0,343 t=0,034	N=1 T=0,068 t=0,068
Tri par sélection	1,86	4	N=16 T=0,316 t=0,019	N=11 T=0,373 t=0,033	N=11 T=0,742 t=0,067	N=2 T=0,201 t=0,1
Remontée des Bulles	0,89	4	N=13 T=0,257 t=0,019	N=5 T=0,101 t=0,02	N=8 T=0,334 t=0,041	N=1 T=0,097 t=0,097
"Heapsort"	5,8	5	N=19 T=0,685 t=0,036	N=22 T=0,953 t=0,043	N=18 T=3,15 t=0,175	N=1 T=0,284 t=0,284

Tableau 9.1

- Le temps d'exécution croît très vite avec le nombre des variables du programme. De ce point de vue apparaît clairement l'avantage des langages permettant une déclaration très locale des variables. On peut aussi déterminer, avant l'analyse proprement dite, la durée de vie de chaque variable. C'est une analyse statique sur un treillis fini de propriétés, et des algorithmes performants existent pour cela ([Allen 76]).
  Il faut remarquer qu'on ne peut pas prévoir quelles variables vont être liées par des relations. L'exemple 5.6.1. est significatif à ce propos : les variables I et J n'apparaissent jamais ensemble dans une même expression, et cependant on découvre des relations qui les lient. Si, dans un but d'efficacité, on effectue plusieurs analyses, chacune portant sur un sous-ensemble des variables du programme, on s'expose à une perte d'information.

- Enfin, le coût dépend de la richesse des résultats obtenus : plus on obtient d'information, plus longue est l'analyse. Il faut donc trouver un compromis entre la richesse des résultats et les performances du système, qui peuvent être améliorées au moyen d'approximations : si l'on supprime des contraintes dans un contexte, le résultat, quoique moins précis, reste correct, et le programme manipule des systèmes moins grands. Rien n'empêche donc de définir une taille maximum arbitraire pour les systèmes de contraintes et les systèmes générateurs, taille au delà de laquelle les contextes sont simplifiés. Le choix d'une stratégie de simplification reste à étudier selon les buts poursuivis : à titre d'exemple, on peut extraire par projections d'un système de contraintes quelconques, un système d'un type particulier (par exemple, système d'intervalles constants$\{ a_{i_k} \leq X_{i_k} \leq b_{i_k}$, $k=1...p\}$ ou symboliques $\{X_{i_k} \leq a_k X_{j_k} + b_k$, $k=1...p\}$). Un autre moyen d'améliorer les performances de l'analyse consiste à se limiter à certains types de contraintes. D'une part le nombre de contraintes est réduit, et d'autre part des algorithmes spécifiques plus efficaces peuvent être applqiués selon le type des contraintes étudiées.

9.3.3. - Nous avons délibérément choisi de ne pas tenir compte des assertions que l'on cherche à prouver, ceci pour garder à la méthode son caractère général de méthode de *découverte* statique. Cette approche peut cependant être combinée avec un point de vue de *vérification.* : la propagation en arrière des assertions que l'on veut vérifier fournit des indications pour l'élargissement et permet d'arrêter la séquence décroissante dès que les résultats suffisent à la preuve cherchée.

9.3.4. - Le volume de mémoire nécessaire au système peut être réduit en ne mémorisant que les contextes nécessaires, qui sont d'une part les contextes incidents aux noeuds en attente de traitement, et d'autre part les contextes sur lesquels est effectué l'élargissement. Les résultats sont alors récupérés par une étape supplémentaire de l'analyse.

10 - COMPARAISON AVEC DES TRAVAUX VOISINS

10.1. - Introduction

Depuis une dizaine d'années, les problèmes relatifs à l'analyse automatique ou semi-automatique des propriétés des programmes ont été un thème de recherches central dans plusieurs domaines : d'abord motivées par des systèmes d'aide à la preuve de correction totale des programmes, (par exemple [King 69]), les méthodes d'analyse ont été simultanément développées en optimisation et en vérification de propriétés. Elles sont actuellement appliquées surtout pour la production de code optimisé par un compilateur, par suppression de tests à l'exécution.

Nous rappellerons d'abord quelles sont les principales méthodes d'analyse des programmes, puis nous étudierons plus en détail les travaux les plus proches des notres.

10.2 - Méthodes d'Analyse des propriétés des programmes

10.2.1. - Approche Syntaxique

Cette méthode, que nous rappelons pour mémoire, est utilisée par [Welsh 77], pour vérifier la validité des opérations sur les variables de type intervalle, en Pascal. Cette méthode consiste à calculer le domaine d'une expression en fonction des domaines déclarés de ses termes. Cette vérification est évidemment très peu coûteuse, mais aussi très partielle puisque les résultats dépendent essentiellement de la précision avec laquelle le programmeur déclare ses variables. D'autre part, cette vérification ne s'applique qu'à des propriétés globales, et les déclarations de type classiques ne permettent pas d'exprimer des relations entre variables.

Les autres approches utilisent la propagation des assertions le long des chemins d'exécution du programme. Elles diffèrent essentiellement par la manière dont sont traitées les boucles.

10.2.2. - Approche Heuristique

Cette méthode consiste à tenter de construire un invariant de boucle, en essayant successivement des candidats de plus en plus forts. Un premier candidat est fourni par la propagation en arrière de la spécification de sortie, ou des propriétés que l'on cherche à prouver. Ce candidat peut alors être renforcé, au besoin, par diverses techniques spécifiques. La propagation des prédicats, selon la méthode de [Floyd 67] ou [Hoare 69], nécessite, si aucune approximation n'est effectuée, l'emploi d'importants démonstrateurs de théorèmes (par exemple [King 72], [Cooper 72]) pour simplifier les assertions. Cette approche a été utilisée, entre autres, par [Wegbreit 74], [German 75], [Katz 76], [Harrisson 77]; [Suzuki 77],[German 78].

10.2.3. - Approche Algorithmique

[Cheatam 76], [Harrisson 77], [Katz 76] ont combiné avec l'approche heuristique une approche algorithmique de découverte d'invariants, par la résolution de systèmes d'équations aux différences finies : sur chaque boucle, l'état des variables au k-ième tour de boucle est exprimé en fonction de l'état au (k-1)-ième tour. La résolution du système d'équations obtenu consiste à en déduire une relation entre les valeurs au k-ième tour et les valeurs d'entrée, puis à éliminer k. Il existe des algorithmes spécifiques pour résoudre certains types particuliers d'équations aux différences finies [Cohen 77], et cette méthode permet alors même la découverte de relations non linéaires. Cependant, il n'existe pas de méthode générale de résolution automatique de tels systèmes. [Berman 78] utilise la même approche, mais applique les méthodes de résolution des équations différentielles. Cette méthode part d'hypothèses assez restrictives, et ses auteurs ne savent pas si elle est complètement automatisable.

10.2.4. - Approche Itérative

Cette approche consiste à propager les spécifications d'entrée ou de sortie, et les propriétés à prouver le long des chemins d'exécution du programme. Comme ce processus n'est pas forcément fini, cette méthode n'a été appliquée qu'à l'étude de propriétés appartenant à des ensembles assurant la convergence de l'itération : Treillis finis comme dans le cas des méthodes booléennes

([Allen 70], [Kildall 73], [Allen 76]), ou treillis vérifiant la condition de chaîne ([Wegbreit 75], [Karr 75]). L'originalité de la méthode développée par P. & R. Cousot est d'avoir adopté une approche itérative, en accélérant la convergence par des mécanismes d'approximations. Deux applications ont été approfondies qui utilisent cette technique d'extrapolation :la méthode d'analyse des intervalles constants décrite dans [Cousot 76], et la nôtre.

10.3 - Travaux ayant des objectifs voisins des notres

10.3.1. - [Wegbreit 74] et [German 75] décrivent l'analyseur "Vista". Ce système combine une approche heuristique et une approche itérative : des candidats invariants sont construits par propagation en arrière des spécifications de sortie, et peuvent être renforcés selon diverses techniques. D'autre part des propriétés appartenant à des ensembles vérifiant la condition de chaîne, sont synthétisées par itération en avant. Notre opinion est que les systèmes basés sur un démonstrateur de théorèmes et des procédés heuristiques mettant en oeuvre des techniques d'exploration et de retour arrière risquent d'être bien plus coûteux que les méthodes de synthèse directe que nous utilisons. Par contre, nos résultats pourraient être avantageusement utilisés par un système comme "Vista".

10.3.2. - Découverte de relations linéaires d'égalité entre les variables

Le problème de la propagation des constantes, c'est-à-dire de la détection des expressions qui peuvent être évaluées à la compilation, a été traité par [Kildall 73]. Le treillis des propriétés étant de profondeur 3 (une expression est soit non initialisée, soit égale à une constante k, soit de valeur variable) la convergence de l'itération est assurée.

Un système plus puissant a été développé par [Karr 76] pour découvrir les équations linéaires vérifiées par les variables. Le treillis des propriétés étudiées étant l'ensemble des sous-espaces vectoriels de R^n, il est encore de profondeur finie et la méthode itérative converge en n étapes au plus. Remarquons que si une équation résulte d'une conjonction d'inéquations, elle échappera à cette analyse, alors que notre méthode pourra la trouver :
exemple : {I≤J} *si* I≥J *alors* {I=J}... *finsi* ; □

10.3.3. - Appartenance d'une variable à un intervalle

Plusieurs méthodes ont été proposées pour prouver qu'un indice de tableau restait entre les bornes déclarées. On distingue les méthodes de *vérification* et les méthodes de *découverte* des intervalles de variation des variables.

10.3.3.1. - [Suzuki 77] présente une méthode de vérification des intervalles par propagation en arrière des assertions à prouver. Devant chaque accès à un élément de tableau, la condition d'accès correct est créée, puis propagée en arrière. Si au cours de cette propagation on obtient une tautologie, la condition est démontrée. Lorsque une boucle est rencontrée, une suite d'invariants sont essayés, correspondant chacun à un tour supplémentaire dans la boucle. La longueur de cette suite est limitée arbitrairement pour éviter les dérivations infinies. Pour prouver un invariant, une condition suffisante est produite à l'aide du démonstrateur de [King 69]. Cette condition est propagée, d'une part en amont de la boucle, et d'autre part dans le corps de la boucle. Si cette condition ne peut être prouvée, elle est utilisée pour construire un candidat invariant plus faible. La complexité des assertions reste raisonnable grâce aux simplifications. Il faut noter cependant que ces simplifications se font de manière arbitraire, par élimination de variables, ce qui réduit la puissance de la méthode. D'autre part, l'approximation faite au niveau des boucles en limitant simplement le nombre d'étapes effectuées au cours de l'analyse, est assurément moins riche que les techniques visant à prévoir le comportement d'une boucle au moyen d'équations aux différences, ou par élargissement.

10.3.3.2. - Un autre système de vérification des intervalles d'entiers est décrit dans [German 78], utilisant d'une part des techniques de synthèse d'invariants (compteurs de boucles, équations aux différences - inutilisables en présence de *goto*), et d'autre part un ensemble varié de procédés heuristiques pour générer des candidats invariants qui sont ensuite prouvés ou infirmés à l'aide d'un démonstrateur de théorèmes. Par exemple on peut essayer de démontrer que les variables modifiées dans une boucle restent inférieures ou supérieures à leurs valeurs initiales.

D'une part, on ne sait pas combien il faut de telles heuristiques pour que le système soit complet, d'autre part l'analyse des relations linéaires entre variables tient implicitement compte des heuristiques proposées par [German 78], à un coût moindre puisqu'il n'y a pas d'essais infructueux.

10.3.3.3. - La méthode de découverte des intervalles de variation des variables de [Harrisson 77] consiste en une étape d'une séquence descendante à partir de $\{X_i \in [\infty, +\infty] \mid i = 1...n\}$, (donc, sans itérer), complétée par une résolution des équations aux différences dans des cas simples. Cette méthode est moins puissante que celle de [Cousot 76].

10.3.3.4. - [Cousot 76] décrit, à titre d'exemple de la théorie générale sur laquelle est fondée le présent travail, une méthode de découverte des intervalles. Notons que le modèle des intervalles permet de rendre compte des affectations et des tests non linéaires, et peut être utilisé pour préciser l'interprétation approchée que nous en donnons. L'analyse des intervalles est un exemple de restriction de la méthode à une classe de contraintes permettant une analyse moins coûteuse moyennant une perte de puissance.

En conclusion, disons que les méthodes d'analyse des intervalles du type [Suzuki 77] ou [Cousot 76] suffisent en général pour supprimer les tests de bornes, lorsque le langage impose que les valeurs numériques des bornes des tableaux soient connues à la compilation (type Pascal). Il n'en est pas de même pour des langages de type Algol, pour lesquels l'usage de notre méthode s'avère nécessaire.

11 - CONCLUSIONS

Un de nos objectifs, en commençant ce travail, était de montrer qu'il est possible de découvrir des propriétés non triviales en appliquant des méthodes approchées. Quoique les mécanismes d'approximation que nous avons choisis aient pu paraître très grossiers, le système obtenu est cependant l'un des plus puissants outils d'analyse automatique de propriétés sémantiques des programmes existant actuellement.

Un développement logique de ce travail serait l'intégration dans un compilateur, d'un préprocesseur susceptible de fournir les informations nécessaires à l'optimisation du code généré, et les messages d'avertissement indiquant à l'utilisateur les erreurs qui ont pu être détectées statiquement.

Les performances de l'analyse doivent être améliorées. On peut définir pour cela une hiérarchie d'interprétations correspondant d'une part à des classes de contraintes de plus en plus générales (signe des variables, équations, intervalles, intervalles symboliques ...) et d'autre part à des opérateurs d'approximation de plus en plus précis.

Les techniques d'optimisation tirant profit des informations collectées par l'analyseur, restent, pour la plupart, à explorer : si la suppression des tests dynamiques dont l'inutilité a été prouvée, ne semble pas poser de problèmes, il n'en est pas de même d'optimisations de plus haut niveau, comme la minimisation de l'espace mémoire, dont des exemples ont été donnés au § 7.3.5.

Un autre développement sera l'étude des constructions linguistiques les plus adéquates pour favoriser l'analyse statique. C'est ainsi, par exemple, que l'absence de branchements inconditionnels, ou d'une primitive *exit* permettant d'abandonner l'exécution d'une boucle en n'importe quel point du corps de la boucle, conduit le programmeur à nier artificiellement la condition d'itération de la boucle pour obtenir la sortie désirée :

Exemple : Dichotomie

programme 1

```
1.  L:=1 ; U:=N ;
2.  tantque L≤U faire
3.     I:=(L+U) div 2 ;
4.     si R=T[I] alors L:=U+1
5.     sinon si R>T[I] alors L:=I+1
6.     sinon U:=I-1 finsi ;
7.  refaire ;
```

programme 2

```
L:=1 ; U:=N ;
tantque L≤U faire
   I:=(L+U) div 2 ;
   si R=T[I] alors exit
   sinon si R>T[I] alors L:=I+1
   sinon U:=I-1 finsi ;
refaire ;
```

L'affectation "L:=U+1" à la ligne 4 du programme 1 ne sert qu'à provoquer l'arrêt lorsqu'on a trouvé l'élément cherché R. Cette affectation perturbe les variations de L, et par suite, l'analyse du programme 1 fournit des résultats moins précis que celle du programme 2.

Inversement, l'influence des techniques d'analyse statique des programmes devrait s'exercer plus fortement sur la conception des langages de programmation. En effet, si l'on veut pouvoir vérifier statiquement les propriétés spécifiées dans le langage -ce qui est raisonnable, eu égard à l'efficacité du code produit par le compilateur- la puissance d'expression du langage est limitée par le niveau des vérifications possibles. Le développement des méthodes de vérification statique est donc un préliminaire indispensable à un nouvel essor dans le domaine des langages.

BIBLIOGRAPHIE

[Allen 70]
Allen F.E., *Control flow analysis*
Sigplan Notices, vol.5, n°7, juillet 70.

[Allen 76]
Allen F.E., Cocke J., *A program data flow analysis procedure*
CACM, vol.19, n°3, mars 76.

[Balinski 61]
Balinski M.L., *An algorithm for finding all vertices of convex polyhedral sets*
Journal of the Society for Industrial and Applied Mathematics, vol.9, n°1, 1961.

[Berman 78]
Berman L., Markowsky G., *Linear and non linear approximate invariants*
RC 7241 (#31188), IBM T.J. Watson Res.Cent. 1978

[Chand 70]
Chand D.R., Kapur S.S., *An algorithm for convex polytopes*
J.A.C.M., vol.17, n°1, janvier 1970.

[Charnes 53]
Charnes A., Cooper W., Henderson A., An introduction to linear programming
J.Wiley, New-York, 1953.

[Cheatam 76]
Cheatam T.E., Townley J.A., *Symbolic evaluation of programs: A look at loop analysis*
TR 11-76, Center for Res. in Comp. Tech., Harward University, 1976.

[Cohen 77]
Cohen J., Katcoff J., *Symbolic solution of limite-difference equations*
ACM Trans. on Mathematical Software, vol.3, n°3, septembre 1977.

[Cooper 72]
Cooper D.C., *Programs for mechanical program verification*
Machine Intelligence, n°6, American Elsevier, New-York, 1972.

[Cousot 76]
Cousot P., Cousot R., *Static determination of dynamic properties of programs*
Proc. 2nd. Int. Symp. on Programming, Dunod, Paris, avril 1976.

[Cousot 77a]
Cousot P., Cousot R., *Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints*
4th. ACM Symp. on Principles of Programming Languages, Los Angeles, Janvier 1977.

[Cousot 77b]
Cousot P., Cousot R., *Static determination of dynamic properties of recursive procedure*
IFIP W.G.2.2. Working Conf. on Formal Description of Programming concepts, St-Andrews (Canada), août 1977.

[Cousot 77c]
Cousot P., Cousot R., *Automatic synthesis of optimal invariant assertions: mathematical foundations.*
ACM Symp. on Artificial Intelligence and Programming Languages, Rochester (N.Y.) Sigplan Notices, vol.12, n°8, août 1977.

[Cousot 78a]
Cousot P., Halbwachs N., *Automatic discovery of linear restraints among variables of a program*
5th ACM Symp. on Principles of Programming Languages, Tucson(Az.), janvier 1978.

[Cousot 78b]
Cousot P., Méthodes itératives de construction et d'approximation de points fix d'opérateurs monotones sur un treillis, Analyse sémantique des programmes
Thèse d'Etat, Université Scientifique et Médicale de Grenoble, mars 1978.

[Cousot 79]
Cousot P., Cousot R., *Systematic design of program analysis frameworks*
6th. ACM Symp. on Principles of Programming Languages. San-Antonio, janvier 19

[Dyer 77]
Dyer M.E., Proll L.G., *An algorithm for determining all extreme points of a convex polytope*
Mahtmetical Programming, n°12, 1977.

[Floyd 67]
Floyd R.W., *Assigning meaning to programs*
Symp. on Applied Mathematics, Amer. Math. Soc., n°19, 1967.

[German 75]
German S.M., Wegbreit B., *A synthetizer of inductive assertions*
IEEE Transactions on Software Engineering, vol. SE-1, nr1, mars 1975.

[German 78]
German S.M., *Automating proofs of the absence of common runtime errors*
5th Symp. on Principles of Programming Languages, Tucson(Az.), janvier 1978.

[Harrisson 77]
Harrisson W.H., *Compiler analysis of the value ranges for variables*
IEEE Transactions on Software Engineering, vol.SE-3, n°3, mai 1977.

[Hoare 69]
Hoare C.A.R., *An axiomatic approach to computer programming*
CACM, vol.12, n°10, octobre 1969.

[Hoare 71]
Hoare C.A.R., *Proof of a program: FIND*
CACM, vol.14, n°1, janvier 1971.

[Ibramsha 78]
Ibramsha M., Rajaraman V., *Detection of logical errors in decision table programs*
CACM, vol.21, n°12, décembre 1978.

[Karr 75]
Karr M., Gathering information about programs
Mass. Comp. Associates Inc., CA 7507 - 1411, juillet 1975.

[Karr 76]
Karr M., *Affine relationships among variables of a program*
Acta Informatica, n°6, 1976.

[Katz 76]
Katz S., Manna Z., *Logical analysis of programs*
CACM, vol.19, n°4, avril 1976.

[Kildall 73]
Kildall G.A., *A unified approach to global program optimization*
1st Symp. on Principles of Programming Languages, Boston, octobre 1973.

[King 69]
King J.C., A program verifier
Ph.D. thesis, Carnegie Mellon University, Pittsburgh(Pen.), septembre 1969.

[King 72]
King J.C., Floyd R.W., *An interpretation-oriented theorem prover over integers*
Journal of Computer and System Science, vol.16, n°4, août 1972.

[King 76]
King J.C., *Symbolic execution and program testing*
CACM, vol.19, n°7, juillet 1976.

[Klee 64]
Klee V., *On the number of vertices of a convex polytope*
Canadian Journal of Mathematics, n°16, 1964.

[Knuth 73]
Knuth D.E., The Art od Computer Programming, vol.3, *Sorting and searching*
Addison-Wensley, Reading(Mass.), 1973.

[Kuhn 56]
Kuhn H.W., *Solvability and consistency for linear equations and inequalities*
American Mathematical Monthly, n°63, 1956.

Lanery 66
Lanery E., Recherche d'un système générateur minimal d'un polyèdre convexe
Thèse de 3ème cycle, Université de Caen, juin 1966.

McMullen 70
Mc Mullen P., *The maximum numbers of faces of a convex polytope*
Mathematika, vol.17, décembre 1970.

Manas 68
Manas M., Nedoma J., *Finding all vertices of a convex polyhedron*
Numerische Mathematik, n°12, 1968.

Manna 74
Manna Z., Mathematical Theory of Computation
McGraw Hill, New-York, 1974.

Matheiss 73
Matheiss T.H., *An algorithm for determining irrelevant constraints and all vertices in systems of linear inequalities*
Operations Res., n°21, 1973.

Saaty 55
Saaty T.L., *The number of vertices of a polyhedron*
American Mathematical Monthly, n°62, 1955.

Sakarovitch 73
Sakarovitch M., Programmation linéaire
Université Scientifique et Medicale de Grenoble, 1973.

Schwartz 75
Schwartz J.T., *Automatic data structure choice in a langage of very high level*
CACM, vol.18, n°12, décembre 1975.

Simonnard 73
Simonnard M., Programmation linéaire
Dunod, Paris, 1973.

Suzuki 77
Suzuki N., Ishihita K., *Implementation of an array bound cheeker*
4th Symp. on Principles of Programming languages, Los Angeles, Janvier 1977.

Von Hohenbalken 78
Von Hohenbalken B., *Least distance methods for the scheme of polytopes*
Mathematical Programming, n°15, 1978.

[Wegbreit 74]
Wegbreit B., *The synthesis of loop predicate*
CACM, vol.17, n°2, février 1974.

[Wegbreit 75]
Wegbreit B., *Property extraction in well -founded property sets*
IEEE Trans. on Soft. Eng., SE-1, vol.3, septembre 1975.

[Welsh 77]
Welsh J., *Economic range checking in PASCAL*
Dept. of Comp. Science, Queens University, Belfast, Northern Ireland, octobre 77.

[Weyl 50]
Weyl H., *The elementary theory of convex polyhedra*
Annals of Mathematical study, n°24, Princeton, 1950.

[Wirth 76a]
Worth N., *Programming languages: what to demand and how to assess them*
Symp. on Software Engineering, Belfast, avril 1976.

[Wirth 76b]
Wirth N., Algorithms + Data structures = Programs
Prentice Hall, Englewood Cliffs (N.J.), 1976.

[Wulf 75]
Wulf W.A., London R.L., Shaw M., *Abstraction and verifications in Alphard, New directions in algorithmic languages*
IFIP W.G.2.1. (S.A. Schumann ed.), IRIA, 1975.

Dernière page d'une thèse

VU

Grenoble, le 22 Février 1979

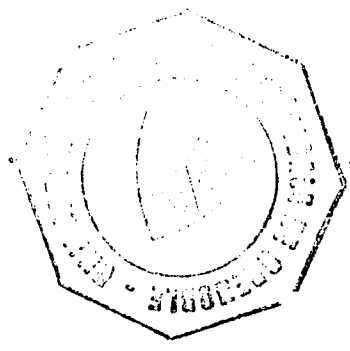
Le Président de la thèse

M Sokoloff

Vu, et permis d'imprimer,

Grenoble, le 27 février 1979

Le Président de l'Université
Scientifique et Médicale



J. Louis Cau

Dr G. CALI