USTL
FLANDRES ARTOIS

UNIVERSITE DES SCIENCES ET TECHNIQUES
DE LILLE FLANDRES ARTOIS

N° d'ordre : 178

THÈSE

Nouveau Régime

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

pour l'otention du titre de

DOCTEUR EN INFORMATIQUE

par

Philippe DEVIENNE

LES GRAPHES ORIENTES PONDERES : UN OUTIL POUR L'ETUDE DE LA TERMINAISON ET DE LA COMPLEXITE DANS LES SYSTEMES DE REECRITURES ET EN PROGRAMMATION LOGIQUE

Soutenue le 30 Novembre 1987 devant la Commission d'Examen

Président :	G.	COMYN
Rapporteurs :	B.	COURCELLE
	P.	DERANSART
Directeur de thèse :	M.	DAUCHET
Examinateurs :	A.	COLMERAUER
	J.P.	DELAHAYE
Invité :	L.	KOTT

UNIVERSITE DES SCIENCES
ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M. H. LEFEBVRE, M. PARREAU.

PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF, LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

PROFESSEUR EMERITE

M. A. LEBRUN

ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PARREAU, J. LOMBARD, M. MIGEON, J. CORTOIS.

PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

M. A. DUBRULLE.

PROFESSEURS - CLASSE EXCEPTIONNELLE

M. CONSTANT Eugène	Electronique
M. FOURET René	Physique du solide
M. GABILLARD Robert	Electronique
M. MONTREUIL Jean	Biochimie
M. PARREAU Michel	Analyse
M. TRIDOT Gabriel	Chimie appliquée

PROFESSEURS - 1ère CLASSE

M. BACCHUS Pierre	Astronomie
M. BIAYS Pierre	Géographie
M. BILLARD Jean	Physique du solide
M. BOILLY Bénoni	Biologie
M. BONNELLE Jean Pierre	Chimie-Physique
M. BOSCQ Denis	Probabilités
M. BOUGHON Pierre	Algèbre
M. BOURIQUET Robert	Biologie végétale
M. BREZINSKI Claude	Analyse numérique
M. BRIDOUX Michel	Chimie-Physique
M. CARREZ Christian	Informatique
M. CELET Paul	Géologie générale
M. CHAMLEY Hervé	Géotechnique
M. COEURE Gérard	Analyse
M. CORDONNIER Vincent	Informatique
M. DEBOURSE Jean Pierre	Gestion des entreprises
M. DHAINAUT André	Biologie animale
M. DOUKHAN Jean Claude	Physique du solide
M. DYMENT Arthur	Mécanique
M. ESCAIG Bertrand	Physique du solide
M. FAURE Robert	Mécanique
M. FOCT Jacques	Métallurgie
M. FRONTIER Serge	Ecologie numérique
M. GRANELLE Jean jacques	Sciences Economiques
M. GRUSON Laurent	Algèbre
M. GUILLAUME Jean	Microbiologie
M. HECTOR Joseph	Géométrie
M. LABLACHE-COMBIER Alain	Chimie organique
M. LACOSTE Louis	Biologie végétale
M. LAVEINE Jean Pierre	Paléontologie
M. LEHMANN Daniel	Géométrie
Mme LENOBLE Jacqueline	Physique atomique et moléculaire
M. LEROY Jean Marie	Spectrochimie
M. LHOMME Jean	Chimie organique biologique
M. LOMBARD Jacques	Sociologie
M. LOUCHEUX Claude	Chimie physique
M. LUCQUIN Michel	Chimie physique
M. MACKE Bruno	Physique moléculaire et rayonnements atmosphériques
M. MIGEON Michel	E.U.D.I.L.
M. PAQUET Jacques	Géologie générale
M. PETIT Francis	Chimie organique
M. POUZET Pierre	Modélisation - Calcul scientifique
M. PROUVOST Jean	Minéralogie
M. RACZY Ladislas	Electronique
M. SALMER Georges	Electronique
M. SCHAMPS Joel	Spectroscopie moléculaire
M. SEGUIER Guy	Electrotechnique
M. SIMON Michel	Sociologie
Mle SPIK Geneviève	Biochimie
M. STANKIEWICZ François	Sciences Economiques
M. TILLIEU Jacques	Physique théorique
M. TOULOTTE Jean Marc	Automatique
M. VIDAL Pierre	Automatique
M. ZEYTOUNIAN Radyadour	Mécanique

PROFESSEURS - 2ème CLASSE

M. ALLAMANDO Etienne	Composants électroniques
M. ANDRIES Jean Claude	Biologie des organismes
M. ANTOINE Philippe	Analyse
M. BART André	Biologie animale
M. BASSERY Louis	Génie des procédés et réactions chimiques
Mme BATTIAU Yvonne	Géographie
M. BEGUIN Paul	Mécanique
M. BELLET Jean	Physique atomique et moléculaire
M. BERTRAND Hugues	Sciences Economiques et Sociales
M. BERZIN Robert	Analyse
M. BKOUCHE Rudolphe	Algèbre
M. BODARD Marcel	Biologie végétale
M. BOIS Pierre	Mécanique
M. BOISSIER Daniel	Génie civil
M. BOIVIN Jean Claude	Spectrochimie
M. BOUQUELET Stéphane	Biologie appliquée aux enzymes
M. BOUQUIN Henri	Gestion
M. BRASSELET Jean Paul	Géométrie et topologie
M. BRUYELLE Pierre	Géographie
M. CAPURON Alfred	Biologie animale
M. CATTEAU Jean pierre	Chimie organique
M. CAYATTE Jean Louis	Sciences Economiques
M. CHAPOTON Alain	Electronique
M. CHARET Pierre	Biochimie structurale
M. CHIVE Maurice	Composants électroniques optiques
M. COMYN Gérard	Informatique théorique
M. COQUERY Jean Marie	Psychophysiologie
M. CORIAT Benjamin	Sciences Economiques et Sociales
Mme CORSIN Paule	Paléontologie
M. CORTOIS Jean	Physique nucléaire et corpusculaire
M. COUTURIER Daniel	Chimie organique
M. CRAMPON Norbert	Tectolique Géodynamique
M. CROSNIER Yves	Electronique
M. CURGY Jean jacques	Biologie
Mle DACHARRY Monique	Géographie
M. DAUCHET Max	Informatique
M. DEBRABANT Pierre	Géologie appliquée
M. DEGAUQUE Pierre	Electronique
M. DEJAEGER Roger	Electrochimie et Cinétique
M. DELORME Pierre	Physiologie animale
M. DELORME Robert	Sciences Economiques
M. DEMUNTER Paul	Sociologie
M. DENEL Jacques	Informatique
M. DE PARIS Jean Claude	Analyse
M. DEPREZ Gilbert	Physique du solide - Cristallographie
M. DERIEUX Jean Claude	Microbiologie
Mle DESSAUX Odile	Spectroscopie de la réactivité chimique
M. DEVRAINNE Pierre	Chimie minérale
Mme DHAINAUT Nicole	Biologie animale
M. DHAMELINCOURT Paul	Chimie physique
M. DORMARD Serge	Sciences Economiques
M. DUBOIS Henri	Spectroscopie hertzienne
M. DUBRULLE Alain	Spectroscopie hertzienne
M. DUBUS Jean Paul	Spectrométrie des solides

M. DUPONT Christophe	Vie de la firme (I.A.E.)
Mme EVRARD Micheline	Génie des procédés et réactions chimiques
M. FAKIR Sabah	Algèbre
M. FAUQUEMBERGUE Renaud	Composants électroniques
M. FONTAINE Hubert	Dynamique des cristaux
M. FOUQUART Yves	Optique atmosphérique
M. FOURNET Bernard	Biochimie structurale
M. GAMBLIN André	Géographie urbaine, industrielle et démographie
M. GLORIEUX Pierre	Physique moléculaire et rayonnements atmosphériques
M. GOBLOT Rémi	Algèbre
M. GOSSELIN Gabriel	Sociologie
M. GOUDMAND Pierre	Chimie physique
M. GOURIEROUX Christian	Probabilités et statistiques
M. GREGORY Pierre	I.A.E.
M. GREMY Jean Paul	Sociologie
M. GREVET Patrice	Sciences Economiques
M. GRIMBLOT Jean	Chimie organique
M. GUILBAULT Pierre	Physiologie animale
M. HENRY Jean Pierre	Génie mécanique
M. HERMAN Maurice	Physique spatiale
M. HOUDART René	Physique atomique
M. JACOB Gérard	Informatique
M. JACOB Pierre	Probabilités et statistiques
M. JEAN Raymond	Biologie des populations végétales
M. JOFFRE Patrick	Vie de la firme (I.A.E.)
M. JOURNEL Gérard	Spectroscopie hertzienne
M. KREMBEL Jean	Biochimie
M. LANGRAND Claude	Probabilités et statistiques
M. LATTEUX Michel	Informatique
Mme LECLERCQ Ginette	Catalyse
M. LEFEBVRE Jacques	Physique
M. LEFEVRE Christian	Pétrologie
Mle LEGRAND Denise	Algèbre
Mle LEGRAND Solange	Algèbre
M. LEGRAND Pierre	Chimie
Mme LEHMANN Josiane	Analyse
M. LEMAIRE Jean	Spectroscopie hertzienne
M. LE MAROIS Henri	Vie de la firme (I.A.E.)
M. LEROY Yves	Composants électroniques
M. LESENNE Jacques	Systèmes electroniques
M. LHENAFF René	Géographie
M. LOCQUENEUX Robert	Physique théorique
M. LOSFELD Joseph	Informatique
M. LOUAGE Francis	Electronique
M. MAHIEU Jean Marie	Optique - Physique atomique
M. MAIZIERES Christian	Automatique
M. MAURISSON Patrick	Sciences Economiques et Sociales
M. MESMACQUE Gérard	Génie Mécanique
M. MESSELYN Jean	Physique atomique et moléculaire
M. MONTEL Marc	Physique du solide
M. MORCELLET Michel	Chimie organique
M. MORTREUX André	Chimie organique
Mme MOUNIER Yvonne	Physiologie des structures contractiles
M. NICOLE Jacques	Spectrochimie
M. NOTELET Francis	Systèmes électroniques
M. PARSY Fernand	Mécanique
M. PECQUE Marcel	Chimie organique
M. PERROT Pierre	Chimie appliquée

M. PERTUZON Emile	Physiologie animale
M. PONSOLLE Louis	Chimie physique
M. PORCHET Maurice	Biologie animale
M. POSTAIRE Jack	Informatique industrielle
M. POVY Lucien	Automatique
M. RICHARD Alain	Biologie animale
M. RIETSCH François	Physique des polymères
M. ROBINET Jean CLaude	EUDIL
M. ROGALSKI Marc	Analyse
M. ROY Jean Claude	Psychophysiologie
Mme SCHWARZBACH Yvette	Géométrie
M. SLIWA Henri	Chimie organique
M. SOMME Jean	Géographie
M. STAROSWIECKI Marcel	Informatique
M. STERBOUL François	Informatique
M. TAILLIEZ Roger	Génie alimentaire
M. THERY Pierre	Systèmes électroniques
M. THIEBAULT François	Sciences de la terre
M. THUMERELLE Pierre	Démographie - Géographie Humaine
Mme TJOTTA Jacqueline	Mathématiques
M. TOURSEL Bernard	Informatique
M. TREANTON Jean rené	Sociologie du Travail
M. TURREL Georges	Spectrochimie infrarouge et Raman
M. VANDORPE Bernard	Chimie minérale
M. VASSEUR Christian	Automatique
M. VAST Pierre	Chimie inorganique
M. VERBERT André	Biochimie
M. VERNET Philippe	Génétique
M. WACRENIER Jean Marie	Electronique
M. WALLART Francis	Spectrochimie infrarouge et Raman
M. WARTEL Michel	Chimie inorganique
M. WATERLOT Michel	Géologie générale
M. WEINSTEIN Olivier	Analyse économique de la recherche et développement
M. WERNER Georges	Informatique théorique
M. WOZNIAK Michel	Spectrochimie
Mme ZINN JUSTIN Nicole	Algèbre

J'exprime à Messieurs les Membres du jury ma profonde gratitude :

à Alain COLMERAUER et Laurent KOTT qui m'ont fait l'honneur de faire partie du jury, qu'ils en soient vivement remerciés,

à Gérard COMYN dont le rôle de Président traduit l'intérêt et la confiance amicales qu'il a toujours su témoigné aux jeunes chercheurs et dont le dynamisme est une source d'encouragements,

à Bruno COURCELLE dont l'intérêt pour ce travail a été constant et qui, en dépit de ses nombreuses activités, a accepté d'en être un rapporteur attentif,

à Max DAUCHET qui a assumé la direction de cette thèse pendant 4 ans et dont les idées originales et les remarques fécondes ont été les éléments-moteurs; je lui exprime mon plus profond respect,

à Jean Paul DELAHAYE, dont les compétences et la connaissance en programmation logique m'ont été très précieuses,

à Pierre DERANSART qui m'a fait l'honneur de bien vouloir juger les résultats de cette thèse et d'en être le rapporteur.

J'exprime bien-sûr ma gratitude à Patrick LEBEGUE, co-auteur de ce travail, avec lequel j'ai eu grand plaisir à travailler.

Les Graphes Orientés Pondérés : un outil pour l'etude de la terminaison et de la complexité dans les systèmes de réécritures et en programmation logique .

Chapitre 1

Introduction.

1) Situation du thème

Notre étude a pour objet le problème de l'arrêt et de la complexité en temps pour une classe simple de programmes genre PROLOG. Nous avons analysé statiquement le comportement des programmes récursifs et, en particulier, du plus simple d'entre eux, que nous avons appelé le **Tant Que PROLOG** (sic) , composé du fait f, de la règle récursive r et d'une question :

Tant Que : f Prédicat (. .) -> ;
r Prédicat (. .) -> Prédicat (. .) ;
Prédicat (. .) ?

Exemple : f ENTIER (Zéro) -> ; [0 ∈ N]
r ENTIER (Succ(x)) -> ENTIER (x) ; [x+1 ∈ N SI x ∈ N]
ENTIER (Succ(Succ(Zéro))) ? [2 ∈ N] ?

(Nous avons choisi la syntaxe du Prolog 2 pour les règles, de plus les variables seront notées en minuscules, les symboles de fonction commenceront toujours par une majuscule).

La recherche d'une solution en chaînage arrière est une suite d'inférences de la question par la règle r, *tant que* il y a incompatibilité avec le fait f , ce processus pouvant bien-sûr être infini.

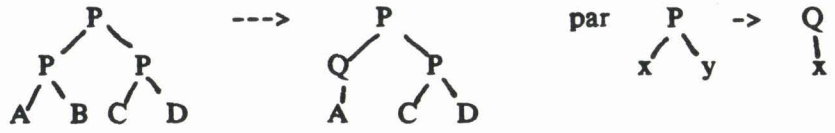
Un tel programme PROLOG est assimilable à un système de réécritures :

f : α ->
r : β -> γ
T ?

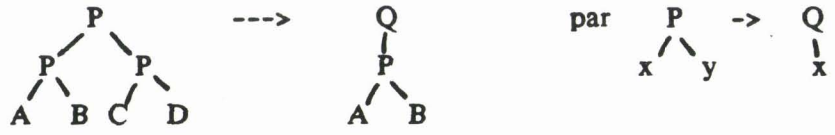
Recherche d'un enchaînement de réécritures de la question T :
$T \text{ -- } r^n . f \text{ -->}$
où T , α , β , γ sont des arbres finis avec variables

C'est dans ce cadre que nous avons abordé notre étude. Mais il est nécessaire, ici, de parler de **Système de Réécriture en Tête.**

On s'interdit par exemple :



Il n'y a au plus qu'une façon d'appliquer une règle pour une question donnée :



Dans le cadre des Systèmes de Réécritures, de nombreuses études ont été faites concernant l'arrêt de systèmes récursifs simples (Voir l'excellent survey de Dershowitz) .
La simplicité de telles règles n'est qu'apparente, comme en témoigne le nombre de travaux qui y sont consacrés. Enumérons quelques uns d'entre eux :

- Systèmes de Réécritures en Tête plus particuliers que le nôtre :

Dans le cadre des Bases de Données Déductives, le langage DATALOG est basé sur les *Clauses de Horn sans symbole de fonction* : Buys (x , y) -> Cheap (x , y) Likes (x , y) ;
Un programme Datalog est dit *bornable* ("bounded") si on peut éliminer ses récursions. Cette propriété est indécidable, même pour les programmes *linéaires* (i.e. chaque corps de règle ne contient au plus qu'un prédicat récursif) [Gaifman , Mairson 87]. Cette propriété *pourrait être* décidable pour les programmes ne contenant qu'une seule règle [Ionnadis 85] , [Naughton 86] .

- Systèmes de Réécritures quelconques , donc plus généraux que le nôtre :

Un ensemble de règles R est dit satisfaisant la *propriété de terminaison* si tout terme clos ne peut pas être réécrit infiniment : ∀ T terme clos , T --R--> toujours finiment .
Ce problème a été démontré indécidable pour 3 règles [Lipton , 77] , pour 2 règles [Deshowitz 85] et enfin même si R ne contient qu'une seule règle [Dauchet 87] .

- Systèmes de Réécritures en Tête : Outre le problème de la terminaison, on se pose celui de l'arrêt pour un terme T donné : R et T donnés satisfont la propriété de terminaison si T ne peut être réécrit que finiment par les règles de R .
Dans un tout autre cadre que celui de la programmation logique, un résultat partiel a été démontré sur cette question [Pereira 84], à propos des processus communicants. Très récemment, l'étude du bouclage d'une règle (ou d'un cycle de règles) a été introduite comme outil majeur d'analyse statique de programmes PROLOG, dans le cadre de la réalisation d'un environnement pour la programmation logique [Pelhat 87]. Des conditions suffisantes de bouclage y sont énoncées.

C'est le problème de terminaison d'une règle récursive en tête que nous étudions, ainsi que celui de l'existence de solutions au Tant Que Prolog. Nous introduisons des outils d'analyse statique qui permettent de décider de l'arrêt d'une boucle, dans un cadre plus large (les termes non nécessairement clos) avec un facteur de complexité linéaire.

Dans ce qui suit, nous nous limitons aux problèmes de décision et de complexité pour le Tant Que Prolog. A ce propos, on peut noter que l'arrêt d'un programme Prolog d'une classe un tout petit plus large est indécidable [Lebègue] (on peut codifier le problème de POST par une série de faits et une règle de la forme : P(ß) -> P(ɣ) Q(δ) ;) .

Mais les résultats obtenus sur le Tant Que, par leur précision quantitative, peuvent permettre l'extension de l'étude à des classes de programmes Prolog plus vastes , par exemple le très classique programme des chemins d'un graphe :

```
EDGE ( . , . )  ->  ;
EDGE ( . , . )  ->  ;
PATH ( x.nil )  ->  ;
PATH ( x.y.z )  ->  EDGE ( x , y )  PATH (y.z)  ;
```

en leur apportant des propriétés de décidabilité et de complexité [Lebègue].

Ces résultats permettent aussi pour certains programmes récursifs :

. de vérifier facilement leur terminaison (*aide aux programmeurs*)

. de les compiler sous une forme sympathique (*rapidité*)

. d'améliorer la stratégie de résolution PROLOG dans ces appels récursifs (*complétude*).

Il est important de noter que tous ces résultats ont pu être obtenus grâce à de nouveaux objets syntaxiques que nous avons introduits et étudiés, les *Graphes Orientés Pondérés* . Outils de démonstration, ils permettent aussi de généraliser la notion d'arbres infinis [Courcelle].

Informellement, un Graphe Orienté Pondéré est un graphe dont les arcs sont pondérés par des entiers relatifs. Lors du dépliage, les pondérations le long d'une branche sont sommées et cette somme indice la feuille variable de la branche. Les arbres ainsi obtenus peuvent être irrationnels. Nous étudions leurs propriétés formelles (interprétation finie et infinie, substitution, équivalence, unification, . . .) .

2) Présentation informelle des principaux résultats.

Les Graphes Orientés Pondérés permettent d'obtenir des algorithmes de décision et d'analyse statique de complexité de programmes PROLOG simples de la forme :

f Prédicat (. .) -> ;

r Prédicat (. .) -> Prédicat (. .) ;

Prédicat (. .) ?

que nous appelons des programmes Tant Que (sic), car leur résolution est exprimable par " Inférer la question avec la règle r, *tant que* il y a incompatibilité avec le fait f ".

La classe étudiée peut paraître simple. En fait nos résultats chapeautent de nombreuses études partielles sur le sujet, et ce travail peut être intégré à un environnement de programmation Prolog pour de larges classes de programmes logiques.

Si l'arrêt d'un programme Prolog non récursif est toujours vrai (trivial), le problème est ouvert dès que l'on ajoute une règle récursive. Par exemple, pour le programme suivant :

r : Prédicat(.) -> Prédicat (.) ;

T ?

peut-on décider de son arrêt, pour tout but T ou seulement certains d'entre eux.

Si ces questions sont résolues dans le cas d'une "décroissance" du but lors de la réécriture, elles restent ouvertes pour une règle quelconque.

Formalisation

L'application d'une règle, notée g -> d est formalisable par la séquence suivante de réécritures :

$$g_1 \rightarrow d_1 \quad V \quad g_2 \rightarrow d_2 \quad V \quad g_3 \rightarrow \ldots \rightarrow d_n \quad V \quad g_{n+1} \rightarrow \ldots$$

les variables étant muettes, on doit les renommer avec chaque nouvelle réécriture. Pour la $i^{ème}$ utilisation de la règle g -> d , nous appliquons celle dont les variables sont indicées de i :

$i^{ème}$ réécriture : utilisation de la règle $g_i \rightarrow d_i$.

Pour la règle r suivante exprimant l'associativité de l'opérateur d'addition

$$(x + y) + z \rightarrow x + (y + z)$$

nous utilisons les règles r_i : $(x_i + y_i) + z_i \rightarrow x_i + (y_i + z_i)$

Introduction des Graphes Orientés Pondérés

Un Graphe Orienté Pondéré est un graphe :

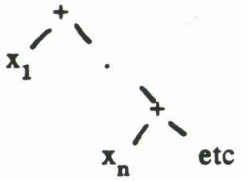
. avec une racine pondérée d'un entier relatif,

. dont les arcs sont aussi pondérés par des entiers relatifs ,

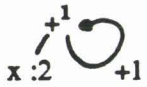
. et dont certaines des variables possèdent une période.

Lors du dépliage du GOP, les pondérations le long d'une branche sont sommées et cette somme sert d'indice à la feuille variable de la branche. Les arbres ainsi obtenus peuvent être irrationnels :

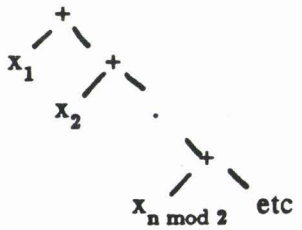
dont le déplié est



Pour tenir compte de certains phénomènes périodiques, les variables d'un GOP peuvent posséder une période ; dans l'interprété, leur indice est alors calculée modulo cette période :



le déplié est

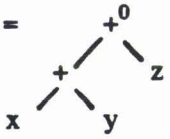


On généralise l'interprétation d'un Gop en considérant le déplié pour une pondération d'entrée supplémentaire. De cette manière, pour notre règle récursive g -> d :

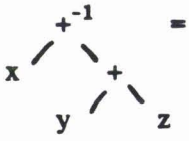
. le déplié du Gop g^0 pour la pondération d'entrée i est l'arbre g_i

. le déplié du Gop d^{-1} pour la pondération d'entrée i est l'arbre d_{i-1}.

Ex : g^0 =

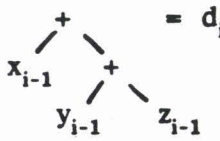


et



= d^{-1}

Le déplié de d^{-1} avec la pondération d'entrée i est :



= d_{i-1}

, toutes les pondérations rencontrées sont sommées (i.e. pondération d'entrée, pondération de la racine, pondérations des arcs) ; cette somme est ici identique sur toutes les branches.

Unification de Graphes Orientés Pondérés

Nous introduisons un algorithme d'unification des GOPS avec ou sans test d'occurrence, ceci en généralisant l'algorithme sur les Graphes Orientés classiques. Il repose sur les deux remarques suivantes :

1) L'unifié de deux Gops G et G' de même sommet racine s et de pondérations racines respectivement p et p', est le Gop G (ou G') dont toutes les variables sont de période |p-p'| .

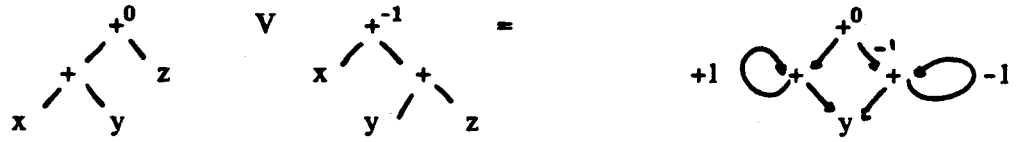
$x^1 \ V \ x^{-1} \quad = \quad x^1 : 2 \ = \ x^{-1} : 2$

2) On ne change rien à l'unification de deux Gops G et G' de sommets racines s et s' et de pondérations p et p', si l'on remplace dans G et G', tout arc allant vers le sommet s' avec une pondération q, par un arc allant sur s avec une pondération corrigée de (p-p') : q +(p-p').

Ex : 

En intégrant ces modifications de pondérations et ces calculs de période, on construit un algorithme en tout point identique à celui sur les Graphes Orientés classiques.

Ex 1 :



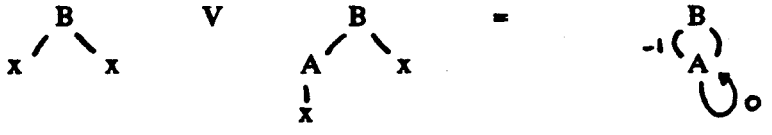
(Si l'on retire toutes les pondérations, cette égalité représente l'unification dans les Graphes)

Ex 2 :

(Même remarque si l'on retire la période et les pondérations dans cette égalité)

Le test d'occurrence dans l'unification des GOPS est équivalent à une détection de boucles de pondération nulle (dans les graphes, c'est la simple détection de boucles dans l'unifié). Si les exemples ci-dessus satisfont le test d'occurrence, il n'est pas vérifié dans l'unification suivante.

Ex 3 :



l'unifié contient une boucle sur le sommet A de pondération nulle.

Gop caractéristique d'une règle récursive

Soit la règle g -> d, son application récursive est formalisée par

. la séquence de réécritures : $g_1 \to d_1 \ V \ g_2 \to d_2 \ V \ g_3 \to \ldots \to d_n \ V \ g_{n+1} \to \ldots$

. c-à-d par la recherche d'une instanciation des arbres g_i et d_{i-1} les rendant, pour chaque i, égaux deux à deux.

. ou encore le Gop $g^0 \ V \ d^{-1}$, car les dépliés, pour chaque i, de g^0 et d^{-1} sont les arbres g_i et d_{i-1}.

Les exemples 1 et 2 (page précédente) correspondent aux Gops caractéristiques de l'associativité et de la règle FRERE (x,y) -> FRERE (y,x) .

De nombreuses informations concernant le comportement de la règle g -> d peuvent être extraites de ce Gop $g^0 \ V \ d^{-1}$:

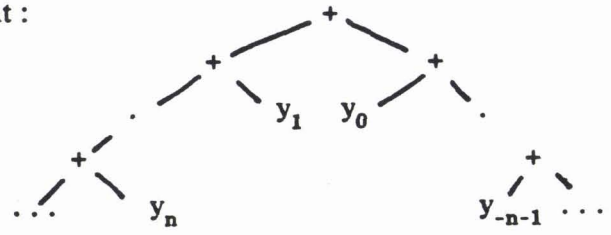
1) Décidabilité de la terminaison d'une règle pour tout terme quelconque.

$\exists$ T (non nécessairement clos) se réécrivant infiniment par g -> d ssi $g^0 \ V \ d^{-1}$ existe (avec ou sans test d'occurrence suivant qu'on applique une résolution avec ou sans test d'occurrence).

La règle associative et la règle FRERE engendrent des résolutions infinies avec ou sans test d'occurrence, car $g^0 \ V \ d^{-1}$ existe . Par contre, la règle de l'exemple 3 engendre toujours une résolution finie avec test d'occurrence : g^0 et d^{-1} ne sont pas unifiables avec test d'occurrence.

2) Plus petit arbre invariant

Le déplié de $g^0 \ V \ d^{-1}$ est le plus petit arbre se réécrivant par g -> d en un arbre équivalent : pour l'associativité .



C'est, ici, un arbre irrationnel; il se réécrit en un terme équivalent, le même en incrémentant de 1 tous les indices.

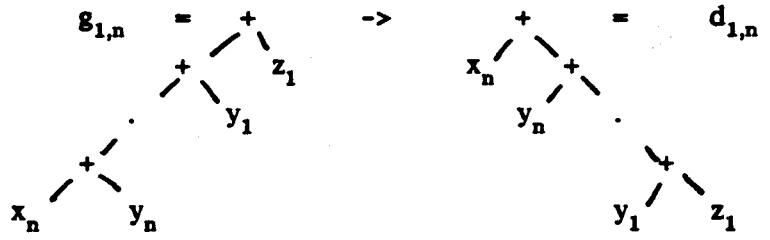
3) Phénomènes périodiques.

Les périodes des variables du Gop caractéristique formalisent la périodicité de la règle :

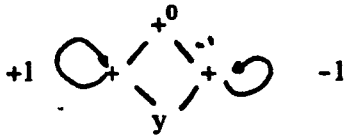
FRERE(x,y) -> FRERE(y,x) est de période 2.

4) Croissance linéaire

Intéressons nous à l'évolution pas à pas de cette récursivité et notons $g_{1,n}$ -> $d_{1,n}$, la règle équivalente à n applications de la règle g -> d . Par exemple, appliquer n fois notre règle associative est équivalent à appliquer une fois la règle suivante :



la croissance des arbres $g_{1,n}$ et $d_{1,n}$ est mesurable par lecture du Gop, grâce aux boucles positives et négatives qu'il contient :



, la branche de pondération croissante du Gop est la branche croissante de $g_{1,n}$, de même la branche de pondération décroissante du Gop, celle croissante dans $d_{1,n}$.

Même si certains phénomènes parasites sur les bords perturbent cette lecture, on peut dire en gros que la présence dans g^0 V d^{-1} d'une boucle positive de longueur b et de pondération p caractérise une croissance de la branche correspondante dans $g_{1,n}$ d'une hauteur b/p à chaque nouvelle réécriture. De même dans $d_{1,n}$ si la pondération de la boucle est négative, la croissance sera de b/(-p) . Si la pondération de cette boucle est nulle, la croissance est immédiatement infinie (i.e. le test d'occurrence n'est pas vérifié) .

La croissance de chacune des branches de $g_{1,n}$ et $d_{1,n}$ est linéairement dépendant de n.

5) Complexité constante.

Si le Gop caractéristique ne contient aucune boucle, alors le Tant Que Prolog est de complexité constante, c-à-d qu'il existe un programme non récursif qui lui est équivalent.

Exemple : FRERE (Jean , Xavier) -> ;
FRERE (x , y) -> FRERE (y , x) ;

équivalent à

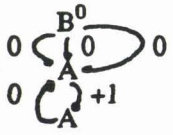
FRERE (Jean , Xavier) -> ;
FRERE (Xavier , Jean) -> ;

La difficulté majeure de ce travail a été de montrer, dans le cadre d'une interprétation finie des Gops (Chap 4), que ces effets parasites sur les bords sont toujours de taille bornée et sont même constants à partir d'un certain nombre de réécritures.

Aussi, la présence d'une boucle positive ou négative est une condition nécessaire de croissance de certaines branches de l'arbre $g_{1,n}$ ou $d_{1,n}$, et non suffisante. Mais on peut vérifier à travers les effets de bord, si les boucles du Gop sont effectives ou non, c-à-d si les branches correspondantes dans $g_{1,n}$ et $d_{1,n}$ croissent linéairement ou sont constantes .

L'exemple suivant illustre ces phénomènes parasites :

Son Gop caractéristique est :



Cette règle contient une boucle positive, pourtant elle a la particularité d'être invariante pour les variables x_1 et y_n :

La boucle positive n'est effective que sur la deuxième branche de $g_{1,n}$.

Principaux résultats de décidabilité

Grâce à cette mise en évidence de la croissance linéaire des termes $g_{1,n}$ et $d_{1,n}$, on peut montrer de façon immédiate :

. la décidabilité de la terminaison d'une règle récursive pour tout terme clos :

∀ T clos , T --r--> finiment

. la décidabilité de la terminaison d'une règle récursive pour un terme clos :

T clos donné , T --r--> finiment

Nous démontrons même que ceci reste vrai pour les termes linéaires (avec variables, mais une seule occurrence de chaque variable) et que cette propriété est décidable avant un nombre de réécritures linéairement dépendant de la hauteur du terme T.

. la décidabilité de l'existence de solutions au Tant Que Prolog pour les questions T et faits f linéaires, en un nombre de réécritures linéairement dépendant de la hauteur de la question T.

. la décidabilité de l'arrêt du programme Tant Que si la question T et le fait sont linéaires, dans le cadre d'une recherche de toutes les solutions (comme en PROLOG) .

Certains résultats plus précis peuvent être obtenus dans le cas de cloture de certaines branches de la question, celles relatives aux chemins de pondération croissante dans le Gop caractéristique .

3) Guide de Lecture

Le Tant Que Prolog

Soit le programme Tant Que où α, β, γ et T sont des arbres finis avec variables :

f	α -> ;	" α est vrai "
r	β -> γ ;	" Si γ Alors β "
	T ?	" $\exists$ T vrai "

Une solution à cette question T est obtenue par des inférences successives de la règle r , tant qu'il y a incompatibilité avec le fait f. En termes de réécritures, T est une solution de ce programme si et seulement si on peut réécrire n fois T par la règle r , puis une fois par f :

$$T \text{ solution} \quad <=> \quad \exists\ n \in N \text{ , tel que } T \text{ --}r^n.f\text{-->}$$

une réécriture étant une instanciation particulière σ des variables de la règle :

$$\forall\ \sigma \text{ substitution , } \sigma(\beta) \text{ --r--> } \sigma(\gamma)$$

Cette recherche peut ne jamais se terminer si la règle récursive peut être appliquée infiniment.

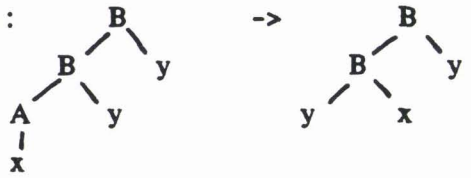
On peut reconnaître ces règles *bouclantes* sur des exemples simples :

Frère (x , y) -> Frère (y , x) ; " *x est le Frère de y SI y est le Frère de x* "

Boire (Goutte) -> Boire (Café) ; " *Pos d'doute, après ch'café, in bot l'goutte* "

la première règle r peut être itérée indéfiniment, puisqu'elle se contente de permuter ses arguments, tandis que la seconde n'est utilisable qu'une seule fois, car il y a incompatibilité entre les deux membres de la règle : Boire (Goutte) et Boire(Café) ne sont pas unifiables .

Mais, dans le cas général, la non-linéarité des arbres tête ou corps de règle, la présence de variables uniquement d'un côté de la règle, et les permutations de ces variables lors de la réécriture annihilent rapidement toute intuition sur la finitude de cette réécriture . La règle ci-dessous ne peut être appliquée que deux fois (avec test d'occurrence) ou indéfiniment (sans test d'occurrence) :



Pour bien formaliser l'application récursive de telles règles, il faut noter que les variables d'une règle n'ont un sens qu'à l'intérieur de cette règle : elles sont muettes, c-à-d différentes à chaque nouvelle réécriture. Pour cela, nous ajoutons un indice, le numéro de la réécriture : à la $i^{ème}$ utilisation de β -> γ , nous appliquons la règle β_i -> γ_i , en indiçant de i toutes les variables de β et γ.

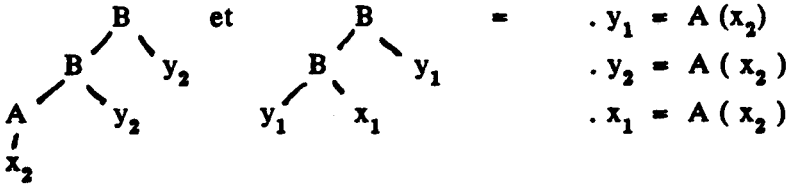
Caractérisation finie.- Il existe des arbres T tels que $T \text{ --}r^n\text{--}>$ ssi il existe une substitution σ_n vérifiant : $\forall\, i \in [2,n]\,,\ \sigma_n(\beta_i) = \sigma_n(\gamma_{i-1})$

Caractérisation infinie.- La règle $\beta \to \gamma$ satisfait la propriété de terminaison quel que soit T, ssi il n'existe pas σ_Z telle que : $\forall\, i \in Z\,,\ \sigma_Z(\beta_i) = \sigma_Z(\gamma_{i-1})$
Dans le cas contraire, le plus petit arbre se réécrivant en un arbre équivalent est $\sigma_Z(\beta_i)$.

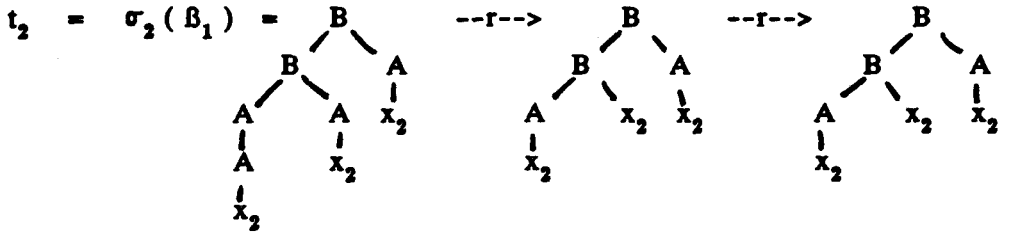
Avec cet indiçage de la règle, il existe un arbre T pouvant être réécrit 2 fois si et seulement si les arbres γ_1 et β_2 sont unifiables. En effet, soit σ_2, leur m.g.u. :

$$\sigma_2(\beta_1) \text{ --r--}> \sigma_2(\gamma_1) = \sigma_2(\beta_2) \text{ --r--}> \sigma_2(\gamma_1).$$

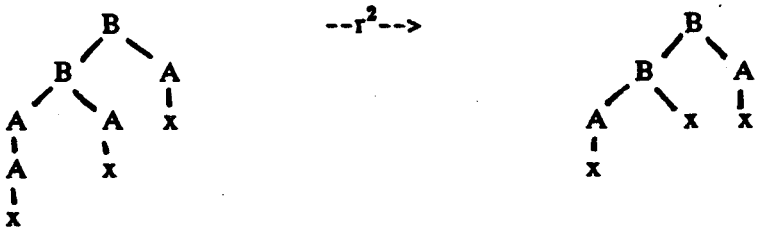
Sur notre exemple, le m.g.u. σ_2 de



Le plus petit arbre T_2 se réécrivant 2 fois par la règle récursive est donc :



Appliquer deux fois notre règle équivaut à appliquer une fois :



Trois réécritures sont impossibles avec test d'occurrence, et sans test d'occurrence la substitution σ_3 est l'instanciation de toutes les variables par l'arbre infini A^∞. Appliquer plus de trois fois la règle r équivaut à utiliser la règle *invariante* suivante :

$$B(B(A^\infty, A^\infty), A^\infty) \quad \to \quad B(B(A^\infty, A^\infty), A^\infty)$$

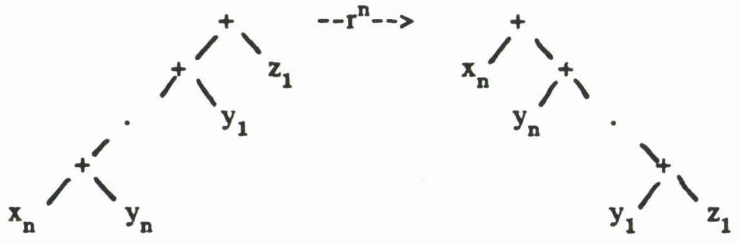
La règle satisfait donc la propriété de terminaison pour tout T, si on applique le test d'occurrence à chaque réécriture, sinon elle devient invariante dès la $3^{ème}$ réécriture.

Justification intuitive des Graphes Orientés Pondérés

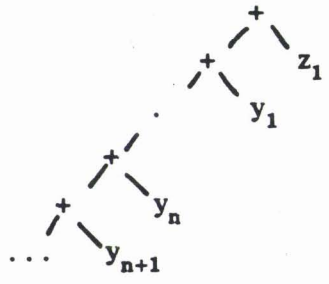
L'indiçage des variables de la règle par la profondeur de réécritures est l'une des clés de la résolution de notre problème et justifie la présentation et l'étude des Graphes Orientés Pondérés. Illustrons-le sur l'associativité de l'opérateur " + " : (x + y) + z = x + (y + z) .

r : ß = 

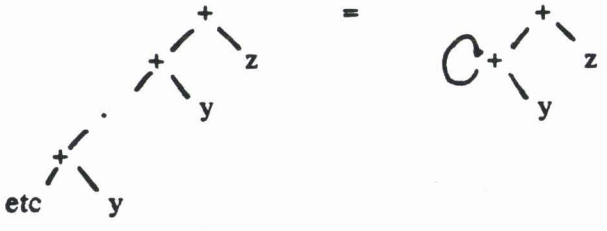
Appliquer n fois cette règle, revient à appliquer la règle suivante :



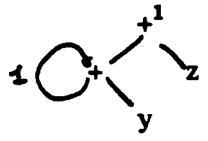
Intéressons-nous à l'arbre de gauche. Tout arbre se réécrivant n fois par la règle r est une instanciation de cet arbre gauche. La hauteur et le nombre de variables de celui-ci dépend directement du nombre de réécritures n . Construire le plus petit arbre pouvant se réécrire infiniment par r revient à faire tendre n vers l'infini et l'on obtient :



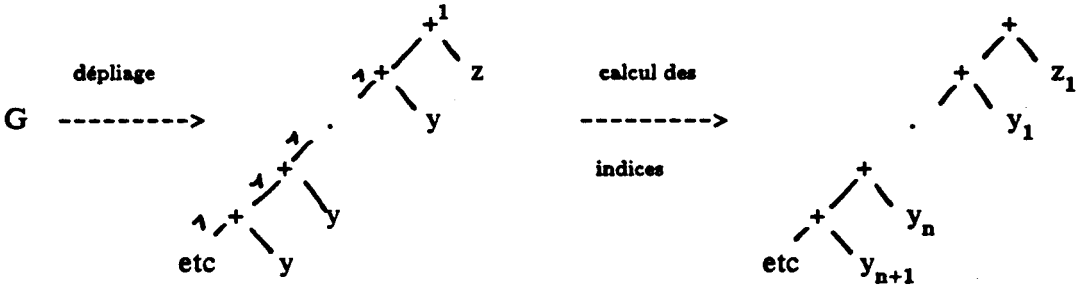
Ce n'est pas un arbre rationnel, car il possède une infinité de variables indicées distinctes. On ne peut donc pas le représenter sous la forme d'un graphe orienté classique, mais en effaçant tous les indices, on le rationalise :



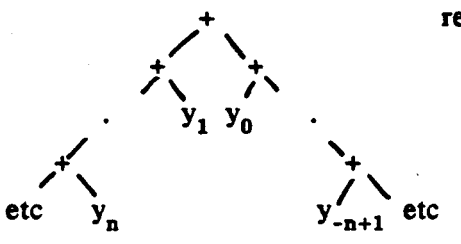
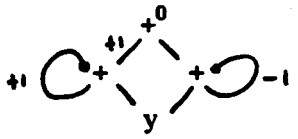
Malheureusement, des informations fondamentales sur le comportement de la règle seront perdues lors de cet effacement. Aussi l'idée est-elle d'ajouter des pondérations sur les arcs d'un Go pour tenter de caractériser cette irrationalité dûe au nombre infini de variables , ces pondérations servant à indicer les variables du déplié :

G =  représentera cet arbre irrationnel

Lors du dépliage de ce graphe, les pondérations rencontrées sur une branche s'ajoutent et indicent les variables.



Le plus petit arbre T se réécrivant en un arbre équivalent est pour cette règle associative :

 représenté par le GOP 

Certains phénomènes périodiques sont inhérents aux réécritures récursives : le prédicat *Frère* présenté plus haut, ou la commutativité de l'opération d'addition " $x + y = y + x$ " en sont des exemples simples. Considérons une Partie d'Echecs notée sous la forme d'une liste du nom des joueurs et de leurs coups :

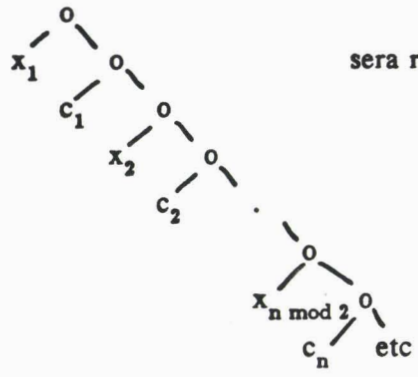
Karpov . Cf3 . Kasparov . Cf6 . Karpov . d4 . . . . . Kasparov . Te1 . Karpov . Mat

Ces parties sont reconnaissables par un programme Tant Que :

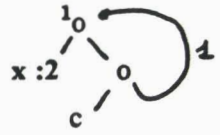
Partie (x , y , x . Mat)) -> ; " *x a perdu* "

Partie (x , y , x . c . z) -> Partie (y , x , z) ; " *x joue le coup c , à y de jouer* "

Une partie infinie est une instanciation de l'arbre irrationnel suivant, les joueurs x_1 et x_2 jouant à tour de rôle des coups c_n quelconques :



sera représenté par le Gop



x:2 signifie que l'indice des variables x_i est calculé modulo 2.

Chaque variable d'un Graphe Orienté Pondéré pourra ainsi posséder une période, appelée *congruence d'indices* ; lors du dépliage, la somme des pondérations de la branche sera traitée modulo la congruence de la variable .

Chap 2 - Définition des Graphes Orientés Pondérés

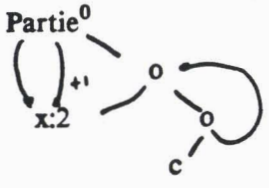
Il est bien connu que le dépliage des graphes orientés génère des arbres infinis rationnels. Nous enrichissons ces objets syntaxiques en pondérant les arcs et en associant aux variables une période appelée congruence d'indices .

Définition.- Un Graphe Orienté Pondéré est : (X , Lab , Succ) / (Rac , Clas) où

- X est l'ensemble des sommets ou noeuds du graphe
- Lab est la fonction Label, associant à chaque sommet son étiquette, fonction ou variable
- Succ est la fonction Successeur de XxN dans XxZ
  Succ (s,k) = (s_k,p_k) : s_k est le $k^{ième}$ successeur de s et la pondération de cet arc est p_k
- Rac est une racine pondérée : Rac = (s_0,p_0) , sommet et pondération tête du Gop
- Clas, la congruence d'indices, associe à une variable sa période, entier naturel .

Seules les pondérations (sur les arcs et la racine) et la congruence d'indices ont été ajoutées aux Graphes Orientés classiques .

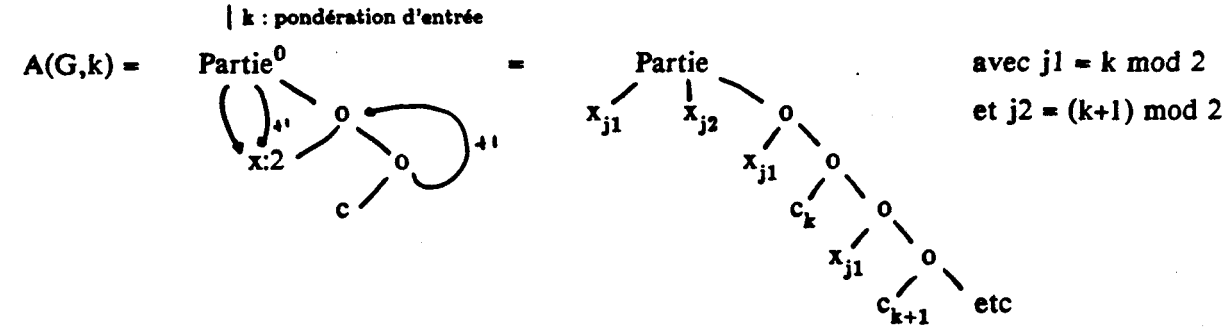
Exemple :



Cinq sommets, dont le sommet racine *Partie* de pondération tête nulle , . . .

Interprétation d'un Gop G pour une pondération d'entrée k : A (G , k) .

Nous considérons une pondération d'entrée pour toute interprétation d'un Gop : ce sera la valeur de départ pour le cumul des pondérations. Lors du dépliage, on effectue une sommation de toutes les pondérations rencontrées, c-à-d la pondération d'entrée, la pondération de la racine et les pondérations des arcs de la branche . Cette somme modulo la congruence de la variable x sert d'indice à x. Par exemple :



Dans notre partie d'Echecs infinie, A(G,k) représente l'état au $k^{ième}$ tour :

. le premier sous-arbre de A(G,k) désigne le joueur qui a la main , $x_{k \bmod 2}$

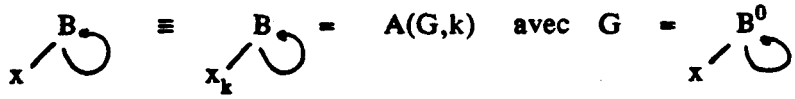
. le deuxième sous-arbre désigne l'autre joueur , $x_{k+1 \bmod 2}$

. le troisième, ce qui reste à jouer de cette partie infinie après les $k-1^{iers}$ coups,

c'est au tour de $x_{k \bmod 2}$ de jouer le coup c_k .

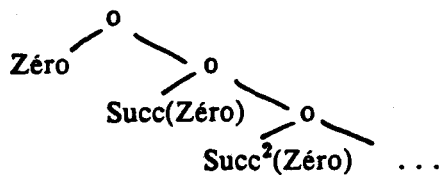
Les Gops généralisent la notion d'arbres infinis rationnels,

a - puisque pour tout arbre rationnel, on peut construire un Gop d'interprété équivalent :



il suffit d'ajouter des pondérations nulles partout, sans aucune congruence .

b- mais ils caractérisent des arbres irrationnels particuliers : un Gop sans variable a toujours un interprété rationnel et donc aucun terme irrationnel clos ne peut être représenté sous forme d'un Gop :



Définition.- L'interprété d'un Gop G , noté I(G) est la fonction qui associe à tout entier relatif k, l'interprété de G pour cette pondération d'entrée k :

$$I(G) = \{ (k , A(G , k)) \ / \ \forall k \in Z \}$$

Chap 2 (suite) - Unification de Graphes Orientés Pondérés

L'unification sur les arbres finis, les graphes sans cycle (directed acyclic graphs : dags), les graphes orientés (gos) est bien connue. Deux termes sont dits unifiables ssi il existe une instanciation commune les rendant égaux. L'unifié correspond à la plus petite de ces instanciations :

$t_1 \vee t_2$ est le plus petit terme vérifiant $\sigma(t_1) = \sigma(t_2)$.

Pour une substitution σ, on notera σ (I(G)) l'instanciation de chacun des arbres de I(G) :

$$\sigma (I(G)) = \{ [k , \sigma(A(G,k))] \ / \ \forall \ k \in Z \}.$$

Définition.- Deux Gops G_1 et G_2 sont dits *unifiables* si il existe une instanciation commune σ , éventuellement infinie, rendant les interprétés $I(G_1)$ et $I(G_2)$ identiques .

Propriété.- L'unification est une opération interne sur les Graphes Orientés Pondérés.
Si G_1 et G_2 sont unifiables, il existe un Gop, noté $G_1 \vee G_2$ tel que

$$I(G_1 \vee G_2) = I(G_1) \vee I(G_2)$$

Notre algorithme d'unification est une généralisation de celui sur les Graphes Orientés classiques. Nous avons choisi de nous inspirer de celui proposé par Fages .

Algorithme de Fages.- Soient deux Gos (X,Lab,Succ)/s et (X,Lab,Succ)/s' de racines s et s' à unifier.
Propriété.- On ne change rien à l'unification si on remplace dans la fonction Succ le sommet s' par le sommet s :
$\forall$ s" $\in$ X , Si Succ(s",i) = s' Alors Succ'(s",i) = s Sinon Succ'(s",i) = Succ(s",i) (inchangé)
Principe de l'algorithme de Fages. A chaque étape, quatre cas peuvent se présenter :

1- les sommets s et s' sont les mêmes, le résultat est immédiat : c'est le même Go

2- les sommets sont différents, mais l'un est étiqueté d'une variable x, par exemple s'.
On remplace dans la fonction Succ le sommet s' par le sommet s.
Après cette modification, le Go unifié est (X,Lab,Succ')/s .

ex : 

3- les sommets s et s' sont différents, mais étiquetés du même symbole de fonction.
On remplace le sommet s' dans la fonction Successeur par le sommet s ;
pour chaque fils s_i et s'_i de s et s', on unifie les Gos de racine s_i et s'_i

4- les sommets s et s' sont étiquetés de symboles de fonction différents : l'unifié n'existe pas.

Propriété de base de notre algorithme.- Soient deux Gops de racines (s,p) et (s',p') à unifier, on ne change rien à l'unification si on remplace dans la fonction Successeur commune le sommet s' par le sommet s en incrémentant la pondération des arcs redirigés de p-p' :

$\forall$ s" $\in$ X , Si Succ(s",i) = (s',p_i) Alors Succ'(s",i) = (s,p_i+p-p') Sinon Succ'(s",i) = Succ(s",i)

Exemple 1 - x^1 V B^0 (y, x)

Pour la pondération k d'entrée, les Gops sont interprétés comme x_{k+1} et $B(y_k, x_k)$.
On recherche donc la plus petite instanciation σ qui, pour toute pondération d'entrée , rend ces interprétés égaux : $\forall k \in Z$, $\sigma(x_{k+1}) = \sigma(B(y_k, x_k))$.
En itérant cette équation sur elle-même :

$$\sigma(x_{k+1}) = B(y_k, \sigma(x_k)) = B(y_k, B(y_{k-1}, \sigma(x_{k-1}))) = \ldots$$

on construit la solution :

$\sigma(x_{k+1})$ = B (y_k, B (y_{k-1}, B (y_{k-n}, ...)))

=

| k : pondération d'entrée

B^0 (y, -1)

L'unifié de ces deux Gops est donc : B^0 (y, -1)

En utilisant la Propriété : x^1 V B^0 (y, x) = x^1 V B^0 (y, -1)

l'arc allant vers le sommet x est redirigé vers le sommet B avec incrémentation de (-1) de la pondération de l'arc redirigé.

L'unification se ramène donc à : x^1 V B^0 (y, -1) = (évident) B^0 (y, -1)

Exemple 2 - $Frère^0$ (x, y) V $Frère^{-1}$ (y, x)

Pour la pondération d'entrée k, les Gops sont interprétés : Frère (x_k , y_k) et Frère (y_{k-1} , x_{k-1})
La plus petite substitution les rendant égaux quel que soit k repose sur les équations suivantes :

$$\forall k \in Z \ , \ x_k = y_{k-1} \text{ et } y_k = x_{k-1} \iff \forall k \in Z \ , \ x_k = x_{k \bmod 2} \text{ et } y_k = x_{k+1}$$

L'unifié est : $Frère^0$ (x :2, +1)

Appliquons la propriété sur chacun des sous-Gops fils

a- Unification de x^0 et y^{-1}. On remplace le sommet y par le sommet x avec incrémentation de 1 des pondérations :

b- Unification de x^{+1} et x^{-1}. Ces deux Gops sont égaux ssi la variable x est de période 2 :

L'unification est terminée : les deux Gops ont le même interprété .

Principe de l'algorithme.- Soient les Gops de racines (s,p) et (s',p') à unifier , à chaque étape de l'unification, quatre cas peuvent se présenter :

1- Les sommets s et s' sont les mêmes s = s'

1-a Si les pondérations sont identiques : c'est le même Gop .

1-b Si les pondérations sont différentes, les Gops ne diffèrent que par leurs pondérations tête : le Gop est de période |p-p'| , c-à-d toutes ses variables sont de périodes |p-p'| (Voir Exemple 2)

2- Les sommets sont différents, mais l'un est étiqueté d'une variable x , par exemple s'.

. si x a une période , alors cette période est vérifiée aussi sur toutes les variables de G_s .

. on remplace le sommet s' par le sommet s avec incrémentation des pondérations redirigées de (p-p').

Après ces modifications, le Gop unifié est G_s (Voir Exemple 1)

3- Les sommets s et s' sont différents et étiquetés du même symbole de fonction.

. on remplace le sommet s' par le sommet s avec incrémentation des pondérations redirigées de (p-p')

. pour chaque fils s_i et s'_i de s et s', on unifie les Gops de racines (s_i,p_i+p) et (s'_i,p'_i+p')

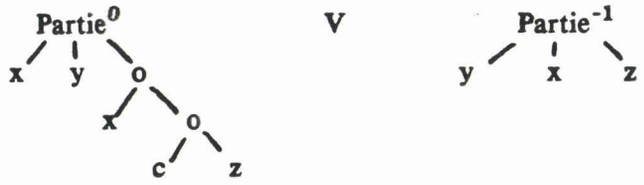
4- Les sommets s et s' sont étiquetés de symboles de fonction différents : L'unifié n'existe pas .

Algorithme d'Unification.- Soient deux Gops G et G' de racines (s,p) et (s',p'), cet algorithme modifie les Gops G et G' jusqu'à ce qu'ils soient égaux, ou échoue si l'unifié n'existe pas.

Procédure UNIFICATION ((s,p) , (s',p'))
Début
  Si (s = s')
    Alors % Toutes les variables du Gop G_s sont de période |p-p'| %
        ∀ x , variable de G_s , Clas(x) = PGCD (Clas(x) , |p-p'|)
    Sinon
      Si (s' est étiqueté d'une variable x')
        Alors % Toutes les variables du Gop G_s sont de période Clas(x') %
            ∀ y , variable de G_s , Clas(y) = PGCD (Clas(y) , Clas(x'))
          % Tout arc allant sur le sommet s' est redirigé vers le sommet s
          et la pondération de ces arcs est incrémentée de (p - p') %
            ∀ s" ∈ X , si Succ(s",i) = (s',p_i) alors Succ(s",i) = (s,p_i+p-p')
        Sinon
          Si (s est étiqueté d'une variable x) % Cas symétrique %
            Alors % Toutes les variables du Gop $G_{s'}$ sont de période Clas(x) %
                ∀ y , variable de $G_{s'}$, Clas(y) = PGCD (Clas(y) , Clas(x))
              % Tout arc allant sur le sommet s est redirigé vers le sommet s'
              et la pondération de ces arcs est incrémentée de (p' - p) %
                ∀ s" ∈ X , si Succ(s",i) = (s,p_i) alors Succ(s",i) = (s',p_i+p'-p)
            Sinon
              Si (s et s' sont étiquetés du même symbole de fonction)
                Alors % Tout arc allant sur le sommet s' est redirigé vers le sommet s
                    et la pondération de cet arc est incrémentée de (p - p') %
                    ∀ s" ∈ X , si Succ(s",i) = (s',p_i) alors Succ(s",i) = (s,p_i+p-p')
                  Pour i := 1 A Nombre de fils de s et s' Faire
                    % Posons Succ(s,i) = (s_i , p_i) et Succ(s',i) = (s'_i , p'_i) %
                    UNIFICATION ((s_i , p+p_i) , (s'_i , p'+p'_i)) ;
                  Fait
                Sinon Echec -- L'unifié n'existe pas.
Fin

Remarque.- L'existence de l'unifié de 2 Gops ne dépend pas des pondérations et congruences.

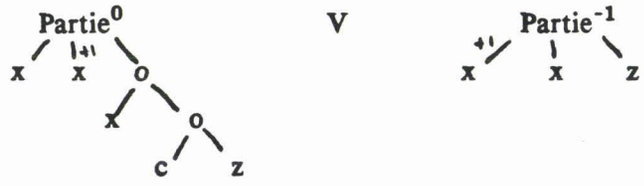
Exemple 3 - Reprenons notre Partie d'Echecs et calculons l'unifié des Gops :



Etape 1- On remplace le sommet Partie de pondération tête (-1) par celui de pondération tête nulle, mais étant inaccessible par d'autres sommets, la fonction Successeur n'est pas modifiée. Leurs étiquettes sont identiques. On unifie leurs sous-Gops fils.

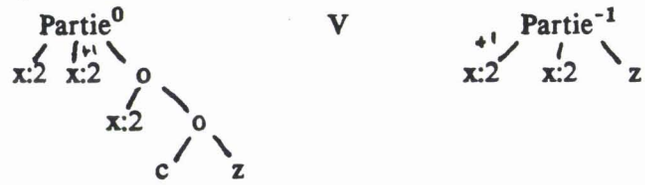
Etape 2 - Unification des Gops "premiers fils" : x^0 V y^{-1} .

Le sommet y est substitué dans la fonction Successeur par le sommet x avec une correction +1 des pondérations. Les Gops à unifier sont maintenant :

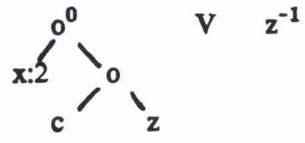


Etape 3 - Unification des Gops "seconds fils" : x^1 V x^{-1} .

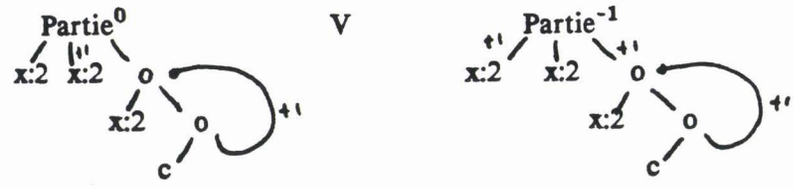
Ces deux sous-Gops sont égaux ssi la variable x est de période 2. On est ramené à l'unification :



Etape 4 - Unification des Gops fils 3 : o^0 V z^{-1}



On remplace le sommet z par le sommet o avec correction +1 de la pondération. Reste à résoudre :



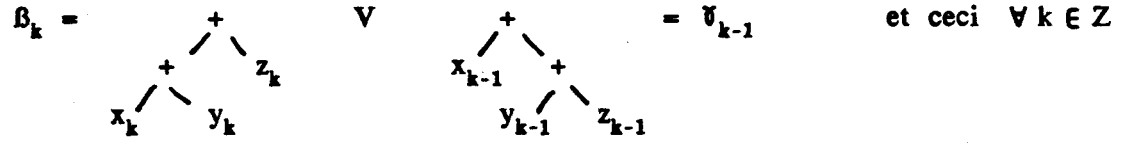
L'unification est terminée : les deux Gops sont égaux (i.e. même interprété) .

Chap 3 - Graphe Orienté Pondéré d'une règle récursive.

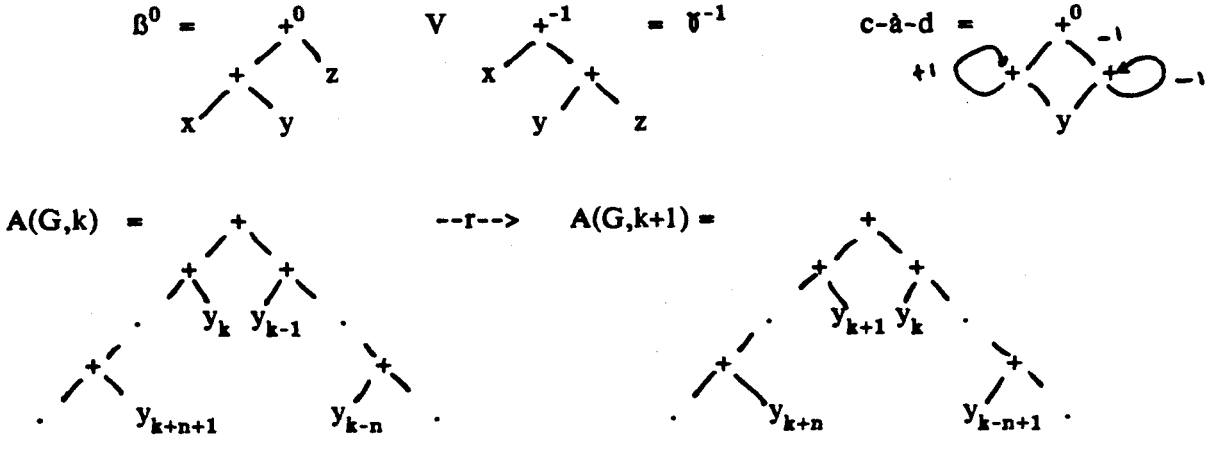
Propriété.- Le Gop $ß^0$ V $ϑ^{-1}$ caractérise l'application "bi-infinie" de la règle ß -> ϑ .
Les Gops $ß^0$ et $ϑ^{-1}$ sont construits sur les arbres ß et ϑ en ajoutant des pondérations nulles sur tous leurs arcs. Notons G l'unifié de $ß^0$ et $ϑ^{-1}$, les arbres A(G,k) sont les plus petits arbres se réécrivant en un arbre équivalent, ils formalisent la séquence "bi-infinie" de réécritures la plus générale:

. . -r-> A(G,-k) -r-> A(G,-k+1) -r-> . . -r-> A(G,0) -r-> . . -r-> A(G,k) -r-> A(G,k+1) -r-> . .

Principe de la preuve.- Nous avons vu que l'application infinie de la règle est caractérisée par la substitution (éventuellement infinie) vérifiant : $\forall k \in Z$, $\sigma(ß_k) = \sigma(ϑ_{k-1})$
Sur l'exemple de l'associativité de l'opération " + " :



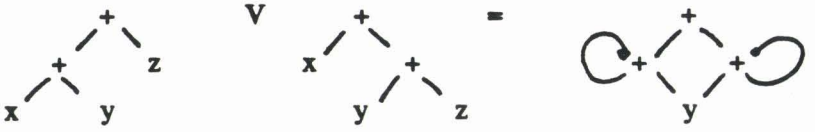
En terme de Graphe Orienté Pondéré, cette unification de deux ensembles infinis d'arbres ($ß_i$) et ($ϑ_{i-1}$) est exactement analogue à celle des Gops $ß^0$ et $ϑ^{-1}$:



Convention.- Le Gop $ß^0$ V $ϑ^{-1}$ est appelé le *Gop caractéristique* de la règle ß -> ϑ .

Propriété.- β^0 V ϑ^{-1} existe ssi β V ϑ existe sans test d'occurrence.
β^0 V ϑ^{-1} et β V ϑ sont identiques aux pondérations et aux périodes près.

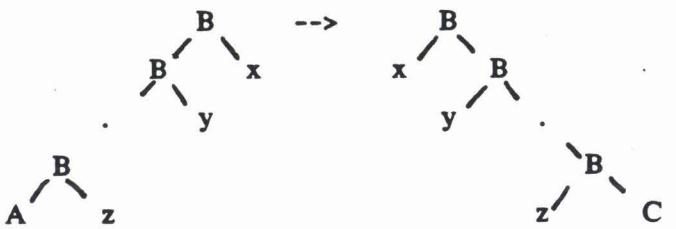
Exemple :



Chap 3 (suite) - Terminaison d'une règle de réécriture en tête sans test d'occurrence

Théorème 1.- La propriété de terminaison est décidable pour une règle de réécriture en tête sans test d'occurrence. Si l'on considère (comme en Prolog) des réécritures en tête sans test d'occurrence, la règle β -> ϑ satisfait pour tout arbre (quelconque, fini ou infini) la propriété de terminaison si et seulement si β et ϑ ne sont pas unifiables sans test d'occurrence .

La règle suivante satisfait la propriété de terminaison, car les deux membres de la règle ne sont pas unifiables :



si elle contient n variables, seules au plus 2^n réécritures sont possibles ici.

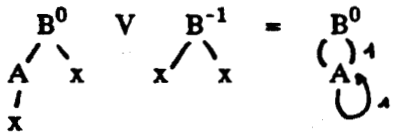
Corollaire.- Un programme Prolog composé d'une seule règle satisfait, pour toute question, la propriété de terminaison, dans le cadre d'une stratégie linéaire classique en chaînage arrière, si et seulement si le premier terme du corps de la règle n'est pas unifiable avec la tête de règle.
En effet, l'arrêt du programme Prolog composé de la seule règle " β -> ϑ δ . . μ " est équivalent à celui de β -> ϑ .

Corollaire.- Un programme Prolog composé d'une seule règle satisfait, pour toute question T, la propriété de terminaison, dans le cadre d'une stratégie équitable pour le choix du terme à réécrire (*fair-strategy*), si et seulement si l'un des termes du corps de la règle n'est pas unifiable avec la tête de règle.

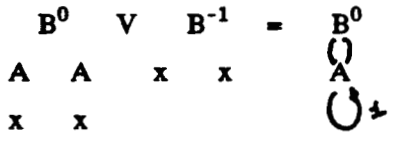
Chap 4 - Interprétation finie des Graphes Orientés Pondérés .

Pour étudier le comportement précis de notre règle récursive, il est fondamental de pouvoir formaliser son évolution à chaque nouveau pas de réécriture, et non plus connaître le résultat *à l'infini*. Par exemple, les deux règles suivantes ont des Gops caractéristiques équivalents :

r_1 : B (A(x),x) -> B(x,x) ;



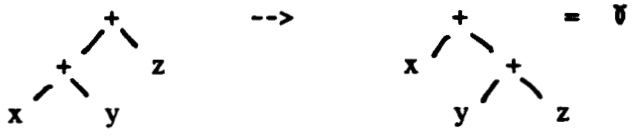
r_2 : B (A(x),A(x)) -> B(x,x) ;



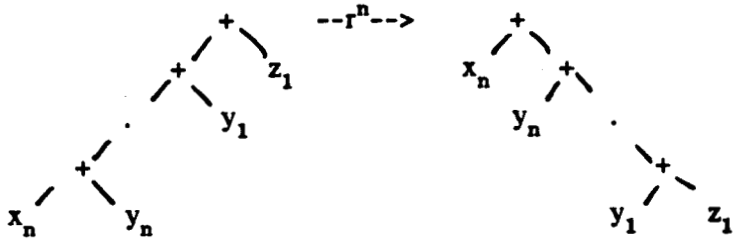
L'interprété dans les deux cas est l'arbre B (A^{∞} , A^{∞}) , pourtant la première règle ne satisfait plus le test d'occurrence dès la $2^{ème}$ réécriture et donc vérifie la propriété de terminaison si l'on applique le test d'occurrence , contrairement à la seconde règle . Seule une étude fine et finie des appels récursifs pourra répondre à ce type de questions. C'est pour cette raison que nous proposons une nouvelle interprétation finie des Graphes Orientés Pondérés.

Pour en montrer le principe, reprenons l'exemple qui nous a servi à introduire les Gops : l'associativité de l'opération " + " :

r : ß =

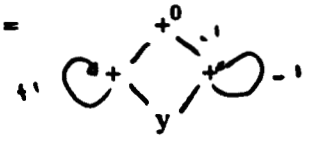


Appliquer n fois cette règle, revient à utiliser la règle :



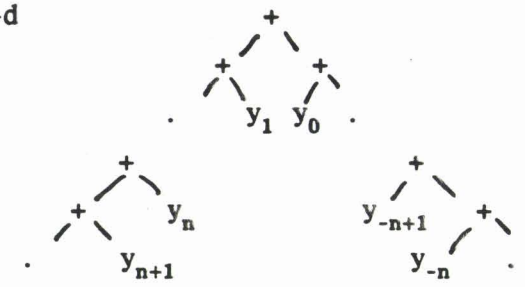
Le Gop caractéristique de cette règle est :

G =

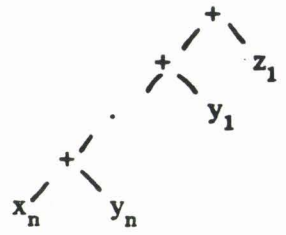


Comparons l'interprété A(G,1) et l'arbre gauche de notre règle itérée n fois :

c-à-d



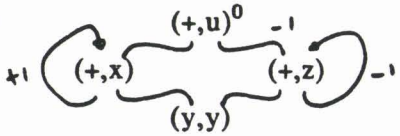
et



on remarque que A(G,1) est une instanciation de l'arbre gauche de r^n, c-à-d que le dépliage du Gop est "trop profond", en particulier, toutes les variables indicées de A(G,1) n'appartenant pas à l'intervalle [1,n] sont *de trop*. Or les indices des variables sont les sommes des pondérations le long de la branche. Pour interpréter finiment notre Gop, il faudrait "arrêter" le dépliage du Gop dès que ces indices "dépassent certaines bornes".

Plus précisément, on interprète un Gop *sur un intervalle de pondérations valides*, en dépliant ses branches tant que la pondération sommée appartient à cet intervalle. Pour gérer ce point de rupture (indice hors de l'intervalle), on associe aux sommets du graphe une seconde étiquette, symbole de variable unique.

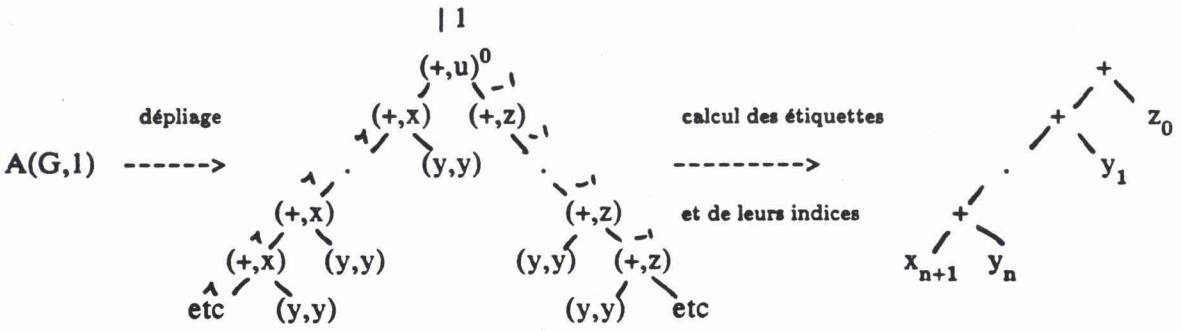
G =



u, x, y, z sont des variables "second Label " caractérisant leur sommet.

Lors du dépliage de ce graphe, un sommet est interprété comme son premier Label si la somme des pondérations est valide, ou est interprété comme son second Label (variable) indicée de cette somme si elle n'est pas valide.

Choisissons l'intervalle [1,n] comme intervalle des pondérations valides, et 1 comme pondération d'entrée :



Cet interprété est équivalent à l'arbre gauche de notre règle r^n ; seules les variables x_n et z_1 ont été remplacées par x_{n+1} et z_0 .

Autre exemple.- Soit la règle r : Succ (x) -> x [x+1 ∈ N SI x ∈ N]
Il est facile de voir qu'une séquence de n réécritures est une instanciation de :

$Succ^n(x)$ -r-> $Succ^{n-1}(x)$ -r-> .. .. -r-> Succ (x) -r-> x

Le Gop caractéristique est ici : $(Succ,u)^0$ V (x,x) = $^0(Succ,u)_{+1}$ avec (x,x)

Si l'on interprète ce Gop sur l'intervalle valide [1,n] , on obtient exactement la séquence la plus générale de n réécritures :

$A(G,1) = Succ^n(u_{n+1})$ -r-> $A(G,2) = Succ^{n-1}(u_{n+1})$ -r-> . . $A(G,n) = Succ(u_{n+1})$ -r-> $A(G,n+1) = u_{n+1}$

Chap 4 (suite) - Définition des IGOPS

Cette partie est beaucoup plus technique, et les résultats finals pour la résolution avec test d'occurrence montreront que le Gop suffit à caractériser l'application finie d'une règle de réécriture récursive.

Définition.- Un IGOP est syntaxiquement : (X , Lab , Succ, Clas) / Rac où

- X est l'ensemble des sommets ou noeuds du graphe
- Lab est la fonction Label associant à chaque sommet deux étiquettes, la première symbole de fonction ou variable, la seconde, symbole de variable, qui caractérise ce sommet
- Succ est la fonction Successeur de XxN dans XxZ

  Succ (s,k) = (s_k,p_k) : s_k est le $k^{ième}$ successeur de s et la pondération de cet arc est p_k
- Rac est une racine pondérée : Rac = (s_0,p_0) , sommet et pondération tête du Gop
- Clas, la congruence d'indice, associe à un sommet sa période, entier naturel .

Par rapport aux GOPS, les fonctions Lab et Clas ont été modifiées : tout sommet du graphe possède ici une congruence d'indices et un second Label , variable qui le caractérise.

Exemple : $(Partie,u)^0$ avec $(x,x):2$, (o,z), (o,v), (c,c)

Les cinq sommets ont une seconde étiquette un symbole de variable qui leur est propre.

Définition.- Un Igop est dit *homogène* si la fonction Congruence d'indices est cohérente avec la fonction Successeur, c-à-d si tout sommet est de congruence multiple de celle de n'importe lequel de ses fils : $\forall$ s , si Succ(s,i) = (s_i , p_i) alors Clas(s) = k * Clas(s_i)

Définition.- Un Igop est dit *parfait* si tous ses sommets étiquetés d'un symbole de fonction sont de congruence nulle .

Interprétation d'un Igop G comme système d'équations : $S_P(G)$.

Nous découpons ce graphe en *tranches*, c-à-d en arbres de hauteur 1. Pour un intervalle P de pondérations valides, le système $S_P(G)$ sera composé de deux types d'équations :

1- équations (*variable indicée = arbre*)

Soit un sommet s de labels (f,v) où f est un symbole de fonction d'arité n, et v sa variable ,

notons - $\forall$ i $\in$ [1,n] , Succ(s,i) = (s_i , p_i)

- x, y, .. ,z les variables ($2^{ème}$ étiquette) des sommets fils de s : s_1, ... , s_n

alors $\forall k \in P$, $[v_k = f (x_{k+p1} , y_{k+p2} , \ldots , z_{k+pn})] \in S_P (G)$

2- équations congruentes ($x_k = x_{k \bmod .}$)

pour tout sommet s de G de congruence non nulle, notons v sa variable ($2^{ème}$ étiquette)

$\forall k \in P$, $(v_k = v_{k \bmod Clas(s)}) \in S_P (G)$ - [k mod Clas(s)] est choisi dans P -

Exemple : Soit G le Igop Partie (page précédente), $S_{[1,n]} (G)$ contient les équations suivantes :

$\forall k \in [1,n]$ - u_k = Partie (x_k , x_{k+1} , z_k) , $z_k = o (x_k , v_k)$, $v_k = o (c_k , z_{k+1})$

- $x_k = x_2$ si k pair , $x_k = x_1$ si k impair

Test d'occurrence d'un Igop : Restriction sur les Pondérations de Boucles (R.P.B.).-

Soit G un Igop homogène, quel que soit P, intervalle d'interprétation , S_P (G) satisfait le test d'occurrence ssi il n'existe pas de boucles de pondération nulle (modulo les congruences des sommets traversés) dans le Igop. Cette condition est vérifiée ssi :

1- tout sommet bouclant de G est de congruence nulle

2- toute boucle élémentaire est de pondération non nulle

3- les pondérations des boucles élémentaires d'un même sommet sont toutes de même signe.

(Par sommet bouclant, on désigne tout sommet accessible à partir de lui-même par la fonction Successeur, une boucle élémentaire du sommet s étant un chemin allant de s à lui-même en ne traversant que des sommets différents).

Le test d'occurrence sur l'unification des arbres finis s'exprime comme la non-existence de boucles dans le Graphe Orienté unifié. Ici, la condition sera donc la non-existence de boucles de pondération nulle (modulo la congruence du sommet bouclant).

Interprétation d'un Igop pour une pondération d'entrée k : A_p (G,k) .

Soient (s,p) la racine de G et v la variable associée à s, posons $\sigma_p(G)$ la substitution correspondant au système d'équations S_p (G), l'interprété de G pour la pondération k est σ (u_{k+p}) .

Définition.- L'interprété d'un Igop sur l'intervalle E de pondérations d'entrée et l'intervalle P de pondérations valides est la fonction qui associe à chaque pondération d'entrée k de E son interprété sur les pondérations valides de P : $I_{E,P}(G) = \{ (k , A_p(G)) \ / \ \forall k \in E \}$

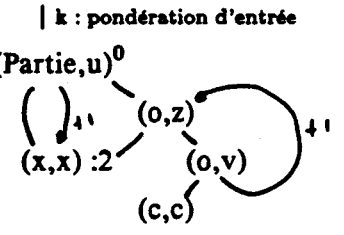
Ces définitions sont un peu obscures, mais elles peuvent être nettement simplifiées si les Igops sont supposés parfaits, car ceux-ci ont la particularité d'avoir, dans leur interprétation $S_p(G)$, une seule équation par variable indicée.

Interprétation d'un Igop parfait G pour une pondération d'entrée k : A_p (G,k)

Comme pour les Gops, les pondérations d'une branche sont sommées : pondération d'entrée, pondération tête et pondérations des arcs; mais ici on "coupe" la branche dès que cette somme n'appartient plus à l'intervalle de pondérations valides P : soit Σ cette somme, si Σ est *valide* (i.e. $\in$ P), rien n'est changé, Σ modulo la congruence de la variable x sert d'indice à x ; par contre si Σ n'est pas valide, cette somme, telle qu'elle, indice la variable du sommet.

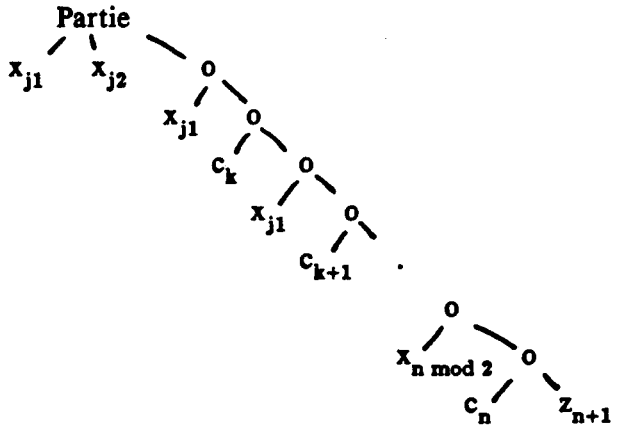
Reprenons notre exemple de Partie d'Echecs :

$A_{[1,n]}(G,k)$ = | k : pondération d'entrée



. si k $\in$ [1,n-1], posons j1 = k mod 2 et j2 = (k+1) mod 2

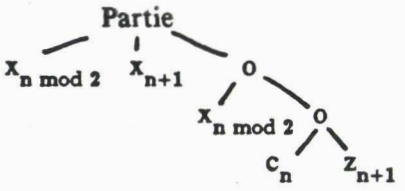
$A_{[1,n]}(G,k)$ =



. si $k \notin [1,n]$, $A_{[1,n]}(G,k) = u_k$

. L'interprétation pour la pondération n est ici un cas particulier :

$A_{[1,n]}(G,n) =$



Dans une partie d'Echecs finie composée d'au moins n coups, les arbres A(G,k) pour k appartenant à [1,n-1] représentent l'état de la partie au $k^{ième}$ tour :

. le premier sous-arbre de A(G,k) désigne le joueur qui a la main , $x_{k \bmod 2}$

. le deuxième sous-arbre désigne l'autre joueur , $x_{k+1 \bmod 2}$

. le troisième, ce qui reste à jouer de cette partie finie après les $k-1^{iers}$ coups,

c'est au tour de $x_{k \bmod 2}$ de jouer le coup c_k .

On peut observer sur cet exemple que l'état de la partie au $n^{ième}$ coup n'est pas correctement caractérisé par l'interprété $A_{[1,n]}(G,n)$, ce sont des effets parasites sur les bords que nous retrouvons dans l'unification des Igops.

Chap 4 (suite) - Unification des IGOPS

Définition.- Deux Gops G_1 et G_2 sont dits unifiables pour un intervalle E de pondérations d'entrée et un intervalle de pondérations valides P si il existe une instanciation commune σ , éventuellement infinie, rendant les interprétés $I_{E,P}(G_1)$ et $I_{E,P}(G_2)$ identiques .

Remarque.- L'unification des Igops n'est pas une opération interne. En général il n'existe pas de Igop $G_1 \vee G_2$ caractérisant l'unifié de G_1 et G_2 sur les intervalles E et P.

Pour simplifier cette présentation, nous supposerons que les Igops G et G' à unifier sont interprétés sur un intervalle unique, intervalle de pondérations d'entrée et de pondérations valides (E = P), et qu'ils ne diffèrent que par leurs racines.

Soient $G = (X,Lab,Succ,Clas)/Rac$ et $G' = (X,Lab,Succ,Clas)/Rac'$ à unifier sur l'intervalle P :

$$I_P(G) \vee I_P(G') .$$

Propriété.- Pour tout intervalle P d'interprétation assez grand, si l'unifié des interprétés de G et G' existe, alors il existe un Igop noté $G \vee G'$ qui encadre l'unifié par son interprétation sur deux intervalles P_m et P_M dits de minoration et de majoration :

$$I_{Pm}(G_1 \vee G_2) \leq I_P(G_1) \vee I_P(G_2) \leq I_{PM}(G_1 \vee G_2)$$

où P_m et P_M sont presque égaux à P, c-à-d que les bornes de ces intervalles sont égales à celles de P à une constante près.

L'algorithme de construction de G V G'. Son principe est similaire à celui d'unification des Gops, seules les congruences sont maintenant traitées sur tous les sommets et non plus uniquement sur les sommets de symbole variable.

Procédure ENCADREMENT ((s,p) , (s',p'))
Début
  Si (s = s')
    Alors % Tous les sommets du Gop G_s sont de période |p-p'| %
      ∀ s" , sommet de G_s , Clas(s") = PGCD (Clas(s") , |p-p'|)
    Sinon
      % Tous les sommets des Igops G_s et $G_{s'}$ sont de période PGCD (Clas(s),Clas(s')) %
        ∀ s" , sommet de G_s ou $G_{s'}$, Clas(s") = PGCD (Clas(s") , Clas(s), Clas(s'))
      Si (s' est étiqueté d'une variable)
        Alors % Tout arc allant sur le sommet s' est redirigé vers le sommet s
          et la pondération de ces arcs est incrémentée de (p - p') %
            ∀ s" ∈ X , si Succ(s",i) = (s',p_i) alors Succ(s",i) = (s,p_i+p-p')
        Sinon
          Si (s est étiqueté d'une variable) % Cas symétrique %
            Alors % Tout arc allant sur le sommet s est redirigé vers le sommet s'
              et la pondération de ces arcs est incrémentée de (p' - p) %
                ∀ s" ∈ X , si Succ(s",i) = (s,p_i) alors Succ(s",i) = (s',p_i+p'-p)
            Sinon
              Si (s et s' sont étiquetés du même symbole de fonction)
                Alors % Tout arc allant sur le sommet s' est redirigé vers le sommet s
                  et la pondération de cet arc est incrémentée de (p - p') %
                    ∀ s" ∈ X , si Succ(s",i) = (s',p_i) alors Succ(s",i) = (s,p_i+p-p')
                  Pour (i := 1 A Nombre de fils de s et s') Faire
                    % Posons Succ(s,i) = (s_i , p_i) et Succ(s',i) = (s'_i , p'_i) %
                    ENCADREMENT ((s_i , p+p_i) , (s'_i , p'+p'_i)) ;
                  Fait
                Sinon Echec -- L'unifié n'existe pas.
Fin

Remarques.-

1- Par rapport à l'algorithme des Gops, seul le calcul des congruences a été modifié.

2- Si les Igops G et G' sont homogènes, alors G V G' l'est aussi.

3- Si le Igop G V G' satisfait la Restriction sur les Pondérations de Boucles, on peut annuler les congruences sur les sommets de symbole de fonction sans altérer la propriété d'encadrement. Le Igop G V G' devient parfait.

Gops et Igops.- Cette dernière remarque permet de simplifier l'utilisation de nos résultats, puisque le Igop Parfait G V G' et le Gop unifié (au sens de l'interprétation infinie) ne diffèrent que par la présence d'un second Label sur chaque sommet. En d'autres termes, si la R.P.B. est vérifiée, le Gop unifié est pleinement suffisant pour exprimer ces propriétés d'encadrement.

Chap 5 - Terminaison d'une règle de réécriture en tête avec test d'occurrence

Caractérisation.- Appliquer n réécritures d'une règle est formalisable par l'unification des Igops β^0 et γ^{-1} interprétés sur [1,n] et [2,n+1] :

$$I_{[1,n],[1,n]}(\beta^0) \ V \ I_{[2,n+1],[1,n]}(\gamma^{-1}) = \{ (k , t_k) / k \in [1,n+1] \}$$

Ces couples (k , t_k) expriment la plus petite séquence de n réécritures :

$$t_1 \text{ --r--> } t_2 \text{ --r--> } \ldots \text{ --r--> } t_n \text{ --r--> } t_{n+1}$$

En effet, l'application de n réécritures par la règle $\beta \rightarrow \gamma$ est caractérisée par :

$\exists \sigma_n$ la plus petite substitution telle que $\forall \ k \in [2,n] \ , \ \sigma_n(\beta_i) = \sigma_n(\gamma_{i-1})$,

ce qui est exprimable par l'unification de $I_{[1,n],[1,n]}(\beta^0)$ et $I_{[2,n+1],[1,n]}(\gamma^{-1})$.

Théorème 2.- Une règle $\beta \rightarrow \gamma$ satisfait la propriété de terminaison pour tout arbre T (quelconque, fini ou infini) si et seulement si :

. soit β V γ n'existe pas

. soit β^0 V γ^{-1} n'a pas de boucles de pondération nulle (modulo la période du sommet bouclant).

Corollaire.- Un programme Prolog composé d'une seule règle satisfait la propriété de terminaison dans le cadre d'une stratégie linéaire classique en chaînage arrière avec test d'occurrence, si et seulement si la tête ß de la règle et le premier terme du corps ϑ de la règle vérifient :

. soit ß V ϑ n'existe pas

. soit $ß^0$ V $ϑ^{-1}$ ne satisfait pas la Restriction sur les Pondérations de Boucles.

Corollaire.- Un programme Prolog composé d'une seule règle satisfait la propriété de terminaison dans le cadre d'une stratégie équitable avec test d'occurrence, si et seulement si la tête ß de la règle et l'un des termes ϑ du corps de la règle vérifient :

. soit ß V ϑ n'existe pas

. soit $ß^0$ V $ϑ^{-1}$ ne satisfait pas la Restriction sur les Pondérations de Boucles.

Chap 5 (suite) - Caractérisation Itérative de la Récursivité.

Plaçons-nous maintenant dans le cas où notre règle récursive ß -> ϑ ne satisfait pas la propriété de terminaison, pour certains termes T à réécrire avec test d'occurrence.

Nous savons que la plus petite substitution $σ_n$ vérifiant : ∀ k ∈ [2,n] , $σ_n(ß_i) = σ_n(ϑ_{i-1})$ permet de formaliser n applications de la règle ß -> ϑ : $σ_n(ß_1)$ --r^n--> $σ_n(ϑ_n)$.

Grâce à la propriété d'encadrement de l'unifié de deux Igops, cette substitution peut être définie itérativement et indépendamment du nombre de réécritures n, sous la forme de l'union de trois systèmes d'équations (*une équation par variable indicée*) :

Sg U Système itératif U $Sd_{/n}$

où . Sg est le système constant des effets de bord à gauche

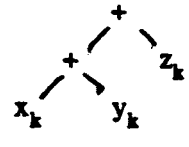
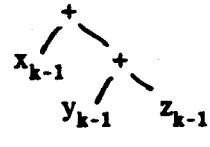
. $Sd_{/n}$ est le système constant "par rapport à n" des effets de bord à droite

. Système itératif est le système $S_{[a,n-b]}$ ($ß^0$ V $ϑ^{-1}$) à quelques équations près

et a, b sont des constantes indépendantes de n.

Exemple.- Reprenons le problème de l'associativité de l'opération " + " :

Les n applications de cette règle sont caractérisées par l'unification des arbres $ß_i$ et $ϑ_{i-1}$:

$ß_k$ =  V  = $ϑ_{k-1}$ et ceci ∀ k ∈ [2,n]

Les contraintes relatives à n applications de cette règle sont donc :

. $z_i = +(y_{i-1}, z_{i-1})$ $\forall i \in [2,n]$

. $x_i = +(x_{i+1}, y_i)$ $\forall i \in [1,n-1]$

On peut donc décomposer ces équations en trois parties :

. Système itératif sur [2,n-1]

$z_i = +(y_{i-1}, z_{i-1})$ et $x_i = +(x_{i+1}, y_i)$

. Système Sg (effet de bord gauche)

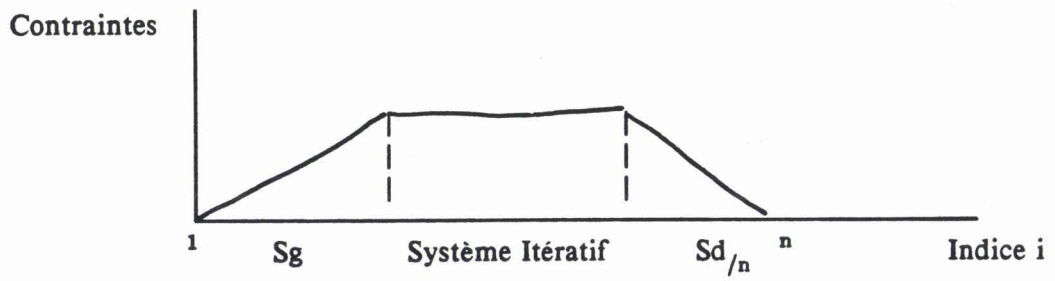
$x_1 = +(x_2, y_1)$

. Système $Sd_{/n}$ (effet de bord droit)

$z_n = +(y_{n-1}, z_{n-1})$

Le diagramme suivant représente l'évolution des contraintes pour un symbole de variable quelconque x de ß ou ℧. Lors des n réécritures, les contraintes sur les variables indicées x_i croissent pendant une phase constante d'accélération (effet de bord gauche), atteignent leur phase de croisière durant laquelle elles sont stables, puis décroissent lors d'une phase de décélération (effet de bord droit) :

Schématisation de la réécriture.- Contraintes sur une variable x_i :



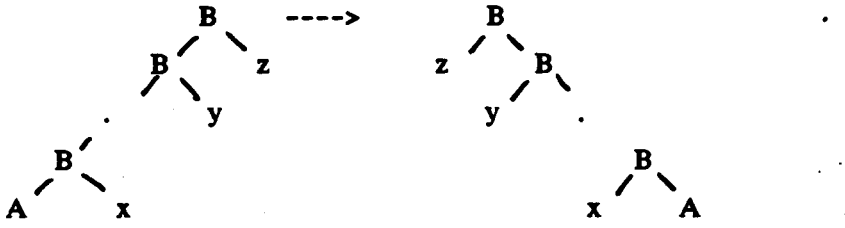
Pour n ≥ taille des effets de bord :

. Sg est la phase d'accélération

. Système Itératif représente les contraintes de croisière

. $Sd_{/n}$, la phase de décélération .

Dans les règles récursives classiques, ces effets de bord sont de taille presque nulle. Mais dans certains cas extrèmes, les effets de bord peuvent être de taille exponentielle par rapport au nombre de variables de la règle :



Si la règle contient k variables (x,y, . . , z) , il faut attendre 2^k réécritures pour que cette règle atteigne sa phase de croisière, puisqu'elle devient alors invariante : appliquer plus de 2^k réécritures de la règle ci-dessus revient à appliquer la règle constante (i.e. ayant le même arbre clos en tête et corps de clause) :

Par exemple pour k = 1, la règle est donc :

Chap 5 (suite) - Décidabilité de la terminaison d'une règle de réécriture en tête.

Grâce à cette caractérisation itérative de la récursivité d'une règle, l'évolution des contraintes lors de la réécriture peut être parfaitement quantifiée. La plupart de ces informations sont extraites du Gop caractéristique de la règle.

Si l'on note $t_{1,n} \rightarrow t_{n+1,n}$ la règle équivalente à n applications de $\beta \rightarrow \gamma$, alors l'évolution de ces arbres $t_{1,n}$ et $t_{n+1,n}$, la croissance linéaire de leurs branches poussantes et les permutations de leurs branches stables (période) peuvent être lues sur le Gop $\beta^0 \vee \gamma^{-1}$:

. la longueur des boucles élémentaires et leur pondération caractérisent le taux de croissance des branches des arbres $t_{1,n}$ et $t_{n+1,n}$; notons nrg_{min} le Nombre de Réécritures minimales nécessaires pour faire pousser au moins de hauteur un toutes les branches poussantes (quand n croît) de l'arbre Gauche $t_{1,n}$:

nrg_{min} = Sup { pondération/longueur / ∀ boucle élémentaire positive de $\beta^0 \vee \gamma^{-1}$ }

. les congruences des variables caractérisent la période des branches stables de $t_{1,n}$ et $t_{n+1,n}$:

la période de la règle r est le PPCM des congruences de $\beta^0 \vee \gamma^{-1}$.

Ces informations, par leurs précisions quantitatives, nous ont permis de démontrer de façon presque immédiate la propiété de terminaison d'une règle ainsi que certaines propriétés de décidabilité du Tant Que dans le cas où la question et le fait sont linéaires (i.e. une seule occurrence de chaque variable).

Rappel : Une règle r satisfait la propriété de terminaison ssi : ∀ T clos , T --r--> finiment

Théorème 3.- La propriété de terminaison est décidable pour une règle de réécriture en tête avec test d'occurrence.

La preuve repose sur l'existence ou le non-existence de branches arbitrairement longues dans l'arbre $t_{1,n}$.

Chap 5 (suite) - Décidabilité du Tant Que pour les questions et faits linéaires.

Théorème 4.- Pour une règle r et un terme fini et linéaire T donnés, la propriété de terminaison est décidable : r : ß -> ϑ

T fini et linéaire ?

Il existe une fonction linéaire f ne dépendant que de la règle récursive r telle que, pour tout terme fini T, r et T satisfont la propriété de terminaison ssi T ne peut être réécrit plus de f(hauteur(T)) fois par la règle.

Principe de preuve : Cette fonction linéaire de la règle est de la forme :

$$f(x) = nrg_{min} * (x+1) + \text{taille des effets de bord}$$

La croissance linéaire des branches poussantes de $t_{1,n}$, caractérisée par nrg_{min} est effective dans la phase de croisière, c'est la signification de la constante "taille des effets de bord".

=> Après f(Hauteur(T)), toutes les branches poussantes de $t_{1,n}$ sont de taille strictement supérieure à celles correspondantes dans T, les autres sont stables .

=> Si pour n_0 = F(Hauteur(T)) et T linéaire, il existe σ telle que $\sigma(T) \text{ --}r_{n0}\text{-->}$ alors cette propriété reste vraie quel que soit $n > n_0$: $\forall n > n_0$, $\exists \sigma$, $\sigma(T) \text{ --}r_n\text{-->}$

Cas particulier.- Ce théorème s'applique sur tout terme clos fini.

Revenons à notre problème Tant Que Prolog, en supposant que la question T et le fait α sont linéaires (i.e. une occurrence de chaque variable) :

Tant que : r_1 α -> ;

r_2 ß -> ϑ ;

T -> ?

Théorème 5.- Si T et α sont linéaires, l'existence de solutions au Tant Que Prolog est décidable en un nombre de réécritures linéairement dépendant de la hauteur de la question T.

Le Tant Que admet une solution ssi il existe, pour $n \leq f'$ (Hauteur(T)), une substitution σ_n telle que : $\sigma_n(T) \text{ --}r_2^n.r_1\text{-->}$

Principe de la preuve : La fonction f' linéaire est un peu différente : elle tient compte de la taille du fait α et de la période de la règle :

$$f'(x) = f(x) + cste_1 + \text{période de la règle}$$

a - La croissance des branches de l'arbre $t_{n+1,n}$ est linéaire ; après un nombre de réécritures indépendant de la question, toutes les branches poussantes de $t_{n+1,n}$ sont de taille strictement supérieure à celles correspondantes dans α : c'est la signification de $cste_1$.

b - La permutation des branches constantes lors de la réécriture nous oblige à attendre une période de la règle dans la recherche d'une solution éventuelle.

Pour un nombre de réécritures supérieur à f'(Hauteur(T)), la résolution du Tant Que devient invariante : une solution pour n+période(r_2) réécritures existe ssi une solution pour n réécritures existe aussi .

Corollaire.- Dans le cadre d'une résolution avec test d'occurrence et recherche de toutes les solutions, l'arrêt du Tant Que est décidable si T et α sont linéaires; cette décidabilité s'obtient en un nombre de réécritures linéairement dépendant de la hauteur de la question.

Si T et α sont linéaires, le Tant Que admet un ensemble infini d'entiers n pour lesquels :

$$\exists\, \sigma\ ,\ \sigma\,(T) \text{ --}r_2^{\,n}.r_1\text{-->}$$

ssi il existe $n \in [f'(\text{Hauteur}(T)\text{-période}(r_2)), f'(\text{Hauteur}(T))]$.

4) EXEMPLES

La Définition des Entiers Naturels.

Zéro -> ;

Succ (x) -> x ;

Nous savons que la séquence A d'arbres obtenus par n itérations de la règle est de la forme :

$Succ^n(x)$ -1^{ier}-> $Succ^{n-1}(x)$ ... -$(n-1)^{ième}$-> Succ(x) -$n^{ième}$-> x

a- Exprimons A sous forme d'unifié de 2 Igops :

$\beta^0 = (Succ,u)^0$ et $\gamma^{-1} = (x,x)^{-1}$

(x,x)

La séquence A est donc formalisée par : $I_{[1,n+1],[1,n]}(\beta^0) \vee I_{[1,n+1],[1,n]}(\gamma^{-1})$

b- Calculons le Igop caractéristique : $\beta^0 \vee \gamma^{-1} = {}^0(Succ,u) + 1$

$\beta^0 \vee \gamma^{-1}$ satisfait la Restriction sur les Pondérations de Boucles, cette règle récursive peut donc boucler (avec ou sans test d'occurrence). Le Igop unifié est parfait : toutes les congruences sont nulles . La séquence A de n réécritures est obtenue, ici, de façon exacte par interprétation du Igop unifié sur [1,n+1] et [1,n] :

$$A = I_{[1,n+1],[1,n]}(\beta^0 \vee \gamma^{-1})$$
$$= \{ (1,Succ^n(x_{n+1})), (2,Succ^{n-1}(x_{n+1})), .. ,(n ,Succ(x_{n+1})), (n+1,x_{n+1}) \}$$

c- Le Igop caractérise itérativement l'utilisation récursive de cette règle : $Succ^n(x_{n+1})$ -r^n-> x_{n+1}

1- le terme gauche de r^n croît linéairement de 1 en hauteur à chaque nouvelle réécriture :

boucle de pondération +1 et de longueur 1

2- le terme droit de r^n est stable par rapport à n : x_{n+1}

3- il n'y a aucun phénomène périodique : les congruences sont toutes nulles En conclusion de ces 3 remarques, on peut construire la fonction linéaire de la règle : f(x) = x+1 .

Pour toute question T, seules f(Hauteur(T)) réécritures sont nécessaires pour décider de l'existence de solutions et de l'existence d'une infinité de solutions.

Soient les questions suivantes :

- T_1 = y ($\in$ V) -- arbre de hauteur 0 : Décidabilité après 1 utilisation de la règle récursive.

Deux solutions : Zéro --r_1--> et Succ(Zéro) --$r_2.r_1$-->

Cette question admet une infinité de solutions car la solution pour une réécriture existe.

Solutions = { $Succ^n$ (Zéro) / $\forall n \in N$ }

- T_2 = $Succ^5$(Zéro) -- arbre de hauteur 5 : Décidabilité après 6 réécritures récursives.

Une seule existe : $Succ^5(Zéro) \xrightarrow{r_2^5 . r_1}$

C'est la seule solution, il n'y a pas de solution pour 6 réécritures .

- T_3 = $Succ^2$(Nil) -- arbre de hauteur 2 : Décidabilité après 3 réécritures récursives.

Aucune solution n'existe, car aucune n'est obtenue après 3 réécritures récursives.

Exemple 2

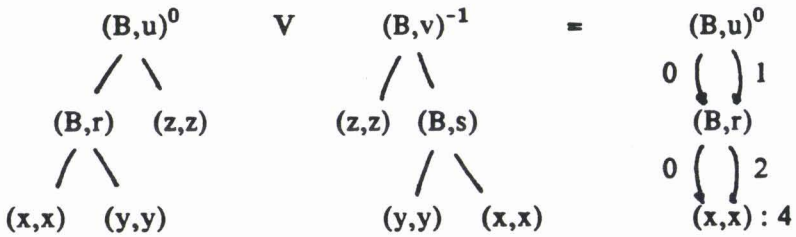
B (B(C,D) , B(E,F)) ->;

B (B(x,y) , z) -> B (z , B(y,x)) ;

Pour plus de clarté, notons la règle récursive sous forme arborescente :

r : ß = 

Calculons le Igop unifié :



. Le Igop Unifié satisfait la Restriction sur les Pondérations de Boucles

=> Cette règle ne satisfait pas la propriété de terminaison quel que soit le terme à réécrire.

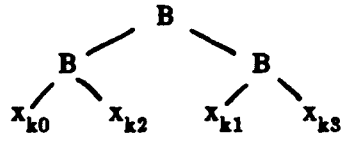
. La seule variable du Igop est de congruence 4

=> La règle est périodique : période 4.

. Le Gop caractéristique ne contient aucune boucle, donc toutes les branches de $t_{1,n}$ et $t_{n+1,n}$ sont stables modulo la congruence de la règle.

=> La fonction linéaire de la règle est : f(x) = 0*x + 4.

. L'interprété du Igop unifié sur l'intervalle [1,n] et pour la pondération d'entrée k est :

$A_{[1,n]}(G,k) =$ 

avec k0 = k mod 4
k1 = (k+1) mod 4
k2 = (k+2) mod 4
k3 = (k+3) mod 4 .

=> La règle se comporte récursivement comme la règle de "permutation" suivante :

Calculons de façon effective les réécritures de cette règle :

. $B(B(x_1,y_1),z_1)$ -r-> $B(z_1,B(y_1,x_1))$
. $B(B(x_1,y_1),B(x_2,y_2))$ $-r^2->$ $B(B(y_1,x_1),B(y_2,x_2))$
. $B(B(x_1,x_3),B(x_2,y_2))$ $-r^3->$ $B(B(y_2,x_2),B(x_1,x_3))$
. $B(B(x_1,x_3),B(x_2,x_4))$ $-r^4->$ $B(B(x_1,x_3),B(x_2,x_4))$ (inchangé)

Cette règle est bien de période 4, puisque r^4 est de la forme : t -> t
Dès la seconde réécriture, cette règle se comporte comme nous l'avons décrit, c-à-d qu'elle est équivalente à :

Le nombre de réécritures nécessaires pour décider de l'existence de solutions est indépendant de la taille de la question : f(x) = 0*x+4.
Soient les questions suivantes :

- T_2 = B(x,B(C,C)) -- Aucune solution en moins de 5 réécritures récursives :
  => aucune solution.
- T_3 = B(B(D,C),B(F,E)) -- une solution en moins de 5 réécritures récursives :
  $T --r_2^2.r_1-->$
  => Il existe une infinité de solutions : $T --r_2^{2+4k}.r_1-->$

Un exemple similaire est B(B(B(x,y),z),w) -> B(w,B(z,B(y,x))) ; (période 8)

Chapitre 2

Les Graphes Pondérés et leurs substitutions.

2-1) Préliminaires.

2-1-1) ARBRES ET SUBSTITUTIONS.

On se donne un alphabet fini gradué F et son application d'arité π , et un ensemble V dénombrable de variables (par convention d'arité nulle). $F_k = \{ f \in F / \pi(f) = k \}$.
On note N l'ensemble des entiers naturels et Z celui des entiers relatifs.

$M^{\infty}(F,V)$ désigne l'ensemble des arbres finis ou infinis construits sur l'alphabet F et l'ensemble de variables V, R(F,V) celui des arbres rationnels, c-à-d les arbres ayant un nombre fini de sous-termes distincts (assimilable aux Gos).

Un *arbre* est une fonction partielle T de $(N-\{0\})^*$ dans $(F \cup V)$ telle que :
(1) $Dom(T) = \{ m \in (N-\{0\})^* / T(m)$ est défini$\}$ est clos par préfixe
i.e. $m \in Dom(T)$ si $mm' \in Dom(T)$;
(2) $\forall m \in Dom(T)$ alors $mi \in Dom(T)$ ssi $1 \leq i \leq \pi(T(m))$.
On note T.m le sous-arbre de T issu du noeud m pour $m \in Dom(T)$ tel que $(T.m).m' = T.(m.m')$ pour tout $m' \in (N-\{0\})^*$.

Une *substitution* σ dans $M^{\infty}(F,V)$ est un ensemble dénombrable de couples de la forme (u,t) où
. u est une variable de V qui apparait au plus une fois en partie gauche des couples de σ
. t est un arbre quelconque de $M^{\infty}(F,V)$
. on notera $Dom(\sigma) = \{ u \in V / (u,t) \in \sigma \}$

$$Var(\sigma) = \bigcup_{(u=t) \in \sigma} Var(t)$$

Le *composé* $\sigma(T)$ (noté aussi $T.\sigma$) d'un arbre T et d'une substitution σ est l'arbre T dont toutes les occurrences de x ont été remplacées par t_x et ceci pour tous les couples (x,t_x) de σ.
Par extension, le composé de deux substitutions σ et σ' est défini comme suit :

$$\sigma.\sigma' = \{ (u,t_u.\sigma') \ / \ \forall \ (u,t_u) \in \sigma \} .$$

Ainsi définie la loi de composition est associative sous certaines conditions ; soient t un terme, arbre ou substitution et σ, σ' deux substitutions tels que Var(t) est inclus dans $Dom(\sigma)$ qui est lui-même inclus dans $Dom(\sigma')$ alors $(t.\sigma).\sigma' = t.(\sigma.\sigma')$.

On définit une relation de préordre sur $M^{\infty}(F,V)$. Soient T et $T' \in M^{\infty}(F,V)$:
. $T \leq T'$ si $\exists \sigma$ telle que $\sigma(T) = T'$
. $T \equiv T' \iff T \leq T'$ et $T' \leq T$.
Deux termes t et t' sont dits *unifiables* si $\exists \sigma$ telle que $\sigma(t) = \sigma(t')$.

2-1-2) CONGRUENCES D'INDICES.

Nous allons considérer par la suite des variables indicées et expliciter cet indice en étudiant l'ensemble dénombrable de variables VxZ.

DEFINITION 1.- On dira qu'une application Ci est une *congruence d'indices* sur VxZ, si Ci est une application de V dans N d'image non nulle sur un sous-ensemble fini de V et qui exprime les égalités suivantes sur VxZ : $\forall k \in Z$ et $v \in V$, $(v,k) = (v,k + Ci(v))$

REMARQUE 2.- Les congruences d'indices sont des relations d'équivalence particulières sur VxZ. Si l'on étend la définition de la fonction Modulo de Zx(Z-{0}) à ZxZ, en posant que k modulo 0 est égal à k, on peut donc écrire : $\forall (v,k) \in VxZ$, (v,k) R $(v,k$ modulo $Ci(v))$.
$(v,k$ modulo $Ci(v))$ sera choisi comme représentant de (v,k) pour la congruence Ci.

DEFINITION 3.- On peut se définir quelques opérations sur ces congruences d'indices, opérations dérivées de celles sur les relations d'équivalences :

1) *Relation de préordre* **: $Ci \leq Ci'$ ssi $\forall x$ et $y \in VxZ$, $x\ R_{Ci}\ y \Rightarrow x\ R_{Ci'}\ y$**

2) *Unification* **: Ci V Ci' la plus petite relation d'équivalence supérieure à Ci et Ci', c-à-d la clôture transitive des relations de R_{Ci} et $R_{Ci'}$.**

PROPRIETE 4.- $Ci \leq Ci'$ ssi $\forall v \in V$, Ci(v) est multiple (éventuel nul) de Ci'(v).

Démonstration : Nous allons montrer dans un premier temps que pour tout Ci, congruence d'indices,et R la relation associée : (v,k) R $(v,k') \Leftrightarrow k = k'$ ou $|k-k'|$ est divisible par Ci(v)

. <=

si $k = k'$ alors (v,k) R (v,k) (R réflexive)

si $|k-k'|$ divisible par Ci(v), supposons pour simplifier que $k > k'$

alors il existe q tel que $k = k' + qCi(v)$

comme par hypothèse $(v,k') = (v,k'+Ci(v)) = \ldots = (v,k'+qCi(v)) = (v,k) \Rightarrow (v,k)$ R (v,k') .

. =>

(v,k) R (v,k'), supposons que $k > k'$, alors il existe une suite finie $k_1, k_2, \ldots, k_n$

telle que $k_1 = k' + Ci(v)$, $k_2 = k_1 + Ci(v)$, ..., $k = k_n + Ci(v) \Rightarrow$ $k-k'$ est divisible par Ci(v),

De façon symétrique si $k < k'$ alors $k'-k$ divisible par Ci(v).

Démontrons la propriété, appelons R et R' les relations d'équivalence associées à Ci et Ci'.

1) $Ci \leq Ci'$: soit $k \in Z$, $(v,k)\ R\ (v,k + Ci(v)) \Rightarrow (v,k)\ R'\ (v,k + Ci(v))$

Immédiat : soit $Ci(v) = 0$, soit $Ci(v)$ est divisible par $Ci'(v)$

2) Inversement :

. Si $Ci(v) = 0$ alors $(v,k)\ R\ (v,k')$ ssi $k = k'$ et $(v,k)\ R'\ (v,k)$ (R' réflexive)

. Si $Ci'(v)$ divise $Ci(v)$ et $Ci(v)$ non nul alors pour k, k' distincts

$(v,k)\ R\ (v,k') \Rightarrow k-k'$ divisible par $Ci(v)$

$\Rightarrow k-k'$ divisible par $Ci'(v)$

$\Rightarrow (v,k)\ R'\ (v,k')$

Donc $(v,k)\ R\ (v,k) \Rightarrow (v,k)\ R'\ (v,k)$

PROPRIETE 5. - L'unifié de deux congruences d'indices est une congruence d'indices.
De plus si l'on étend la définition du "plus grand commun diviseur " de $(N-\{0\})x(N-\{0\})$ à NxN, en posant : $\forall k \in N$, $Pgcd(k,0) = Pgcd(0,k) = k$ alors

$$\forall v \in V\ ,\ (Ci\ V\ Ci')(v) = Pgcd(Ci(v),Ci'(v)).$$

Démonstration : Appelons C la congruence d'indices définie à partir de Ci et Ci' de la façon suivante : $\forall\ v \in V$, $C(v) = Pgcd(Ci(v),Ci'(v))$.

i) En application de la propriété 5 : C est supérieur à Ci et à Ci'.

Si v est de congruence C(v), v est aussi de Congruence Ci(v) et Ci'(v).

ii) Réciproquement: v de congruence Ci(v) et Ci'(v) => v est de Congruence C(v) ?

Pour cela nous allons nous inspirer de l'algorithme d'Euclide du calcul du Pgcd :

. si $Ci(v) = 0$, $C(v) = Ci'(v)$ (de même si $Ci'(v)=0$, $C(v)=Ci(v)$)

. si $Ci(v) = Ci'(v)$, $C(v) = Ci(v) = Ci'(v)$

. si $Ci(v)$ et $Ci'(v)$ sont non nuls et différents, alors supposons que $Ci(v) > Ci'(v)$

v de congruence Ci(v) et Ci'(v) $\Rightarrow \forall k \in Z$, $(v,k) = (v,k+Ci(v)) = (v,k+Ci'(v))$

Posons $k = k'-Ci'(v) \Rightarrow \forall k' \in Z$, $(v,k'+Ci(v)-Ci'(v)) = (v,k')$

$\Rightarrow$ v est de congruence Ci'(v) et $(Ci(v)-Ci'(v))$.

On peut itérer le procédé, conformément à l'algorithme d'Euclide et obtenir ainsi le Pgcd de Ci(v) et Ci'(v).

2-1-3) ENSEMBLES ORDONNES D'ARBRES.

DEFINITION 6.- On appelle un ensemble ordonné d'arbres, une fonction de Z dans $M^{\infty}(F,VxZ)$ représentable sous la forme de couples (entier relatif, arbre) :

$$E = \{ (k,t_k) / \ k \in Z \ , t_k \in M^{\infty}(F,VxZ) \}$$

DEFINITION 7.- La relation de préordre que l'on utilisera sur les ensembles ordonnés T et T', est : $T \leq T'$ ssi $\exists \ \sigma$, $T . \sigma$ est inclus dans T'
(loi de composition ensembliste : $\{(k,t_k)\}.\sigma = \{(k,t_k.\sigma)\}$).
Deux ensembles ordonnés sont équivalents, $T \equiv T'$ ssi $T \leq T'$ et $T' \leq T$.

2-2) Les Graphes Ordonnés Pondérés.

2-2-1) DEFINITION SYNTAXIQUE.

DEFINITION 8.- On appelle *graphe ordonné pondéré (GOP)*, construit sur F et V, la donnée du triplet (X,Lab,Succ) où

- ***X* est un ensemble fini de sommets.**
- ***Lab* est une application de X dans FUV qui à chaque sommet associe une étiquette.**
- ***Succ* est une fonction partielle de XxN dans XxZ : *Succ* (s , i) = (s', q),**

s' est le $i^{ème}$ successeur du sommet s avec la "pondération" q.

DEFINITION 9.- On appellera un *GOP pointé*, la donnée d'un Gop, d'une racine pondérée et d'une congruence d'indices sur VxZ : *G / (Rac,Clas)*

- ***Rac* est un élément particulier de XxZ qui désigne la "racine" du graphe avec sa "pondération en tête".**
- ***Clas* est une congruence d'indices.**

NOTATIONS.-

. On note g .i, pour tout i entier positif non nul, inférieur à nsucc(s), le sous-GOP pointé de g dont la racine est Succ(s,i) et de même congruence Clas .

Si l'on pose g = G/(Rac,Clas) et Rac=(s,p) alors g .i = G/(Succ(s,i),Clas) .

. De même, on note $\forall$ m $\in$ $(N-\{0\})^*$: g .m le sous-GOP pointé de g *s'il existe* dont la racine est : pour $m=i_1 . . i_k$, Succ(Succ(. . .(Succ(s,i_1), i_2)...., i_k). On note m $\in$ Dom(g).

DEFINITION 10.- On se définit les chemins sur les graphes par des mots de $(N-\{0\})^*$ et la fonction *Descendant* de (XxZ)x$(N-\{0\})^*$ dans (XxZ) de manière récursive par :

Desc((s,p) , nil) = (s,p)

Desc((s,p) ,i.m) = Desc((s_i,p_i+p) , m) si Succ(s,i) = (s_i,p_i).

A partir d'un sommet de G, seul un sous-ensemble de sommets de X sont accessibles par la fonction Succ. Pour tout gop pointé g, on appellera *Var(g)*, l'ensemble des labels variables de ces sommets accessibles.

DEFINITION 11.- Deux GOPS pointés g et g' seront dits *isomorphes*, noté g = g', si :

. X et X' sont isomorphes pour les fonctions Lab,Succ et Rac de g et g'.

. les congruences d'indices de g et g' sont identiques.

DEFINITION 12.- Soient g un Gop et q un entier relatif, on notera translaté(g,q), le Gop pointé g' dont la pondération de la racine Rac' est celle de Rac incrementé de q :

$$\text{translaté}\ (\ G/((s,p),Clas)\ ,\ q\ ) \ = \ G\ /\ ((s,p+q),Clas).$$

2-2-2) INTERPRETATION D'UN GOP

DEFINITION 13.- L'interprétation I(g) d'un Gop pointé g est un ensemble ordonné d'arbres :

$$I(g) = \{\ (\ k\ ,\ A(g,k)\ )\ /\ \forall\ k \in Z \ \text{ et } \ A(g,k) \in M^{\infty}(F,VxZ)\ \}$$

. $\forall$ m $\in$ Dom(g), notons Desc(Rac,m) = (s,p) et Lab(s) = x,

$$A(g,0)(m) = x \qquad \text{si } x \in F$$

$$= x_{p \text{ modulo Clas}(x)} \qquad \text{si } x \in V.$$

. A(g,k) = A(translaté(g,k),0).

Intuitivement, dans A(g,k), toutes les pondérations sur les arcs sont effacées, et on associe à chaque symbole feuille de V, un indice égal à la somme des pondérations rencontrées le long du chemin dont la tête de pondération k . (Cette somme est calculée modulo la congruence d'indices).

Remarque : Tous les arbres de I(g) sont équivalents pour la relation de préordre classique sur $M^{\infty}(F,VxZ)$.

PROPRIETE 14.- Si l'on pose $R^{\infty}(F,VxZ) = \{\ A(g)\ /\ g \in G^{z}(F,V)\ \}$ alors

$$R(F,VxZ) \subsetneq R^{\infty}(F,VxZ) \subsetneq M^{\infty}(F,VxZ).$$

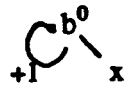
Démonstration :

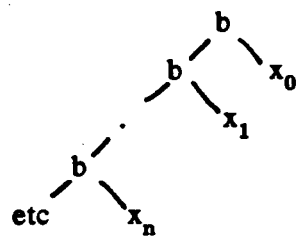
. $R(F,VxZ) \subsetneq R^{\infty}(F,VxZ)$?

A chaque Go de R(F,VxZ), on peut associer facilement un Gop ayant le même interprété en transformant la fonction Succ du Go en celle du Gop.

Posons le Go $(X_r,Succ_r,Lab_r)$ et son Gop associé (X,Succ,Lab) où $Succ_r$ est une fonction de XxN dans X et Succ de XxN dans XxZ alors : $\forall\ (s,i) \in X$, $Succ(s,i) = (Succ_r(s,i),0)$

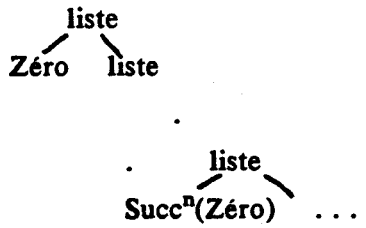
Mais il existe des Gops g tels que A(g) n'appartient pas à R(F,VxZ) :

Soit g =  où b ∈ F, v ∈ V , Clas(x) = 0

A(g,0) =  arbre irrationnel

. $R^{\infty}(F,VxZ) \subsetneq M^{\infty}(F,VxZ)$

L'irrationalité des Gops est dûe au nombre infini de variables de leur interprété. La plupart des arbres irrationnels ne sont pas des interprétés de Gops, en particulier aucun des arbres fermés (sans variables), puisque un Gop "fermé" a un interprété rationnel. Ainsi, l'arbre suivant ne peut pas être obtenu par interprétation d'un Gop.



PROPRIETE 15.- Deux gops pointés isomorphes ont même interprété :

g = g' => I(g) = I(g') c-à-d A(g) = A(g') et Clas = Clas' sur Var(g) ∪ Var(g')

(La réciproque est fausse)

Démonstration : Découle directement de la définition de l'interprété.

2-2-3) CONGRUENCE DE SOMMETS OU DE GOPS POINTES.

DEFINITION 16.- Soit g, un Gop pointé, on dira par extension que *g est de congruence c* ou encore que le sommet s pour la fonction Clas est de congruence c si g et translaté(g,c) ont même interprété : I(g) = I(translaté(g,c))

PROPRIETE 17.- Soit g, un Gop pointé, g est de congruence c ssi c est un multiple du PPCM des congruences des variables de g sachant que :

$\forall$ E inclus dans Z, PPCM ({0} $\cup$ {E}) = 0 et PPCM ($\varnothing$) = 1

Démonstration : Soit v, une variable de g, il existe un chemin m de Dom(g) tel que :

Desc(Rac,m) = (s_v,p) et Lab(s_v) = v, c-à-d $A(g,0)(m) = v_{p \bmod Clas(v)}$

Or la définition de l'interprété nous donne aussi : $A(g,k)(m) = v_{p+k \bmod Clas(v)}$

et g est de congruence c : $v_{p+k \bmod Clas(v)} = v_{p \bmod Clas(v)}$ si k est un multiple de c et cette équation est déductible de la congruence de v que si c est un multiple de Clas(v).

Ceci étant vrai pour toutes les variables de g, c est un multiple du PPCM des congruences des variables de v.

CORROLAIRE 18.-

. Tout gop pointé est au moins de congruence nulle.

. Tout gop pointé sans variable est de congruence 1.

. Tout sommet de label constante est de congruence 1

. Tout sommet de label variable v est de congruence Clas(v)

. Tout sommet s est de congruence c, Pgcd des congruences de ses sommets "père".

2-3) Les Substitutions Pondérées.

2-3-1) DEFINITION SYNTAXIQUE.

DEFINITION 19.- On appelle *substitution pondérée* σ , la donnée d'un Gop G , d'une fonction Sub de V dans XxZ, et d'une congruence d'indices Clas :

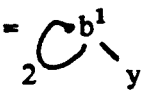
$$\sigma = G/(Sub,Clas)$$

sachant que si $Sub(u) = Rac_u$ alors $g_u = G/(Rac_u,Clas)$ est de congruence Clas(u), c-à-d qu'il y a cohérence entre U et Sub(u) dans leur congruence.

On notera : $Dom(\sigma) = Dom(Sub) = \{ u \in V / Sub(u)$ est définie $\}$

$$Var(\sigma) = \cup\ Var(g_u) \quad où\ u \in Dom(\sigma)$$

Exemples :

1) $Sub(x) =$  , $Sub(z) = a^2$, $Sub(w) = w^0$ avec w, x, y, z $\in$ V et a, b $\in$ F

si x est de congruence 3 => y est de congruence diviseur de 3, c-à-d 1 ou 3

w et z sont de congruences quelconques.

2) Seule information dans σ , une congruence définie sur u et v.

par ex : Sub = $\emptyset$ et Clas(u) = 2 et Clas(v) = 5

DEFINITION 20.- Soient σ et σ' deux substitutions pondérées quelconques σ et σ' seront dites *isomorphes*, noté $\sigma = \sigma'$ si elles ont même domaines et congruences et si

$$\forall\ (u,Rac_u) \in \sigma \text{ et } (u,Rac'_u) \in \sigma' ,\ G/(Rac_u,Clas_\sigma) = G'/(Rac'_u,Clas_{\sigma'})$$

Remarque : On pourra toujours supposer qu'un ensemble de Gops pointés et de substitutions pondérées partage le même GOP G = (X, Lab et Succ), c-à-d qu'on peut construire des gops pointés et de substitutions pondérées isomorphes et définies sur un Gop commun.

2-3-2) LOI DE COMPOSITION.

DEFINITION 21-. Manipulation syntaxique élémentaire.
Soient G un Gop pointé et s, s' deux sommets de G, G[s <- (s',p')] représente le Gop dans lequel le sommet s a été remplacé par le sommet de s' avec une incrémentation p' des pondérations :

$G' = G[s \leftarrow (s',p')] = (X, Lab, Succ')$ -- c-à-d X et Lab inchangés

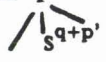
et $\forall s'' \in X$, $Succ'(s'',i) = (s, p'+q)$ si $Succ(s'',i) = (s',q)$

$= Succ(s'',i)$ sinon.

Soient g = G/((s,p),Clas) un Gop pointé et (s',p') un sommet pondéré de G, g[s <- (s',p')] représente le Gop pointé suivant : $G'/(Rac',Clas)$ avec $G' = G[s \leftarrow (s',p)]$

et $Rac' = (s' , p+p')$ si $s = s'$

$= Rac$ sinon.

Exemple : si g =  et alors g[s <- (s',p')] = 

DEFINITION 22.- Soient g un Gop pointé et σ une substitution pondérée construits sur le même gop G. Si la congruence de g est inférieure à celle de σ, le *composé* de g et de σ est le Gop g dont tous les sommets de label u appartenant à Dom(σ) ont été remplacés par Rac_u , et dont la nouvelle congruence d'indice est Clas' .
Posons g = (X,Lab,Succ)/(Rac,Clas) et σ = (X,Lab,Succ)/(Sub,Clas'), si Clas $\leq$ Clas' alors :

. $g.\sigma = g\,[s \leftarrow Rac_u]$ pour tous les sommets s avec Lab(s) = u $\in$ Dom(σ)

. Clas de $g.\sigma$ = Clas'

2-3-3) INTERPRETATION D'UNE SUBSTITUTION PONDEREE.

DEFINITION 23.- ***L'interprétation d'une substitution pondérée*** **σ est la substitution I(σ) définie comme suit :**

$$I(\sigma) = \{ u_k = A(g,k) \ / \ \forall k \in Z \text{ et } u \in Dom(\sigma)\}$$
$$\cup \{ u_k = u_{k \bmod Clas(u)} \ / \ \forall u \in V\text{-}Dom(\sigma) \text{ et } Clas(u) \neq 0 \}$$

Exemple : $\sigma = \{ (u = \overset{b^{-1}}{\circlearrowright}_{1} \; y^{3}) \}$ et Clas (y:3),(u:6)

$I(\sigma) = \{ (x_k = \; b \; (\; b \; (\; . \; (\; b \; (\; . \; (\; y_{k+n \bmod 3}) \; y_{k \bmod 3}) \; y_{k-1 \bmod 3})))) \; / \; \forall k \in Z \} \; \cup \; \{ y_k = y_{k \bmod 3} \; / \; \forall k \in Z \}$

PROPRIETE 24.- Deux substitutions pondérées isomorphes ont même interprété.

$\sigma = \sigma' \Rightarrow I(\sigma) = I(\sigma')$. (La réciproque est fausse)

2-3-4) INTERPRETATION DU COMPOSE D'UN GOP ET D'UNE SUBSTITUTION.

PROPRIETE 25.- Soient g un gop pointé et σ une substitution pondérée : $I(g.\sigma) = I(g).I(\sigma)$

Démonstration : montrons que $\forall k \in Z$, $A(g.\sigma,k) = A(g,k).I(\sigma)$

Soient $G = (X,Lab,Succ)$, $g = G/(Rac,Cg)$ et $\sigma = G/(Sub,C\sigma)$

Notons Desc et Desc', les fonctions Descendant de g et $g.\sigma$=G'/Rac'

et g_u le gop pointé de racine $Sub(u)=(s_u,q_u)$

Quelque soit $m \in Dom(g.\sigma)$:

(1) soit $m = m_1.m_2$ tel que $Desc(Rac+k,m1) = (s,p_s)$, $Lab(s) = u \in Dom(\sigma)$,

$\Rightarrow Desc'(Rac'+k,m1) = (s_u,p_u+p_s)$ c-à-d $A(g.\sigma,k)(m) = A(g_u,p_s+k)(m_2)$

or $A(g,k)(m_1) = u_p$ avec $p = ps \bmod Cg(u)$ et $(u_p = A(g_u,p)) \in I(\sigma)$.

De plus g_u est de congruence $C\sigma(u)$, donc aussi $Cg(u)$

$\Rightarrow A(g_u,p) = A(g_u,p_s)$ donc $(A(g,k).I(\sigma))(m) = A(g_u,p)(m_2) = A(g_u,p_s)(m2) = A(g.\sigma,k)(m)$

(2) soit il n'existe pas $m_1m_2 = m$ tel que $Desc(Rac,m1) = (s,q_s)$ avec $Lab(s) = u \in Dom(\sigma)$;

alors le chemin m reste inchangé et Rac' = Rac :

$\Rightarrow Desc'(Rac+k,m) = Desc(Rac+k,m) = (s,p)$

si $Lab(s) = f \in V$ alors $A(g.\sigma,k)(m) = A(g,k)(m)$

si $Lab(s) = u \in V$ alors $A(g.\sigma, k)(m) = u_{p \bmod C\sigma(u)}$

et $(A(g,k).I(\sigma))(m) = u_{p \bmod Cg(u)}$.

Or $\forall k \in Z$, $(u_k = u_{k \bmod C\sigma(u)}) \in I(\sigma)$, en particulier pour $k = p \bmod Cg(u)$

Comme $Cg \leq C\sigma$, on obtient : $A(g.\sigma,k)(m) = (A(g,k).I(\sigma))(m)$.

2-4) Relation de préordre et Equivalence sémantique.

2-4-1) RELATION DE PREORDRE.

Les GOPS permettent donc de caractériser certains ensembles finis ou infinis d'arbres de $M^{\infty}(F,VxZ)$.

RAPPEL : Soient T et T' deux ensembles ordonnés d'arbres, $T \leq T'$ si il existe une substitution quelconque Σ telle que $T \,.\, \Sigma$ est inclus dans T'

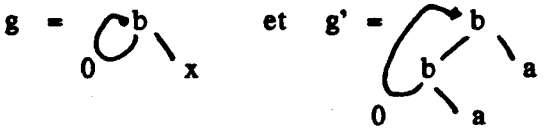
DEFINITION 26.- . Soient g et g' deux gops pointés, $g \leq g'$ ssi $I(g) \leq I(g')$
Deux gops pointés g et g' seront donc dits équivalents, noté $g \equiv g'$ si : $g \leq g'$ et $g' \leq g$.
De même, soient deux substitutions pondérées σ et σ', $\sigma \leq \sigma'$, $\exists\, \Sigma$, $I(\sigma) \,.\, \Sigma = I(\sigma')$.
Deux substitutions pondérées σ et σ' seront dites équivalentes, noté $\sigma \equiv \sigma'$ si $\sigma \leq \sigma'$ et $\sigma' \leq \sigma$.

Exemples : Soit g = b^0 (a, x^p) et x de congruence k.

Soit g'= b^0 (a, y^q) et y de congruence k.

Substitutions - g -> g' : (x = (y,q-p) , x et y de congruence k
g' -> g : (y = (x,p-q)) , x et y de congruence k

Remarque : Il existe g et g' telle que $g \leq g'$ et il n'existe pas σ, une substitution pondérée telle que $g.\sigma = g'$:



2-4-2) EQUIVALENCE SEMANTIQUE.

DEFINITION 27.- Deux GOPS pointés g et g' sont *sémantiquement équivalents* ssi ils ont la même interprétation : $g \sim g'$ ssi $I(g) = I(g')$

PROPRIETE 28.- Deux Gops pointés g et g' sont équivalents sémantiquement :

ssi $A(g) = A(g')$ et $Clas = Clas'$ sur $Var(g) \cup Var(g')$

ssi $\forall$ m chemin, posons $Desc((s,p),m) = (s,p)$, $Desc((s',p),m) = (s',p')$,

. $Lab(s) = Lab'(s')$

. Si $Lab(s) = x \in V$, $Clas(x) = Clas'(x)$ et $p \equiv p'$ modulo $Clas(x)$.

PROPRIETE 29.- $g \sim g' \Rightarrow g \equiv g'$ (La réciproque est fausse)

DEFINITION 30.- Deux Substitutions pondérées σ et σ' sont *sémantiquement équivalentes* ssi elles ont la même interprétation : $\sigma \sim \sigma'$ ssi $I(\sigma) = I(\sigma')$

PROPRIETE 31.- Deux Substitutions pondérées σ et σ' sont sémantiquement équivalentes ssi :

. $Clas_\sigma = Clas_{\sigma'}$

. $\forall u \in Dom(\sigma) \cap Dom(\sigma')$, $g_u \sim g'_u$

$\forall u \in Dom(\sigma) - Dom(\sigma')$, $g_u \sim u^0$

$\forall u \in Dom(\sigma') - Dom(\sigma)$, $g'_u \sim u^0$

2-5) Unification des Gops et de leurs interprétés.

2-5-1) DEFINITION ET PROPRIETES.

DEFINITION 32.- Deux gops pointés g_1, g_2 sont dits *unifiables* ssi il existe un ensemble ordonné d'arbres qui majore $I(g_1)$ et $I(g_2)$: $\exists\ \sigma$ de $M^{\infty}(F,VxZ)$, $I(g_1).\sigma = I(g_2).\sigma$.
Un unifié de g1 et g2 est le plus petit ensemble ordonné d'arbres, noté $g_1\ V\ g_2$ vérifiant :

$$g_1\ V\ g_2 = I(g_1).\sigma = I(g_2).\sigma\ .$$

PROPRIETE 33.- Soient $g_1 = G/(Rac_1,Clas_1)$ et $g_2 = G/(Rac_2,Clas_2)$ alors

$$g_1\ V\ g_2 = G/(Rac_1,Clas)\ V\ G/(Rac_2,Clas)$$

avec $Clas = Clas_1\ V\ Clas_2$ sur les variables communes de g_1 et g_2.

c-à-d
$Clas(u) = Pgcd(Clas_1(u),Clas_2(u))$ si $u \in Var(g_1) \cap Var(g_2)$
$= Clas_1(u)$ ou $Clas_2(u)$ si $u \in Var(g_1)-Var(g_2)$ ou $\in Var(g_2)-Var(g_1)$
$= 0$ si $u \in V - Var(g_1) \cup Var(g_2)$

Démonstration : Notons σ, une substitution vérifiant : $\forall\ k \in Z,\ A(g_1,k).\sigma = A(g_2,k).\sigma$.
Soit $u \in Var(g_1) \cap Var(g_2)$, il existe m_1 et m_2 tels que :
$A(g_1,k)(m_1) = u_{p1}$ et $A(g_2,k)(m_2) = u_{p2}$ comme $\forall\ k \in Z,\ A(g_1,k).\sigma = A(g_2,k).\sigma$
. $[A(g_1,k).\sigma](m_1) = [A(g_2,k).\sigma](m_1)$ et u est de congruence $Clas_1(u) = c_1$
$\Rightarrow \forall\ k \in Z\ ,\ [A(g_2,k).\sigma](m_1) = [A(g_2,k+c_1)](m_1)$
. $[A(g_2,k).\sigma](m_2) = [A(g_1,k).\sigma](m_2)$ et u est de congruence $Clas_2(u) = c_2$
$\Rightarrow \forall\ k \in Z,\ [A(g_1,k).\sigma](m_2) = [A(g_1,k+c_2)](m_2)$
Dans σ , la variable u est de congruence c_1 et c_2, c-à-d $Pgcd(c_1,c_2)$ dans l'unifié.

CORROLAIRE 34.- Soient g et g' deux Gops à unifier, on peut se ramener à l'unification de deux Gops ne différant donc que par leur racine pondérée.

PROPRIETE 35.- Soient g = G/((s,p),Clas) et g' = G/((s',p'),Clas) avec s et s' différents, posons G' = G[s'<-(s,p-p')], h = G'/((s,p),Clas) et h' = G'/((s',p'),Clas) si I(g) V I(g') = I(g).σ = I(g').σ alors I(h).σ = I(h').σ = I(g) V I(g')

Démonstration : Par récurrence sur la longueur des chemins m des Dom's.

Soient I(g) V I(g') = { (k,tg_k) } , I(h).σ = { (k,th_k) } et I(h').σ = { (k,th'_k) }

Desc la fonction descendant sur G et Desc' la fonction descendant sur G'.

Montrons que $\forall m \in Dom(tg_k) \Rightarrow m \in Dom(th_k)$ et $Dom(th'_k)$

et les sommets correspondants portent la même étiquette.

. m = nil. Les racines de h et h' étant inchangés :

$(A(g,k).\sigma)(nil) = (A(h,k).\sigma)(nil)$ et $(A(g',k).\sigma)(nil) = (A(h',k).\sigma)(nil)$

$\Rightarrow tg_k(nil) = th_k(nil) = th'_k(nil)$ et $nil \in Dom(th_k) \cap Dom(th'_k)$.

. $|m| = n > 0$ on pose $m = i_1.i_2. \ldots .i_n$

1) S'il n'existe pas $j \leq n$ tel que $Desc((s,p+k),i_1..i_j) = (s',q)$, c'est à dire que le chemin m ne passe pas par s'. Le chemin m issu de s est inchangé dans G'

$\Rightarrow (A(g,k).\sigma(m)) = (A(h,k).\sigma)(m)$ c-à-d $tg_k(m) = th_k(m)$ et $m \in Dom(th_k)$.

De même s'il n'existe pas $j' \leq n$ tel que $Desc((s',p'+k),i_1..i_{j'}) = (s',q)$ c-à-d que le chemin m ne passe pas par s'. Le chemin m issu de s' est inchangé dans G'

$\Rightarrow (A(g',k).\sigma(m)) = (A(h',k).\sigma)(m)$ c-à-d $tg_k(m) = th'_k(m)$ et $m \in Dom(th'_k)$.

2) S'il existe $j \leq n$ tel que $Desc((s,p+k),i_1..i_j) = (s',q)$, posons $m = m_1.m_2$ avec $m_1 = i_1 ..i_j$. $m_1 <> nil$ car s et s' sont différents et $Desc'((s,p+k),m_1) = (s,q+p-p')$

$\Rightarrow th_k(m) = (A(h,k).\sigma)(m) = (A(h,q-p').\sigma)(m_2) = th_{q-p'}(m_2)$

et $tg_k(m) = (A(g,k).\sigma)(m) = (A(g',q-p').\sigma)(m_2) = tg_{q-p'}(m_2)$

Or par hypothèse de reccurence ($|m_2| < |m|$) : $th'_{q-p'}(m_2) = tg_{q-p'}(m_2)$ et $m_2 \in Dom(th_{q-p'})$

$\Rightarrow th_k(m) = tg_k(m)$ et $m \in Dom(th_k)$

De même s'il existe $0 < j' \leq n$ tel que $Desc((s',p'+k),i_1..i_{j'}) = (s',q)$, posons $m = m_1.m_2$ avec $m_1 = i_1..i_j <> nil$ et $Desc'((s,p+k),m_1) = (s,q+p-p')$

$\Rightarrow th'_k(m) = (A(h',k).\sigma)(m) = (A(h,q-p').\sigma)(m_2) = th_{q-p'}(m_2)$

et $tg_k(m) = (A(g',k).\sigma)(m) = (A(g',q-p').\sigma)(m_2) = tg_{q-p'}(m_2)$

Or par hypothèse de reccurence ($|m_2| < |m|$) : $th_{q-p'}(m_2) = tg_{q-p'}(m_2)$ et $m_2 \in Dom(th_{q-p'})$

$\Rightarrow th'_k(m) = tg_k(m)$ et $m \in Dom(th_k)$

RAPPEL.- Soit g, g' deux gops de même sommet et de pondération tête p et p' :

$g \sim g'$ ssi $g' = $ translaté$(g, p-p')$ c-à-d ssi g et g' sont de congruence $|p-p'|$.

COROLLAIRE 36.- On ne change pas de problème d'unification si avant d'unifier g et g', on applique certaines manipulations syntaxiques , noté $G[s' <- (s,p)]$:

a) la nouvelle congruence des variables accessibles à partir de s est égale au Pgcd de l'ancienne congruence et de, soit $|p-p'|$ si $s = s'$, soit Clas(Lab(s') si s' est étiqueté d'une variable,

b) on remplace le sommet s' par s avec incrémentation p-p' des pondérations.

Ces manipulations syntaxiques peuvent être caractérisées par une substitution pondérée σ que nous noterons : $s' <- (s,p)$.

Posons $\sigma = G/(Sub\sigma, Clas\sigma)$ et soit Clas, la congruence de g et g'

. si $s = s'$ alors $Sub\sigma = \emptyset$ et $Clas_\sigma(v) = Pgcd(|p'-p|, Clas(v))$ si $v \in Var(g)$

$= Clas(v)$ sinon

. si $s <> s'$ et $Lab(s') \in V$ alors $Sub\sigma(Lab(s')) = (s, p-p')$

et $Clas_\sigma(v) = Pgcd(Clas(Lab(s)), Clas(v))$ si $v \in Var(g)$.

$= Clas(v)$ sinon

. si $s <> s'$ et $Lab(s) \in F$ alors $Sub\sigma = \emptyset$ et $Clas\sigma = Clas$

2-5-2) ALGORITHME D'UNIFICATION.

Soit G le Gop de représentation des Gops pointés g et g' de racine respectivement Rac =(s,p) et Rac'=(s',p'). Pour tout sommet s, notons :

Nsucc(s) le nombre de successeurs de s (l'arité de Lab(s))

Succ(s,i) le ième successeur pondéré de s, Succ(s,i).som désignera le sommet et Succ(s,i).pond la pondération

Estvar(s) = vrai si s est étiqueté par un symbole de variable
= faux sinon,

Nous donnons l'algorithme sous forme d'une fonction récursive unificateur(Rac,Rac'), qui calcule un unificateur principal de g et g' ou bien échoue si les gops ne sont unifiables :

```
Fonction unificateur( (s,p) , (s',p') )
Début
    Si s = s'
        Alors
            G := G[s<-(s,|p-p'|)]
            Résultat s<-(s,|p-p'|)
        Sinon
            Si Estvar(s')
                Alors  G := G[s'<-(s,p-p')] ; Résultat s'<-(s,p-p')
                Sinon
                    Si Estvar(s)
                        Alors  G := G[s<-(s',p'-p)] ; Résultat s<-(s',p'-p)
                        Sinon
                            Si Lab(s)=Lab(s')
                                Alors
                                    G := G[s'<-(s,p-p')]; σ := ∅ ;
                                    Pour i := 1 A Nsucc(s) Faire
                                        s1 := Succ(s,i).som;   p1 := p+Succ(s,i).pond;
                                        s1':= Succ(s',i).som;  p2 := p'+Succ(s,i).pond;
                                        σ:= unificateur( (s1,p1), (s1',p1') ) . σ ;
                                    Fait;
                                    Résultat σ ;
                                Sinon Echappement : Echec
                            Esi;
                    Esi;
                Esi;
        Esi;
Fin;
```

Terminaison de l'algorithme : A chaque appel récursif, un noeud est indirigé, donc disparait.

CORROLAIRE 37.- L'existence de l'unifié de deux gops pointés ne dépend pas des pondérations des arcs, les valeurs des pondérations influent uniquement sur celles de l'unifié et sa congruence d'indices.

PROPRIETE 38.- L'unification des Gops est une opération interne.

Soient g_1 et g_2 deux gops pointés, leurs interprétés sont unifiables ssi il existe un Gop g tel que :

$I(g) = I(g_1) \vee I(g_2)$, g est donc noté $g_1 \vee g_2$.

Il existe une substitution pondérée telle que $g_1 \vee g_2 = g_1.\sigma = g_2.\sigma$

Démonstration : Découle directement de l'algorithme d'unification.

On peut généraliser facilement la loi de composition et définir celle de deux substitutions, opération interne et associative dans le cadre de notre algorithme.

ALGORITHME IMPLEMENTE :

On suppose ici que les sommets variables sont partagées dans le Gop G commun aux deux gops pointés à unifier, c-à-d qu'il n'existe deux sommets de G de même label variable.

Comme dans l'algorithme de Fages, la complexité de cet algorithme dépend crucialement de la façon dont on substitue deux sous-gops (sommet ou congruence). Le remplacement physique oblige à connaître les prédécesseurs des sommets, donc à gérer des pointeurs arrières.

Nous préférerons remplacer les sommets en créant des indirections et distribuer les congruences aux feuilles de symbole variable, par simple parcours de graphe.

Pour chaque sommet on associera une fonction :

Ind(s') = (s,p) correspondant à G remplacé par G[s' <- (s,p-p')] si le sommet est indirigé

= (0,0) sinon.

Cette fonction a deux champs que l'on nommera Ind(s).som et Ind(s).pond .

Les sommets substitués ou remplacés forment donc des gops par la fonction Ind, dont la racine est le représentant canonique. le calcul du représentant canonique consiste à descendre le chemin d'indirection en sommant les pondérations :

```
Fonction repr(s,p) =
Début
|     Tant Que Ind(s) <> (0,0) Faire
|     |     (s,p) := repr(Ind(s).som,Ind(s).pond+p);
|     Fait;
|     Résultat (s,p)
Fin;
```

On en déduit donc une procédure unifier(Rac,Rac') de deux Gops de racine (s,p) et (s',p') qui s'arrête en échec si les Gops ne sont pas unifiables. En cas de succès, la substitution associée est un unificateur principal qui peut être obtenue facilement par exploration des indirections à partir des noeuds de Label variable.

```
Procédure unifier(Rac,Rac')
Début
    Si s = s'
        Alors Congruence(s,|p-p'|)
        Sinon
            Si Estvar(s')
                Alors
                    Ind(s') := (s, p-p');
                    Congruence( s, Clas(Lab(s')) )
                Sinon
                    Si Estvar(s)
                        Alors
                            Ind(s) := (s',p'-p);
                            Congruence( s', Clas(Lab(s)) )
                        Sinon
                            Si Lab(s) = Lab(s')
                                Alors
                                    Ind(s') := (s,p-p');
                                    Pour i := 1 A Nsucc(s) Faire
                                        s1  := repr(Succ(s,i)).som;
                                        p1  := p + repr(Succ(s,i)).pond;
                                        s1' := repr(Succ(s',i)).som;
                                        p1' := p' + repr(Succ(s,i)).pond;
                                        unifier((s1,p1),(s1',p1'));
                                    Fait;
                                Sinon Echappement : Echec
                            Fsi;
                        Fsi;
                Fsi;
        Fsi;
Fin;
```

Avec une procédure de distribution des congruences : Congruence(s,c).
Pour permettre un parcours du graphe plus facile, on utilisera la fonction Nsucc' copie exacte de la fonction Nsucc que l'on pourra modifier sans altérer le graphe. En mettant à zéro, pour un sommet s, Nsucc'(s), on marque les arcs venant de s, et on évite d'emprunter plusieurs fois le même arc et donc empêcher tout bouclage.

```
Procédure congruence(s,c);        % s ∈ X et c ∈ N %
Début
|       Nsucc':=Nsucc;
|       distrib(s,c);
Fin;
```

```
Procédure distrib(s,c);
Début
|       Si Estvar(s) Alors Clas(Lab(s)) := Pgcd( Clas(Lab(s)) , c ) Fsi;
|       k := Nsucc'(s);
|       Si k <> 0
|       |   Alors
|       |       Nsucc'(s) := 0;
|       |       Pour i := 1 A k Faire
|       |       |       s_i :=repr( Succ(s,i) ).som ;
|       |       |       distrib(s_i,c)
|       |       Fait;
|       Fsi;
Fin;
```

=> **Unification des gops pointés g et g' de sommets (s,p) et (s',p') :**

1) **unification des congruences d'indices de g et g' sur leurs variables communes.**
2) **appel de la procédure unifier((s,p),(s',p'));**

COROLLAIRE 39.- Deux Gops pointés g_1, g_2 sont équivalents sémantiquement ssi lors de l'algorithme d'unification, aucune indirection et ni aucune nouvelle congruence n'est générée sur les sommets de label variable.

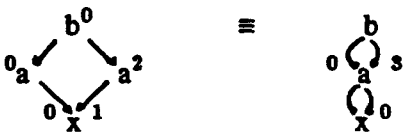
2-6) Equivalence syntaxique.

On veut ici se définir des modifications syntaxiques sur les gops pointés qui conservent l'interprétation. On peut appliquer l'une de ces quatre régles syntaxiques sans modifier l'équivalence sémantique. Soit g un gop pointé = (X,Succ,Lab)/(Rac,Clas).

PROPRIETE 40.- Emondage - Si un sommet s de X n'est pas accessible à partir de Rac, alors on peut remplacer X par X'=X-{s} et Lab, Succ par leur projection sur X', sans modification de l'interprétation de g. De même si une variable v n'est accessible à partir de g alors on peut remplacer Clas(v) par n'importe quelle valeur, posons la égale à zéro. Le gop pointé obtenu par émondage est équivalent sémantiquement à g .

PROPRIETE 41.- Minimisation de la taille - Soient s1 et s2 deux sommets quelconques de G, et soient g1 et g2 les gops pointés de racines respectivement (s1,0) et (s2,0). Si il existe p (∈Z) tel que g2 ≡ translaté(g1,p) alors pour tout gop pointé g = G/(Rac,Clas) g ≡ g[s2<-(s1,p)]. Appelons cette opération "minimisation de la taille de G".

Exemple :



PROPRIETE 42.- Modification de pondérations - Soient s_0 un sommet de X tel que Lab(s_0) ∈ F et k un entier relatif quelconque de Z. Construisons g' = (X,Succ',Lab)/(Rac',Clas) de la manière suivante : . Rac' = (s_0,p +k) si Rac = (s_0,p)

= Rac sinon

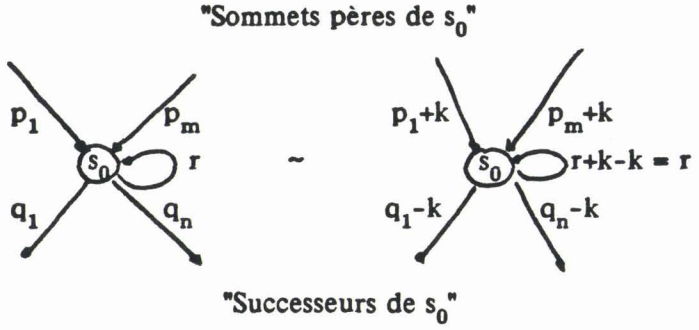
. si Succ(s,i) = (s',p') alors Succ'(s,i) = (s',p') si s = s' = s_0 ou s et s' <> de s_0

= (s',p'+k) si s' = s_0 et s <> s_0

= (s',p'-k) si s = s_0 et s' <> s_0

Ainsi construit g' est équivalent sémantiquement à g : g'~ g .

Intuitivement, en incrémentant de k toutes les pondérations des arcs entre un sommet père et s_0, puis en décrémentant de k toutes celles liant s_0 et un de ces successeurs, on ne modifie pas l'interprétation du gop pointé. (Les pondérations liant directement s_0 à lui-même ne sont pas modifiées par cette incrémentation et cette décrémentation .)



Démonstration : Comme $Clas_g = Clas_{g'}$, montrons que :

pour tout chemin m ∈ Dom(A(g,j)), A(g,j)(m) = A(g',j)(m).

Seules les pondérations ont été modifiées, les domaines sont inchangés et il nous suffit d'étudier les chemins aboutissant à des variables. Montrons donc que, ∀ le chemin m partant de la racine à une variable, la somme des pondérations est identique dans g et g'.

Notons Σ(S,m) la somme des pondérations rencontrées en partant du sommet s, par le chemin m dans g , et Σ'(s,m) dans g'.

Deux cas peuvent se présenter :

1) Le chemin m ne passe par par le sommet s_0, les pondérations n'ont pas été modifiées le long de ce chemin : Σ(Rac,m) = Σ'(Rac',m) (avec Rac = Rac' dans ce cas)

2) Le chemin m passe au moins une fois par s_0, on peut décomposer m en un nombre fini de sous-chemins $m_1, m_2, \ldots, m_n$ tels que

. $m = m_1.m_2. \ldots .m_n$ tel que ∀ i ∈ [1,n-1] ,le chemin $m_1.m_2. \ldots .m_i$ aboutit à s_0

. il n'existe pas de décompositions plus fines de m .

Etudions les pondérations sur chaque de ces sous chemins :

. $\Sigma(Rac,m_1) = \Sigma'(Rac,m_1) + k$ (si m_1 = nil , s_0 est la racine)

Il n'existe pas de sous-chemin strict de m passant par s_0 seule la pondération du dernier arc sur le chemin m a été modifiée c-à-d incrémentant de k.

. $\Sigma(s_0,m_i) = \Sigma'(s_0,m_i)$, ∀ i ∈ [2,n-1]

Car si le chemin est de longueur 1, c-à-d si l'arc emprunté va de s_0 à lui-même, la pondération n'a pas été modifiée.

De même si le chemin m_i est de longueur strictement supérieur à 1, deux pondérations ont été modifiées celle du premier arc, celui quittant s_0 et celle du dernier arc aboutissant à s_0 et donc la première est incrémenté de k, l'autre décrémenté .

D'où le résultat $\Sigma(s_0,m_i) = \Sigma'(s_0,m_i)$, ∀ i ∈ [2,n-1]

. $\Sigma(s_0,m_n) = \Sigma'(s_0,m_n) - k$

Sur le chemin m_n, seule la pondération du premier arc a été modifiée, décrémentant de k , d'où le résultat.

Or $\Sigma(Rac,m) = \Sigma(Rac,m_1) + \Sigma(s_0,m_2) + .. \Sigma(s_0,m_n)$

et $\Sigma'(Rac,m) = \Sigma'(Rac,m_1) + \Sigma'(s_0,m_2) + .. \Sigma'(s_0,m_n)$

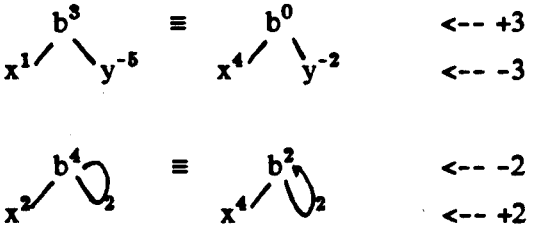
$= \Sigma(Rac,m_1) + k + \Sigma(s_0,m_2) + .. \Sigma(s_0,m_n) - k$

$= \Sigma(Rac,m)$

Donc pour tout $m \in Dom(A(g,j))$, $A(g,j)(m) = A(g',j)(m)$.

De plus g et g' ont la même congruence => $I(g) \sim I(g')$.

Exemples :



PROPRIETE 43.- Modification des pondérations modulo la congruence - Soient s un sommet du gop pointé g et un arc allant sur s de pondération p, alors on peut remplacer p par p ± Congruence de s sans modification de l'interprété de g :

$\forall k \in Z, \quad s^p \equiv s^{p+ k*Congruence(s)}$

Démonstration : Si s est de congruence c , pour tout sous gop pointé g1, de g, de racine (s,p) alors g1 ≡ translaté(g1,k*c) = gop de racine (s,p +k*c) .

On peut donc incrémenter n'importe quel pondération entre un sommet s' et s de k*c, $k \in Z$.

DEFINITION 44.- On dira que deux Gops pointés g, g' sont syntaxiquement équivalents :

$g \vdash g'$ si : . soit on peut passer de g à g' ou de g' à g en utilisant une des quatres propriétés

. soit il existe g" telle que $g \vdash g"$ et $g" \vdash g'$

PROPRIETE 45.- Equivalence Syntaxique <=> Equivalence Sémantique .

Deux gops pointés g et g' sont syntaxiquement équivalents ssi ils le sont sémantiquement :

$$g \vdash g' \iff g \sim g' .$$

Démonstration :

1) $g \vdash g' \Rightarrow g \sim g'$ .Voir Propriétés 40,41,42,43, ces transformations syntaxiques cons-

ervent l'interprété .

2) $g \sim g' \Rightarrow g \vdash g'$. Deux gops sont sémantiquement équivalents ssi l'algorithme d'unification fournit comme résultat une substitution vide, c-à-d aucune indirection et ni aucune nouvelle congruence sur les variables. Si on ne s'autorise dans l'unification que des indirections sur des labels de fonction, seule la propriété 41 est appliquée dans l'algorithme. D'où le résultat.

Chapitre 3

Etude de l'arrêt d'une régle récursive simple (sans test d'occurrence)

3-1) Introduction.

Après avoir présenté les GOS PONDERES, leur avoir associé des substitutions pondérées, une loi de composition interne, des opérations de manipulations syntaxiques donnant l'équivalence entre sémantique et syntaxe, et bien-sûr un algorithme d'unification, nous allons utiliser ces objets pour étudier les phénomènes de réécriture récursive.

Etudions le plus simple des programmes récursifs, le TANT QUE, en PROLOG II sous la forme suivante :

P(α) -> ;	" P(α) est vrai "
P(ß) -> P(ϑ);	" si P(ϑ) alors P(ß) "
P(T) ?	" Existe-t-il T tel que P(T) soit vrai "

où α, ß, ϑ, T sont des arbres rationnels construits sur F U V et P est un symbole de prédicat n'appartenant pas à F.

Ce programme peut engendrer une résolution infinie en temps, nous obligeant la plupart du temps à utiliser des artifices de programmation (le CUT ou le "/" en PROLOG), pour éviter de s'enfoncer dans une branche infinie de l'arbre de résolution.

Pour tenter d'apporter un élément de réponse, nous sommes amenés à nous poser la question suivante : **pour une règle récursive simple P(ß) -> P(ϑ), existe-t-il ou non des questions P(T) ou des données P(α) pour lesquelles le mécanisme de résolution ne pourrait répondre finiment ?**

3-2) Formalisation.

REECRITURE EN TETE : On peut réécrire P(T) sans ajout d'information par P(ß)->P(ϑ) ssi :

$$\exists\ \sigma, \text{ une substitution telle que } T = ß.\sigma\ .$$

T se réécrit en ϑ.σ, ce qui sera noté : $T = ß.\sigma \dashrightarrow T' = ϑ.\sigma$.
On peut faire disparaître le symbole de prédicat sachant que l'on réécrit en tête.

INFERENCE : Inférer P(T) avec la règle P(ß)->P(ϑ) correspond à une unification de P(T) avec la partie droite de la règle, P(ß), et à réécrire l'unifié :

Unification : $T \vee ß = ß.\sigma = T.\sigma$, s'il existe, avec ou sans test d'occurrence

Réécriture en tête : $T.\sigma \dashrightarrow ϑ.\sigma$

en vérifiant que T et ß ne partagent aucune variable (les variables de ß->ϑ sont muettes).

RENOMMAGE DES VARIABLES : **Pour éviter cette "capture de variables", nous renommerons celles de la règle après chaque réécriture, en les indiçant du numéro de dérivation.**

Pour la $i^{ème}$ dérivation, la règle appliquée sera $P(\beta_i) \rightarrow P(\gamma_i)$.

CHAINAGES AVANT OU ARRIERE : Pour résoudre notre programme TANT QUE, le mécanisme de résolution peut opérer de deux manières différentes. Etudions chacune d'elles, pour montrer qu'elles sont symétriques.

En chaînage avant, l'algorithme de résolution peut être écrit :

Données : α, β, γ, T
Début
  A := α;
  n := 1;
  Si (A et t unifiables) Alors l'unifié est solution
  Tant Que (A et γ_n unifiables) Faire
    Soit A V $\gamma_n = \gamma_n.\sigma_n$;
    A := $\beta_n.\sigma_n$; (A : nouveau fait)
    Si (A et t unifiables) Alors l'unifié est solution
    n := n+1;
  Fait
Fin

Soit $\sigma(n) = \sigma_1. \ldots .\sigma_n$ alors $\alpha.\sigma(n) = \gamma.\sigma(n) \dashrightarrow \beta.\sigma(n)$ en n pas par la régle $\gamma \rightarrow \beta$.

De même, en chaînage arrière, l'algorithme de résolution fournissant l'ensemble des solutions peut être écrit comme suit :

Données : α, β, γ, T
Début
  A := T;
  n := 1;
  Si (A et α unifiables) Alors l'unifié est solution
  Tant Que (A et β_n unifiables) Faire
    Soit A V $\beta_n = \beta_n.\sigma_n$;
    A := $\gamma_n.\sigma_n$; (A : nouveau but)
    SI (A et α unifiables) Alors Soit A V α = A.σ,
      $T.\sigma_1. \ldots .\sigma_n.\sigma$ est solution
    n:=n+1;
  Fait
Fin

Soit $\sigma(n) = \sigma_1. \ldots .\sigma_n$ alors $t.\sigma(n) = \beta.\sigma(n) \dashrightarrow \gamma.\sigma(n)$ en n pas par la régle $\beta \rightarrow \gamma$.

Ce mécanisme de recherche de toutes les solutions ne se terminera jamais si, on peut réécrire P(α), peut être réécrit une infinité de fois par la règle P(γ)->P(β) (chaînage avant), ou respectivement P(T) par P(β)->P(γ) (chaînage arrière).

3-3) Caractérisation des régles bouclantes.

La symétrie des deux algorithmes nous ramène à la question : Existe-t-il un arbre T se réécrivant une infinité de fois par la régle ß -> ϑ ?

Nous dirons si un tel arbre existe que la règle *boucle*. L'existence de T est liée à celle d'une suite infinie d'arbres (t_i) : $t_0 = T$ et $\forall i \in N$, $t_i \dashrightarrow t_{i+1}$ par $(ß_i \to ϑ_i)$

PROPRIETE 1.- Un arbre T peut se réécrire en T', sans ajout d'information, n fois par ß -> ϑ, ssi il existe une substitution σ(n) telle que :

- $T = ß_1.σ(n)$
- $ß_i.σ(n) = ϑ_{i-1}.σ(n) \quad \forall i \in [2,n]$
- $T' = ϑ_n.σ(n)$

PROPRIETE 2.- La règle ß->ϑ boucle ssi il existe une suite d'arbres $(t_i)_{i \in Z}$, telle que $t_i \dashrightarrow t_{i+1}$ par (ß -> ϑ). Les t_i sont tous équivalents (c-à-d égaux au nom des variables près).

Démonstration : Soit A l'ensemble des arbres se réécrivant une ∞ de fois par ß->ϑ, A est non vide puisque ß->ϑ boucle. Montrons que A possède un plus petit élément $t_{0,+\infty}$ que nous pouvons construire comme limite de la suite $(t_{0,n})$ des arbres se réécrivant n fois , c-à-d solution du système :

$$t_{0,n} = ß_1.σ(n) \quad \text{et} \quad ß_i.σ(n) = ϑ_{i-1}.σ(n) \quad \forall i \in [2,n]$$

La suite $(t_{0,n})$ est croissante et converge dans $M^{\infty}(F,VxZ)$, puisque $t_{0,n+1}$ se réécrit n+1 fois, donc aussi n fois : $t_{0,n} \leq t_{0,n+1}$. Soit $t_{0,+\infty}$,cette limite , c'est le plus élément de A.
A ce plus petit élément de A on peut associer la suite $(t_{i,+\infty})$ définie comme suit :
Mais $(t_{i,+\infty})$ est aussi une suite croissante :

- $t_{0,+\infty}$ se réécrit en $t_{1,+\infty}$ et $t_{1,+\infty} \in A$
- $t_{1,+\infty} \geq t_{0,+\infty}$ car $t_{0,+\infty}$ est le plus petit élément de A => $\exists\, σ$ telle que $t_{1,+\infty} = t_{0,+\infty}.σ$

On peut donc écrire : $t_{0,+\infty} \dashrightarrow t_{1,+\infty} = t_{0,+\infty}.σ \dashrightarrow t_{2,+\infty} = t_{0,+\infty}.σ.σ \dashrightarrow \ldots \dashrightarrow t_{n,+\infty} = t_{0,+\infty}.σ^n$
La suite d'arbres $(t_{0,+\infty})$ est croissante et converge dans $M^{\infty}(F,VxZ)$.
Notons $t_{+\infty,+\infty}$ cette limite qui vérifie : $t_{+\infty,+\infty} \dashrightarrow t'_{+\infty,+\infty} \equiv t_{+\infty,+\infty}$ par ß->ϑ ou ϑ->ß.
Cet arbre est "invariant" par ß->ϑ et par ϑ->ß.

CONSEQUENCE 3.- La règle ß->ϑ boucle ssi ϑ->ß boucle.

PROPRIETE 4.- Soient β^0 et γ^{-1} les Gops pointés obtenus en ajoutant à β et γ une pondération tête 0 et -1, les autres pondérations sur les arcs et les congruences de β et γ étant nulles alors : $A(\beta^0 \vee \gamma^{-1}, i)_{i \in Z}$ est la plus petite suite $(t_i)_{i \in Z}$ d'arbres se réécrivant une infinité de fois.

Démonstration : La démonstration repose sur la Z-Unification dans les gops pointés.
$I(\beta^0) = \{ (i,\beta_i) / i \in Z \}$ et $I(\gamma^{-1}) = \{ (i,\gamma_{i-1}) / i \in Z \} = \{ A(\gamma^{-1},i) / i \in Z \}$
Et nous savons que $I(\beta^0) \vee I(\gamma^{-1}) = I(\beta^0 \vee \gamma^{-1})$.
La suite $(t_i)_{i \in Z}$ est donc égale à $A(\beta^0 \vee \gamma^{-1},i)_{i \in Z}$.

L'unification que nous appliquons ici, est celle utilisée en Prolog II, c-à-d sans d'occurrence, dans les arbres rationnels. Nous étudierons le problème avec test d'occurrence dans le Chap 5 .

THEOREME 5 - Pour une règle récursive simple $P(\beta) \rightarrow P(\gamma)$, il existe des buts P(T) et des données $P(\alpha)$ pour lesquelles le mécanisme de résolution, classique avec recherche de toutes les solutions et sans test d'occurrence ne pourra répondre finiment ssi l'unifié de β et γ existe.

Démonstration : Montrons donc que la règle $\beta \rightarrow \gamma$ boucle ssi l'unifié de β et γ existe.

1) Si $\beta \vee \gamma$ existe, alors la donnée $P(\beta \vee \gamma)$ en chaînage avant ou le but $P(\beta \vee \gamma)$ en chaînage arrière engendre une résolution infinie en temps puisque $P(\beta \vee \gamma)$ se réécrit en un terme équivalent par $\beta \rightarrow \gamma$ et $\gamma \rightarrow \beta$.

2) Inversement nous savons que la règle boucle ssi l'unifié des Gops β^0 et γ^{-1} existe , ce qui est équivalent à l'existence de l'unifié de β et γ.

REMARQUES 6. - $A(\beta^0 \vee \gamma^{-1}, 0)$ est le plus petit invariant de $\beta \rightarrow \gamma$ ou $\gamma \rightarrow \beta$, c-à-d se réécrivant à un arbre équivalent. Il est, en général, irrationnel.

On peut facilement passer de $(\beta^0 \vee \gamma^{-1})$ à $(\beta \vee \gamma)$ en effaçant dans le gop pointé toutes les pondérations.

PROPRIETE 7.- Soit M le nombre de sommets du gop G sur lequel sont construits les gops pointés β^0 et β^{-1}. Les pondérations $ind_1,...,ind_k$ des indirections du graphe unifié vérifient :

. $k \leq M-1$

. $\forall i \in [1,k]$, $|ind_i| \leq 3^{i-1}$

$\forall v \in Var(G)$, $Clas(v) \leq 3^{M-1}$

Démonstration : Par récurrence sur le nombre de noeuds de G.

En effet $k \leq M-1$, puisque après chaque ajout d'une indirection, un noeud disparait dans le gop G, et comme il reste au moins un noeud après unification , au plus M-1 indirections possibles si M est le nombre de noeuds de G.

. Pour i=1, la première indirection est opérée entre les deux sommets de β et β avec la pondération -1 : $ind_1 = \pm 1 \in [-3^0,3^0]$

. Supposons que $\forall j \in [1,i]$, $ind_j \in [-3^{j-1},3^{j-1}]$ et montrons que $ind_{i+1} \in [-3^i,3^i]$.

Dans l'algorithme d'unification : Unification de leurs fils. La différence de pondérations de leurs sommets pères étant égale à l'une des indirections p_j précédemment ajoutées.

1) recherche de leurs représentants et soit ind et ind' les sommes des indirections rencontrées, dans cette recherche. Il est facile de voir que $|ind-ind'| \leq \sum |ind_j|$ $(j \leq i)$

2) il faut donc unifier les sommets fils de s et s' avec des pondérations égales à p+ind et p'+ind' (puisque les pondérations du Gop sont toutes nulles).

$\Rightarrow$ $|p+ind| - |p'+ind'| \leq |p-p'| + |ind-ind'| \leq ind_i + \sum ind_j$ $(j \leq i)$

3) si ils ont même représentant, alors on modifie éventuellement la congruence:

la nouvelle congruence : $\leq ind_i + \sum ind_j$ $(j \leq i)$

. si leurs représentants sont différents, on distribue éventuellement certaines congruences,

et on indirige : $|ind_{i+1}| \leq |ind_k| + |ind_1| + |ind_2| + .. + |ind_i|$

$\leq 3^{i-1} + 3^0 + 3^1 + .. + 3^{i-1} \leq 3^i$

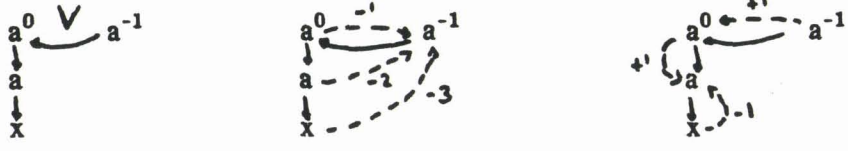
$\forall v \in Var(G)$, $0 \leq Clas(v) \leq 3^{M-1}$ car comme nous venons de le voir les sommets unifiés dans l'algorithme, ont une différence de pondérations d'au plus égal à $3^{M-2} + (3^{M-2} + 3^{M-3} + .. + 3^0) \leq 3^{M-1}$

$\Rightarrow$ $\forall v \in Var(G)$, $Clas(v) \leq 3^{M-1}$.

CONJECTURE 8.- : Avec un algorithme d'unification optimisé (ordre des sommets fils unifiés et sens des indirections), si ß et ϑ contiennent N variables, les pondérations ind_1, ... ,ind_k des indirections portant sur les sommets étiquetés d'une variable vérifient : $\forall i \in [1,k]$, $-2^{i-1} \leq ind_k \leq 2^{i-1}$

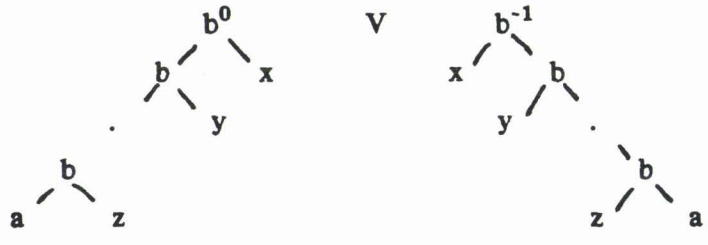
De plus $\forall v \in Var(G)$, $0 \leq Clas(v) \leq 2^{N-1}$.

Exemples :



Optimisé

Un cas extrême est :



Cette conjecture aurait une influence directe sur les résultats démontrés dans le chapitre 5, elle maximise le nombre nécessaire de réécritures dans le TQ Prolog, avant obtention des principaux critères de décidabilité (existence d'une solution, nombre fini de chemins de preuve, ...).

Chapitre 4

Définitions et propriétés des IGOPS ou interprétation finie des Gops

Pour pouvoir pleinement caractériser les contraintes portant sur les variables indicées pour une application finie d'une régle récursive , nous allons introduire un nouvel objet le Igop , qui nous permettra d'étendre les possibilités des Gops et de leur interprétation .

4-1) Préliminaires. Congruences d'indices.

DEFINITION 1.- Restriction ou extension d'intervalles à une constante près.

Soient P un intervalle quelconque de Z et a, b deux entiers relatifs, on note P + [a,b] , l'intervalle P dont les bornes ont été incrémentées, respectivement de a et b .

[A,B] + [a,b] = [A+a,B+b] ,]+∞,B[+ [a,b] =]+∞,B+b]

De même pour P - [a,b] , les bornes de P sont décrémentées.

(si après cette manipulation, la borne inférieure n'est pas plus petite que la borne supérieure, on considérera alors que c'est l'ensemble vide).

DEFINITION 2.- Interprétation finie d'une congruence d'indices.

Soit un intervalle P quelconque de Z, Clas une congruence d'indices, soit s un sommet quelconque de second label x et de congruence Clas(s) :

$\forall$ k et k' $\in$ P , $x_k = x_{k'}$ ssi (k-k') modulo Clas(s) = 0.

On pourra toujours se choisir un représentant canonique d'un sommet pondéré (s,p) pour la congruence Clas, représentant dont la pondération appartiendra à un certain intervalle P d'interprétation, il sera noté : $(s,p) \bmod_P \text{Clas}(s) = (s_0,p_0)$ avec $p_0 \in P$ si $p \in P$

$= (s,p)$ sinon

REMARQUE.- En règle générale , l'unifié de deux congruences n'est pas une congruence.

Les congruences 3 et 5 sur P = [0,5] :

congruence 3 => $(x_0 = x_3)$, $(x_1 = x_4)$ et $(x_2 = x_5)$ et congruence 5 => $(x_0 = x_5)$

Leur unification donne les égalités suivantes : $(x_0 = x_2 = x_3 = x_5)$ et $(x_1 = x_4)$.

PROPRIETE 3.- L'unifié de deux congruences c et c' sur un intervalle P de taille supérieure à c+c' est la congruence Pgcd(c,c') sur P.

Démonstration : (on suppose que $c \geq c'$)

Soit une variable x de congruences c et c' sur P : on peut noter ces congruences sous la forme :

$\forall i \in P+[0,-c]$, $x_i = x_{i+c}$ et $\forall j \in P+[0,-c']$, $x_j = x_{j+c'}$ (1)

=> $\forall i \in P+[0,-c] \cap P+[0,-c'] = P+[0,-c]$, $x_{i+c'} = x_{i+c}$

c-à-d $\forall j \in P+[c',c'-c]$, $x_j = x_{j+c-c'}$ (2)

On peut écrire les équations (1) sous la forme :

$\forall i \in P+[0,-c]$, $x_i = x_{i+c}$ et $\forall j \in P+[c'-c,-c]$, $x_{j+c-c'} = x_{j+c}$ (1)

c-à-d $\forall j \in P+[0,-c] \cap P+[c'-c,-c] = P+[0,-c]$, $x_j = x_{j+c-c'}$ (3)

(2) et (3) => $\forall j \in P+[c',c-c'] \cup P+[0,-c]$, $x_j = x_{j+c-c'}$

c-à-d si $Card(P) \geq c+c'$, $\forall j \in P+[0,c-c']$, $x_j = x_{j+c-c'}$

On en déduit alors une congruence de $|c-c'|$ sur P . En appliquant l'algorithme d'Euclide, on obtient donc : (Congruence Pgcd(c,c') sur P).

CORROLAIRE 4.- L'unifié d'une congruence c sur P et d'une congruence c' sur P' est la congruence Pgcd(c,c') sur P ∪ P' si c et c' sont non nulles et si P ∩ P' contient au moins c+c' éléments.

Démonstration : (on suppose que $c \geq c'$)

En effet, si P ∩ P' contient au moins c+c' éléments, on peut appliquer la propriété ci-dessus :

Congruence Pgcd(c,c') sur P ∩ P'.

On peut alors étendre cette nouvelle congruence Pgcd(c,c') sur P et P' en utilisant les congruences c et c' respectivement sur P et P' .

Si l'on note c" = Pgcd(c,c'), montrons que $\forall i \in P-[0,-c"]$, $x_i = x_{i+c"}$.

Comme c est non nul et P ∩ P' contient au moins c+c' éléments, il existe i_0 tel que :

$i_0 \equiv i \pmod{c}$ avec i_0 et $i_0+c' \in P \cap P'$ => $x_i = x_{i0} = x_{i0+c"} = x_{i+c"}$

=> $\forall i \in P-[0,c"]$, $x_i = x_{i0}$

De la même façon pour c' sur l'intervalle P'.

D'où le résultat : x et y sont de congruence Pgcd(c,c') sur P U P' si $Card(P \cap P') \geq c+c'$.

4-2) Les IGOPS. Définition et Interprétations.

4-2-1) DEFINITION SYNTAXIQUE D'UN IGOP.

DEFINITION 5.- On appelle *Igraphe ordonné pondéré (IGOP)* , construit sur F et V, la donnée du quadruplet : (X,Lab,Succ,Clas) où

- *X* est un ensemble fini de sommets.

- *Lab* est une application de X dans F ∪ V x V qui à chaque sommet associent deux labels (notés Lab1(s) ∈ FUV et Lab2(s) ∈ V) tels que :

. Lab1(s) = Lab2(s) si Lab1(s) ∈ V

. Lab2 est une application injective sur V. Le second label caractérise le sommet s de X, on confondra souvent le sommet et son second label.

- *Succ* est une fonction partielle de XxN dans XxZ : Succ(s,i) = (s',p'), le ième successeur de s est le sommet s' avec la pondération p'. De plus si on appelle nsucc(s) le nombre de successeurs du sommet s, on a alors nsucc(s)=π(Lab(s)).

- *Clas* est une congruence d'indices, application de X dans N

NOTATIONS : On appelle pondération père d'un sommet s, toute pondération d'un arc allant sur s , de même une pondération fils de s entre s et l'un de ses sommets successeurs.

On notera : . $Pond_G(s)$, le plus petit intervalle $[-a_s,b_s]$ contenant toutes les pondérations pères de s dans G (a_s, b_s ∈ N)

. MaxClas(G), le maximum des congruences des sommets de G

. MaxPond(G), le maximum des pondérations en valeur absolue.

4-2-2) INTERPRETATION D'UN IGOP COMME SYSTEME D'EQUATIONS.

Nous allons donner une interprétation des IGOPS comme système d'équations ou comme une substitution pondérée interprétée, non plus sur Z, mais sur un intervalle P quelconque de Z.

DEFINITION 6.- Interprétation d'un IGOP G sur l'intervalle P : $S_P(G)$.

Soit G = (X,Lab,Succ,Clas), nous noterons $S_P(G)$, le système composé de deux types d'équations, pour tout i ∈ P et s ∈ X, posons Lab1(s) = u et Lab2(s) = x :

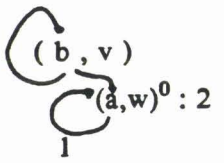
1- $(x_i = t)$ **si** $u \in F$

avec $t = u$ **si** $u \in F_0$

$t = u(Lab2(s_1)_{q1}, \ldots, Lab2(s_n)_{qn})$ **si** $u \in F_n$ **et** $Succ(s,j) = (s_j,p_j)$, $q_j = p_j+i$

2- $(x_i = x_{i \bmod Clas(s)})$ **(équations congruentes).**

Exemple :

$G =$ 

$S_P(G)$ est composée des équations suivantes :

$\forall i \in P, (v_i = b(v_{i+1}, w_i)), (w_i = a(w_{i-1}))$

et $(w_i = w_{i \bmod 2})$

PROPRIETES 7.-

1) A tout système $S_P(G)$, on peut associer un système équivalent dans lequel il n'existe pas deux équations portant en partie droite sur la même variable indicée.

2) On peut construire la substitution $\sigma_P(G)$ associée à $S_P(G)$:

$\sigma_P(G) = S_P(G).S_P(G).S_P(G). \ldots = S_P(G)^{\omega}$

3) Si P est fini, alors $\sigma_P(G)$ est rationnelle, c-à-d composée d'équations de la forme :

$(x_i = t)$ où t est un arbre rationnel.

4) P est inclus dans P' => $S_P(G)$ est inclus dans $S_{P'}(G)$ => $\sigma_P(G) \leq \sigma_{P'}(G)$.

Démonstration :

1) $S_P(G)$ peut contenir deux équations par variable indicée, une par rapport à un arbre non réduit à une variable, l'autre par rapport à une variable de même symbole, mais on peut toujours construire un système équivalent et déterministe (au plus une équation par variable indicée).

Une variable x peut être doublement définie dans $S_P(G)$ ssi elle étiquette un sommet s tel que Lab1(s) = u ∈ F, Lab2(s) = x et Clas(s) est non nulle :

$\forall i \in P , (x_i = u(...)) \text{ et } x_i = x_{i \bmod Clas(s)} \in S_P(G)$

La congruence de la variable x entraîne la même sur toutes les variables de u(...) sur un intervalle égal à P à un décalage près (pondération de l'arc). Ces égalités entre variables indicées de même symbole ne sont pas toujours explicitement décrites dans les équations congruentes.

Ces égalités sont les seules équations déductibles de ces doubles contraintes. En les rajoutant donc explicitement, et en ne conservant des équations non congruentes que celles relatives à un représentant de ces variables indicées égales, on déterminise le système, c-à-d qu'il ne contient plus qu'une équation par variable indicée en partie droite.

2) La forme des équations congruentes : $x_j = x_{j0}$, nous évite tout cycle dans l'expression des contraintes d'égalités entre variable indicées $\Rightarrow \sigma_P(G) = S_P(G)^\omega$

3) Si P est fini, $S_P(G)$ contient un nombre fini d'équations, la substitution associée est donc rationnelle

4) Si P est inclus dans P', le système $S_P(G)$ est inclus syntaxiquement dans $S_{P'}(G)$, c-à-d que toutes les équations de $S_P(G)$ apparaissent sous la même forme dans $S_{P'}(G)$.
Une conséquence immédiate est que : $\sigma_P(G) \leq \sigma_{P'}(G)$.

4-2-3) Définition et interprétation d'un IGOP pointé.

DEFINITION 8.- On appellera un *IGOP pointé* la donnée d'un IGop et d'une racine pondérée :

G/Rac **où Rac est un élément de XxZ**

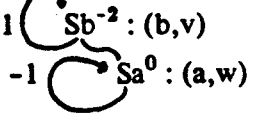
Exemple : g = G /(Sb,-2) avec G = (X,Lab,Succ,Clas) , F = { a, b } et V = { v, w }

. X = {Sa,Sb}

. Lab(Sb) = (b,v) , Lab(Sa) = (a,w)

. Succ(Sb,1) = (Sb,1) , Succ(Sb,2) = (Sa,0) , Succ(Sa,1) = (Sa,-1)

. Clas(Sb) = 0 , Clas(Sa) = 0



DEFINITION 9.- ***Interprétation d'un Igop pointé dans les ensembles ordonnés d'arbres.***

Soient g un Igop pointé construit sur G et de racine (s,p) et E,P deux intervalles de Z, on définit l'interprété de g sur E et P par : $I_{E,P}(g) = \{ (k,A_P(g,k)) / \forall\ k \in E \}$

posons $Lab2(s) = x$, $A_P(g,k) = t$ **si il existe** $(x_{p+k} = t) \in \sigma_P(G)$

$= x_{p+k}$ **sinon**

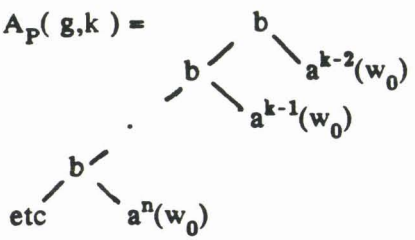
c-à-d $I_{E,P}(g) = \{ (i,x_{p+i}) / \forall\ i \in E \}.\ \sigma_P(G)$

E sera appelé l'intervalle des pondérations d'entrée et P l'intervalle d'interprétation de g.

Exemple : $g = G/(Sb,-2)$, pour $E = P = [1,+\infty[$: $I_{E,P}(g) = \{ (k,A_P(g,k) / \forall\ k \in E \}$

$\forall\ k \in [1,2]$, $A_P(g,k) = v_{k-2}$

$\forall\ k \in [3,+\infty[$, $A_P(g,k) =$ (arbre irrationnel)



REMARQUE 10 - Nous présenterons une interprétation équivalente, mais beaucoup plus proche de celle des Gops à la fin de ce chapitre, dans le cas de Igops particuliers, appelés Igops parfaits.

4-3) Relation de préordre.

DEFINITION 11.- La relation d'ordre sur les systèmes d'équations est naturellement :

$S \leq S'$ ssi toute équation de S est une conséquence des équations de S'.

RAPPEL : La relation de préordre que l'on utilisera sur les ensembles ordonnés T et T', est :

$T \leq T'$ si il existe une substitution σ telle que $T . \sigma$ est inclus dans T'

(la loi de composition utilisée est ensembliste : $\{ (k,t_k) \} . \sigma = \{ (k,t_k.\sigma) \}$)

PROPRIETE FONDAMENTALE 12.- Soient g un IGop pointé et E,E',P,P' des intervalles de Z, si E est inclus dans E' et si P est inclus dans P', il existe une substitution σ ne dépendant que de P et P' telle que : $I_{E,P}(g).\sigma$ est inclus dans $I_{E',P'}(g)$

Démonstration : De façon immédiate, nous savons que :

si P est inclus dans P' alors $\sigma_P(G) \leq \sigma_{P'}(G)$

Il existe donc σ tel que $\sigma_P(G).\sigma = \sigma_{P'}(G)$, σ est bien-sûr indépendante de tout sommet de G, la définition même de l'interprété d'un Igop pointé nous assure la validité du théorème :

$\forall$ g construit sur G et $k \in E$, $A_P(g,k).\sigma = A_{P'}(g,k)$.

=> si P est inclus dans P' , $I_{E,P}(g).\sigma = I_{E,P'}(g)$

donc si E est inclus dans E' , $I_{E,P}(g).\sigma$ est inclus dans $I_{E',P'}(g)$

DEFINITION 13.- Soient s et s' deux sommets différents mais de même congruence, on note G[s' <- (s,p)], le Igop G dans lequel le sommet s' a été remplacé par le sommet s avec incrémentation de p des pondérations :

posons $G = (X,Lab,Succ,Clas)$, $G[s' <-(s,p)] = (X,Lab,Succ',Clas)$, $\forall s_1 \in X$, $Succ(s_1,i) = (s_2,p_2)$

si $s_2 = s'$ alors $Succ'(s_1,i) = (s,p+p_2)$

sinon $Succ'(s_1,i) = (s_2,p_2)$ (inchangé)

DEFINITION 14.- Soit s un sommet de G, on note G[s <- (s,p)] le Igop dont la congruence de s, Clas(s), est remplacée par Pgcd(Clas(s),p) : $G[s <-(s,p)] = (X,Lab,Succ,Clas')$ avec $Clas'(s) = Pgcd(Clas(s),p)$ et $\forall s' <> s$, $Clas'(s') = Clas(s')$.

PROPRIETE 15.- Soit G' = G[s <- (s,p)] , $\sigma_P(G) \leq \sigma_P(G')$.

Démonstration : En effet , nous avons ajouté dans $S_P(G')$ de nouvelles équations congruentes, il y a donc inclusion syntaxique => $\sigma_P(G) \leq \sigma_P(G')$.

DEFINITION 16.- Un Igop est dit *homogène* si la fonction Clas est compatible avec la fonction Succ, c-à-d tout sommet est au moins de la congruence de ses pères :

$\forall s_p$, sommet père de s, $Clas(s_p)$ est un multiple (éventuellement nul) de Clas(s).

PROPRIETE 17.- Pour tout Igop G interprété sur P, on peut construire un Igop G' homogène et un intervalle P_m défini à une constante de bornes près par rapport à P tel que :

$$S_{Pm}(G') \leq S_P(G) \leq S_P(G') ,$$

G' correspondant au Igop G dont toute les congruences sont distribuées sur les sommets fils.

Démonstration : Soit s un sommet de G, de congruence non nulle, et soit s_i un sommet fils quelconque de s, montrons qu'il existe un intervalle P_m défini à une constante près par rapport à P, tel que :

$S_{Pm}(G') \leq S_P(G) \leq S_P(G')$ où $G' = G[s_i <-(s_i,Clas(s)]$

Posons $Lab_1(s) = u$, $Lab_2(s) = x$ et $Lab_2(s_i) = y$. Nous savons,ici, que $u \in F$, c-à-d dans $S_P(G)$, la variable x est représentée par : $\forall k \in P , x_k = u(.. , y_{k+pi} , ..)$ et $x_k = x_{k0}$.

On peut donc déduire que y est de congruence Clas(s) sur $P + [p_i,p_i]$.

Si $Clas(s_i)$ est non nul et si $P \cap P+[p_i,p_i]$ contient au moins $Clas(s)+Clas(q_i)$ éléments, on peut en déduire que : s_i est de congruence $Pgcd(Clas(s),Clas(s_i))$ sur $P \cup P+[p_i,p_i]$, donc sur P.

Si $Clas(s_i)$ est nul : s_i est de congruence Clas(s) sur $P+[p_i,p_i]$.

Posons donc : $P_m = P \cap P+[p_i,p_i]$ si $Clas(s_i) = 0$

$= P$ sinon.

Dans chacun des cas, $S_{Pm}(G)$ est inclus syntaxiquement dans $S_P(G)$ car P_m est inclus dans P.

Or les équations congruentes : $\forall i \in P_m$, $y_i = y_{i \bmod c}$ avec $c = Pgcd(Clas(s),Clas(s_i)$ peuvent être ajoutées, puisque elles sont déductibles des équations congruentes sur les x_i, si on pose donc :

$G' = G[s_i <-(s_i,Clas(s)]$ alors $S_{Pm}(G') \leq S_P(G)$,

comme de plus $S_P(G)$ est inclus syntaxiquement dans $S_P(G')$ => $S_{Pm}(G') \leq S_P(G) \leq S_P(G')$

Cette descente d'une congruence d'un sommet sur l'un de ses fils peut être itérée au tant de fois qu'il le faut, pour rendre de Igop G homogène. Or le nombre d'itérations est limitée, tout sommet ne peut modifier qu'une fois les congruences de tous les sommets qui lui sont accessibles. Il existe des constantes positives a, b, min dépendant directement de G de ses pondérations et congruences tel que , le Igop G' homogène vérifie l'inéquation :

$$S_{P+[a,-b]}(G') \leq S_P(G) \leq S_P(G') \quad \text{si Card(P)} \geq \text{min}$$

On peut négliger les contraintes de taille minimale de l'intervalle P en majorant les constantes a et b de telle sorte que si Card(P) < min, alors P_m soit vide (borne inf > borne sup), car dans ce cas $S_{Pm}(G)$ est vide aussi.

4-4) Unification de l'interprété de deux Igops pointés.

CADRE D'ETUDE : Notre propos n'est pas ici de donner un algorithme d'unification de deux Igops quelconques, mais plutôt dans le cadre de l'étude des règles récursives simples de montrer comment l'unifié de deux Igops peut être encadré par un Igop commun interprété sur des intervalles différents. Dans le cas général, d'ailleurs, l'unifié de deux Igops pointés n'est pas un Igop pointé, l'unification n'est donc pas ici une opération interne.

Dans le cadre de notre étude, nous supposerons que les deux Igops à unifier ne différent que par leur racine pondérée, c-à-d :

. ils sont interprétés sur le même intervalle P (= E)

. ils partagent le même Igop G =(X,Lab,Succ,Clas)

4-4-1) DEFINITION DE L'UNIFIE DE DEUX IGOPS POINTES

DEFINITION 18.- Deux Igops pointés sont dit unifiables sur E et P si leurs interprétés $I_{E,P}(g)$ et $I_{E,P}(g')$ le sont : $\exists\ \sigma$ telle que $I_{E,P}(G).\sigma = I_{E,P}(g').\sigma$.
L'unifié de g et g' sur E et P, est le plus petit ensemble T ordonné d'arbres vérifiant :

$$T = I_{E,P}(g).\sigma = I_{E,P}(g').\sigma .$$

PROPRIETE 19.- Soient E, E', P, P' des intervalles de Z tels que E est inclus dans E' et P inclus dans P' alors : $I_{E,P}(g)\ V\ I_{E,P}(g') \leq I_{E',P'}(g)\ V\ I_{E',P'}(g')$

Démonstration :Elle découle directement de la propriété 12. Puisque il existe σ indépendant de g et g' telle que : $I_{E,P}(g).\sigma = I_{E',P'}(g)$ et $I_{E,P}(g').\sigma = I_{E',P'}(g')$

PROPRIETE 20.- L'unification de 2 Igops pointés g et g' construit sur le même Igop G sur l'intervalle P : $I_P(g)\ V\ I_P(g')$, peut être représenté en terme de systèmes d'équations, sous la forme de l'unification des deux systèmes d'équations suivants :

$$S_P(G)\ V\ \{ (y_{i+p'} = x_{i+p}) / \forall i \in P \}$$

avec (s,p) et (s',p') sommets racine de g et g' et Lab2(s) = x et Lab2(s') = y.
Posons $S = S_P(G)\ V\ \{ (x_{i+p} = y_{i+p'} / \forall i \in P \}$ alors $I_P(g)\ V\ I_P(g') = \{ (i,x_{i+p}) / \forall i \in P \} . S^{\infty}$

4-4-2) MANIPULATIONS SYNTAXIQUES

Nous allons ici décrire , comme dans l'unification des Gops, les manipulations possibles pour l'unification de deux Igops pointés, sachant que l'on cherche un encadrement et non la solution exacte. Nous noterons g et g' , ces Igops pointés construits sur G, de racines (s,p) et (s',p') avec Lab2(s) = x et Lab2(s') = y.

LEMME 21.- Soient G' = G[s <- (s,p-p')] avec p ≥ p' et S = $S_P(G)$ ∨ { ($x_{i+p} = x_{i+p'}$ / ∀ i ∈ P }. Pour P assez grand, on peut construire deux intervalles P_m et P_M définis à une constante de bornes prés par rapport à P, tel que : $S_{Pm}(G') \leq S \leq S_{PM}(G')$

où . $P_m = P$ si Clas(x) non nul et Card(P) ≥ Clas(x)+|p|+(p-p')

$= P \cap P+[p',p]$ si Clas(s) nul

. $P_M = P \cup P+[p',p]$

Démonstration :

a) Clas(s) nul. Le second système d'équations exprime une congruence (p-p') de la variable x sur l'intervalle P+[p',p] : $\forall j \in P+[p',p]$, $x_j = x_{j \bmod (p-p')}$

On minimise les contraintes contenues dans les deux systèmes si on choisit Pm tel que :

P_m inclus dans P et P_m inclus dans P+[p',p] c-à-d $P_m = P \cap P+[p',p]$

De même on maximise les contraintes si l'on choisit P_M tel que :

P inclus dans P_M et P+[p',p] inclus dans P_M c-à-d $P_M = P \cup P+[p',p]$.

b) Clas(s) non nul. Nous savons que x est de congruence Clas(s) sur P. Le second système exprime une congruence (p-p') sur P+[p',p]. En appliquant le résultat sur l'unification de congruence, on peut donc dire que x est de congruence Pgcd(Clas(s),p-p') sur P ∪ P+[p',p] si P ∩ P+[p',p] contienne au moins Clas(s)+(p-p') éléments.

Si p et p' ≤ 0 $P \cap P+[p',p] = P+[0,p]$ donc de taille ≥ Card(P) - |p|

p' ≤ 0 et p ≥ 0 $P \cap P+[p',p] = P$ . .. = Card(P)

p et p' ≥ 0 $P \cap P+[p',p] = P+[p',0]$ . .. ≥ Card(P) - |p'| ≥ Card(P) - |p|

=> Card (P ∩ P+[p',p]) ≥ Card(P) - |p|

La condition sur l'intervalle P est donc : Card(P) ≥ Clas(s) + |p-p'| + |p|

L'unification des deux congruences est la congruence Pgcd(Clas(s),p-p') sur P ∪ P+[p',p].

Donc on minore si l'on garde P comme intervalle d'interprétation ($P_m = P$) et on majore en choisissant $P_M = P \cup P+[p',p]$.

Nous pouvons donc écrire l'algorithme de calcul de P_m et P_M pour G[(s <- (s,p-p')]

. a_m , b_m $\in$ N tels que Pm = P +[a_m,-b_m]

. a_M , b_M $\in$ N tels que PM = P +[-a_M,b_M]

. card_min : taille minimale de P.

Procédure congruence (s,p,p',a_m,b_m,a_M,b_M,card_min);

Début

| *Si p = p' Ou (p-p') multiple de Clas(s)*
| | *Alors % Cas trivial %* a_m:=0; b_m:=0; a_M:=0; b_M:=0; card_min:=0;
| | *Sinon*
| | *q := Sup(p,p'); q' := Inf(p,p'); % q > q' %*
| | *% Nouvelle congruence de s : Pgcd(Clas(s),q-q') %*
| | a_M:= -Inf(q',0); b_M:= Sup(0,q);
| | *Si Clas(s) est non nul*
| | | *Alors* a_m:=0; b_m:=0; card_min := Clas(s) + |q| + (q-q');
| | | *Sinon* a_m:= Sup(q',0); b_m:= -Inf(0,q); card_min:= 0;
| | *Fsi;*
| *Fsi;*
| G[s <-(s,(p-p'))] ;

Fin;

Exemple : $(u,u)^0$ V $(u,u)^{-3}$ sur l'intervalle P (u de congruence nulle) => $S_P(G) = \emptyset$

L'unification des 2 Igops s'écrit : $S_P(G)$ V { $u_{i-3} = u_i$ / $\forall$ i $\in$ P }.

Application du Lemme 1 : G' = G[u <- (u,3)] c-à-d Clas(u) devient 3.

=> $S_{Pm}(G') \leq \{ u_{i-3} = u_i / \forall i \in P \} \leq S_{PM}(G')$.

avec $P_m = P \cap P+[-3,0] = P$ et $P_M = P \cup P+[-3,0] = P+[-3,0]$

LEMME 22.- G' = G[s' <-(s,p-p')] , x <> y et Clas(s) = Clas(s').
Soit S = $S_P(G)$ V { ($x_{i+p} = y_{i+p'}$ / ∀ i ∈ P } , on peut construire deux intervalles P_m et P_M définis à une constante de bornes prés par rapport à P, tel que :

$$S_{Pm}(G') \text{ V } \{(y_i = x_{i+p-p'})/ \forall i \in P_m \} \leq S \leq S_{PM}(G') \text{ V } \{(y_i = x_{i+p-p'})/ \forall i \in P_M \}.$$

avec $P_m = (P + [p'-a_y,p'-b_y]) \cap P$

$P_M = (P + [-p'+a_y,-p'+b_y]) \cup P$ avec $[a_y,b_y] = Pond_G(y)$

Rappel : $Pond_G(y)$ désigne le plus petit intervalle de Z contenant zéro et toutes les pondérations père de s' dans G .

Démonstration : On ne change rien à l'unification des deux systèmes si l'on remplace dans certaines équations , les variables y_k apparaissant en partie droite par $x_{k+p-p'}$, si k ∈ P+[p',p'], c-à-d conformément aux équations du second système.
Mais tous les indices de cette variable y n'y appartiennent pas en général. On ne peut donc substituer globalement sans changer l'unifié.

Posons le problème de façon plus générale, soient P et P' deux intervalles de Z, étudions le système $S_P(G) \cup \{ (y_i = x_{i+p-p'}) / \forall i \in P' \}$

Cette substitution peut être faite globalement si toutes les pondérations de la variable y dans le 1er système appartiennent à P'. Or ces pondérations appartiennent à $P+Pond_G(y)$: $P+Pond_G(y)$ est inclus dans P'.

1) On minimise les contraintes en diminuant l'intervalle d'interprétation P de G , soit P_m cet intervalle défini à une constante de bornes près sur P tel que :

P_m inclus dans P et dans $(P + [p',p']) - Pond_G(y)$

c-à-d $P_m = (P + [p'+a_y,p'-b_y]) \cap P$ où $[-a_y,b_y] = Pond_G(y)$

. $S_{Pm}(G) \leq S_P(G)$ car P_m inclus dans P

. $\{(y_i=x_{i+p-p'})/ \forall i \in P_m+Pond_G(y) \} \leq \{(y_{i+p'}=x_{i+p})/ \forall i \in P\}$

car $P_m+Pond_G(y)$ inclus dans $(P + [p',p'])$

. $S \geq S_{Pm}(G)$ V $\{(y_i=x_{i+p-p'})/ \forall i \in P_m+Pond_G(y) \}$

$= S_{Pm}(G')$ V $\{(y_i=x_{i+p-p'})/ \forall i \in P_m+Pond_G(y) \}$

. $S \geq S_{Pm}(G')$ V $\{(y_i=x_{i+p-p'})/ \forall i \in P_m \}$

car P_m inclus dans $P_m+Pond_G(y)$

2) On maximise les contraintes en choisissant PM défini à une constante près sur P tel que:

P inclus dans PM et P+[p',p'] -$Pond_G(s')$ inclus dans PM

c-à-d PM = (P + [-p'-$a_{s'}$,-p'+$b_{s'}$]) ∪ P. $S_{PM}(G) \geq S_P(G)$ car P inclus dans P_M

. $\{(y_i=x_{i+p-p'})/ \forall i \in P_M\} \geq \{(y_{i+p'}=x_{i+p})/ \forall i \in P\}$

car P est inclus dans P_M-[p',p']

. $S \leq S_P(G)$ V $\{(y_i=x_{i+p-p'})/ \forall i \in P_M\} = S_P(G')$ V $\{(y_i=x_{i+p-p'})/ \forall i \in P_M\}$

car P+[p',p']-$Pond_G(s')$ inclus dans PM

=> $S \leq S_{PM}(G')$ V $\{(y_i=x_{i+p-p'})/ \forall i \in P_M\}$

L'algorithme de calcul des intervalles P_m et P_M s'écrit donc :

. a_m , $b_m \in N$ tels que $Pm = P + [a_m,-b_m]$

. a_M , $b_M \in N$ tels que $PM = P + [-a_M,b_M]$

Procédure substitution-même-congr (s,p,s',p',a_m,b_m,a_M,b_M)

Début

| % $Pond_G(s') = [-a_{s'},b_{s'}]$ %

| $a_m := Sup(p'+a_{s'},0)$; $b_m := -Inf(p'-b_{s'},0)$;

| $a_M := -Inf(p'+a_{s'},0)$; $b_M := Sup(p'-b_{s'},0)$;

| $G[s' <-(s,(p-p'))]$

Fin;

Exemple : $(Entier,u)^0$ V $(x,x)^{-1}$ sur l'intervalle P (congruences nulles)

$(x,x)^0$

$S_P(G) = \{ u_i = Entier(x_i) / \forall i \in P \}$

L'unification des deux Igops : $S_P(G)$ V $\{ x_{i-1} = u_i / \forall i \in P \}$

Application du Lemme 2 : G' = G[x <- (u,1)] :

on se ramène donc à : $S_P(G') = \{ u_i = Entier(u_{i+1}) / \forall i \in P. \}$ V $\{ x_i = u_{i+1} / \forall i \in P. \}$

avec $P_m = P + [0,-1]$ et $P_M = P + [0,1]$ car p' = -1 et $Pond_G(s') = [0,0]$

LEMME 23 : G' = G[s' <- (s,p-p')], s et s' distincts.

Soit $S = S_P(G)$ V { $(x_{i+p} = y_{i+p'}$ / ∀ i ∈ P }, pour P assez grand, on peut construire deux intervalles Pm et PM définis à une constante de bornes prés par rapport à P, tel que :

$$S_{Pm}(G') \text{ V } \{(y_i=x_{i+p-p'})/ \forall i \in Pm \} \leq S \leq S_P(G') \text{ V } \{(y_i=x_{i+p-p'})/ \forall i \in PM \}$$

avec G'= G [s'<-(s,p-p')] , Lab1(s) = x et Lab1(s') = y

Démonstration :

1) Harmonisation des congruences de s et s'. Posons Clas(s) = c et Clas(s') = c'. Etudions ici l'unification des congruences de x et y et du second système :

$$\{ x_i = x_{i0} \pmod c)/ \forall i \in P \} \text{ V } \{ y_i = y_{iO} \pmod{c'})/ \forall i \in P\} \text{ V } \{x_{i+p} = y_{i+p'} / \forall i \in P \}$$

Montrons qu'il existe Pm tel que x et y sont de congruences Pgcd(c,c') sur Pm :

y est de congruence c sur P+[p'-p,p'-p] ∩ P+[p',p'].

$\forall i \in P+[0,-c]$, $x_i = x_{i+c}$ (congruence sur x)

$\forall i \in P+[p,p]$, $x_i = y_{i+p'-p}$ (2ème système)

$\forall i \in P+[0,-c] \cap P+[p,p-c]$: $x_i = y_{i+p'-p}$ et $x_{i+c} = y_{i+p'-p+c}$

$\Rightarrow y_{i+p'-p} = y_{i+p'-p+c}$

$\Rightarrow j \in P+[p'-p,p'-p-c] \cap P+[p',p'-c]$, $y_j = y_{j+c}$

c-à-d y de congruence c sur P+[p'-p,p'-p] ∩ P+[p',p'].

De façon symétrique, x est de congruence c' sur P+[p-p',p-p'] ∩ P+[p,p].

Posons P_y = P+[Sup(p'-p,p'),Inf(p'-p,p')] et P_x = P+[Sup(p-p',p),Inf(p-p',p)], y est de congruence c sur P_y et x de congruence c' sur P_x.

Grâce aux résultats sur les congruences d'indices, on peut dire que :

. si c et c' non nuls et si P_y ∩ P et P_x ∩ P contiennent chacun au moins c+c' éléments, c-à-d Card(P) ≥ c+c' + |p| + |p'| alors x et y sont de congruence Pgcd(c,c') sur au moins P (car vrai sur P ∪ P_y ou P ∪ P_x).

. si c nul et c' non nul, la variable x est de congruence c' sur P_x

. si c' nul et c non nul, la variable y est de congruence c sur P_y.

Donc en conclusion:

$S_{Pm}(G_2)$ V $\{x_{i+p}=y_{i+p'}/ \forall i \in P_m\} \leq S \leq S_P(G_2)$ V $\{x_{i+p}=y_{i+p'}/ \forall i \in P\}$ car $S_P(G) \leq S_P(G'')$

où $G_2 = G_1[s <-(s,c')]$ et $G_1 = G[s' <-(s',c)]$

et P_m = P si c et c' sont non nulles ou égaux

= P_x = P+[p-p',p-p'] ∩ P+[p,p] ∩ P si c = 0

= P_y = P+[p'-p,p'-p] ∩ P+[p',p'] ∩ P si c' = 0

2) On peut maintenant le Lemme 22, sur les sommets s et s' puisque on a harmoniser leurs congruences. D'où pour le Igop G' = G_2[s' <-(s,p-p')], on obtient deux intervalles P_m et P_M définis à une constante de bornes prés par rapport à P.

CONCLUSION : Quelque soit s et s', On peut construire deux intervalles Pm et PM définis à une constante de bornes prés par rapport à P, tel que :

. si s = s', $S_{Pm}(G') \leq S \leq S_{PM}(G')$

. si s <> s', $S_{Pm}(G') \vee \{(y_i = x_{i+p-p'}) / \forall i \in P_m\} \leq S \leq S_{PM}(G') \vee \{(y_i = x_{i+p-p'}) / \forall i \in P_M\}$

Nous pouvons donc écrire l'algorithme de calcul de Pm pour (s,p) et (s',p') quelconques, posons c = Clas(s) et s' = Clas(s') :

. a_m , $b_m \in N$ **tels que** $P_m = P + [a_m, -b_m]$

. a_M , $b_M \in N$ **tels que** $P_M = P + [-a_M, b_M]$

. card_min : taille minimale de P.

Procédure substitution (s,p,s',p',a_m,b_m,a_M,b_M,*card_min);*

Début

```
|   Si s = s'
|   |     Alors % G' = G[s <-(s,|p-p'|)] %
|   |           congruence(s,p,p',a_m,b_m,a_M,b_M,card_min);
|   |     Sinon % G' = G[s' <-(s,p-p'))] %
|   |       Si (c et c' non nulles) ou ( c et c' égaux)
|   |       |     Alors a_m1 := 0; b_m1 := 0; card_min := c+c'+|p|+|p'|
|   |       |     Sinsi c est nulle
|   |       |     |     Alors % G_1 = G[s <-(s,c')] %
|   |       |     |                 a_m1:= Sup(p-p',p,0);  b_m1:= -Inf(p-p',p,0);
|   |       |     |     Sinon % G_2 = G_1[s' <-(s',c)] %
|   |       |     |                 a_m1:= Sup(p'-p,p',0);  b_m1:= -Inf(p'-p,p',0);
|   |       |     Fsi;
|   |       Fsi;
|   |       % G'= G_2[s' <-(s,p-p')] %
|   |       substitution-même-congr (s,p,s',p',a_m2,b_m2,a_M,b_M);
|   |       a_m:=a_m1+a_m2; b_m:=b_m1+b_m2;
|  Fsi;
Fin;
```

4-4-3) ALGORITHME D'ENCADREMENT DE L'UNIFIE DE DEUX IGOPS.

DEFINITION 24 : Un sommet s est dit *isolé* dans G si il n'est pas accessible à partir d'un autre sommet de G.

PROPRIETE 25.- Soient s_y un sommet isolé de G et y sa variable, y n'apparait pas en partie gauche des équations non congruentes de $S_P(G)$.

PROPRIETE 26.- On peut construire un Igop Gu (Igop unifié), un système d'équations noté $Ind_P(Gu)$, deux intervalles P_m et P_M définis à une constante de bornes près par rapport à P tels que :

$$S_{Pm}(Gu) \cup Ind_{Pm}(Gu) \leq S \leq S_{PM}(Gu) \cup Ind_{PM}(Gu)$$

où $Ind_P(Gu)$ est de la forme : $\cup \{ (y_i = x_{i+q}) / \forall i \in P \}$ et y étiquette un sommet isolé de Gu.

Principe de l'algorithme.- Trois cas peuvent se présenter :

1er Cas - Les sommets s et s' sont identiques (c-à-d x = y)

Nous savons conformément au Lemme 21, qu'il existe P_m et P_M tels que :

$S_{Pm}(G') \leq S_P(G) \vee \{ (y_{i+p'} = x_{i+p}) / \forall i \in P \} \leq S_{PM}(G')$

2ème Cas - Les sommets s et s' sont différents, mais l'un est étiqueté d'une variable, par exemple s', Lab1(s') $\in$ V. Conformément au Lemme 23, on peut construire P_m et P_M tels que :

$S_{Pm}(G') \vee \{ (y_{i+p'} = x_{i+p}) / \forall i \in Pm \} \leq S \leq S_{PM}(G') \vee \{ (y_{i+p'} = x_{i+p}) / \forall i \in PM \}$

avec G' = G[s' <- (s,p-p')].

or $S_{Pm}(G') \vee \{ y_i = x_{i+q} / \forall i \in Pm \} = S_{Pm}(G') \cup \{ y_i = x_{i+q} / \forall i \in Pm \}$

et $S_{PM}(G') \vee \{ y_i = x_{i+q} / \forall i \in PM \} = S_{PM}(G') \cup \{ y_i = x_{i+q} / \forall i \in PM \}$

car la variable y qui n'apparaissait pas en partie gauche des équations non congruentes de G, n'apparait plus non plus en partie gauche des équations non congruentes de G'.

=> $Ind_{P\#}(Gu) = \{ y_i = x_{i+q} / \forall i \in P\# \}$ et $P\# = P_m$ ou P_M

3ème Cas - Les sommets sont différents et étiquetés par des fonctions, c-à-d Lab1(s) et Lab1(s') $\in$ F.

a) Appliquons le Lemme 23, on peut se ramener à l'étude de :

$S_{P\#}(G') \vee \{ y_i = x_{i+q} / \forall i \in P\# \}$ où s' est un sommet isolé , G' = G[s'<-(s,p-p')] et P# est un intervalle de minoration ou majoration défini à une constante de bornes par rapport à P.

b) Nous avons donc à résoudre $S_P(G')\ V\ \{ y_i = x_{i+q} / \forall i \in P \}$

Supposons que les deux systèmes sont interprétés sur des intervalles différents, par exemple sur P_1 (1er système) et P_2 (2ème système) :

1er système : $\forall i \in P_1$, $x_i = Lab_1(s)(\dots)$, $y_i = Lab_1(s')(\dots)$.

2ème système : $\forall i \in P_2$, $y_i = x_{i+q}$

Si $P_2+[q,q]$ est inclus dans P_1, les équations du second système se transforment totalement en égalités sur les parties droites des équations du 1er système :

1er système : $\forall i \in P_1$, $x_i = Lab_1(s)(\dots)$, $y_i = Lab_1(s')(\dots)$.

2ème système $\equiv \forall i \in P_2$, $Lab_1(\dots) = Lab_2(\dots)$

Choisissons donc : $P_m = P \cap P+[-q,-q]$ et $P_M = P \cup P+[q,q]$

$S_{Pm}(G')\ V\ \{(y_i = x_{i+q})/ \forall i \in P_m \} \leq S \leq S_{PM}(G')\ V\ \{(y_i = x_{i+q})/ \forall i \in P_M \}$

Une condition évidente pour P_m non vide est que s et s' aient même labels de fonction :

$$Lab_1(s) = Lab_1(s') .$$

* Si l'arité du Label est nulle, le second système devient redondant : $S_{Pm}(G') \leq S \leq S_{PM}(G')$
* Si l'arité est non nulle, le second système d'équations se reportant totalement sur les successeurs de s et s', on peut le remplacer donc par :

$$\underset{k\in[1,\text{arité}]}{V} \{ (v(k)_{i+pk'} = u(k)_{i+q+pk})/ \ \forall i \in P\# \} \quad \text{où} \quad P\# = P_m \text{ ou } P$$

avec $Succ(s,k) = (s_k,p_k)$, $u(k) = Lab2(s_k)$ et $Succ(s',k) = (s_k',p_k')$, $v(k) = Lab2(s_k')$

Pour $P\# = P_m$ ou P_M, on se ramène donc à l'étude de :

$$S = S_{P\#}(G')\ V\ [\ \underset{k\in[1,\text{arité}]}{U} \{ (v(k)_{i+pk'} = u(k)_{i+q+pk})/ \ \forall i \in P\# \}]$$

On itère le procédé de minimisation et maximisation pour chacune de ces équations :

1er successeur : on calcule G", Ind(G"), $P"_m$ et $P"_M$ tel que :

$S_{P\#}(G')\ V\ \{ (v(1)_{i+p1'} = u(1)_{i+q+p1})/ \ \forall i \in P\# \}]$ est encadré par

$S_{Pm"}(G")\ U\ Ind_{Pm"}(G")$ et $S_{PM"}(G")\ U\ Ind_{PM"}(G")$ (par récurrence).

La forme du second système est toujours le même, mais dans le Igop G", les 1er successeurs de s et s' sont identiques, on se reprend donc l'encadrement commencé en b) sans tenir compte maintenant des 1ers successeurs de s et s' :

$$S_{P\#}(G")\ V\ [\ \underset{k\in[2,\text{arité}]}{V} \{ (v"(k)_{i+pk'} = u"(k)_{i+q+pk})/ \ \forall i \in P\# \}]\ U\ Ind_{P\#}(G")$$

Les équations du système Ind(G") sont négligées car elles sont attachées à un sommet isolé de G".

REMARQUE : L'algorithme de réduction du système S décrit ici, n'est pas optimisé, l'accent ayant été mis sur la facilité de compréhension (?).

ALGORITHME : Le calcul du Igop Gu, du système Ind(Gu) et des intervalles P_m et P_M qui vérifient l'inéquation $S_{Pm}(Gu) \cup Ind_{Pm}(Gu) \leq S \leq S_{PM}(Gu) \cup Ind_{PM}(Gu)$, s'écrit donc :

. a_m , $b_m \in N$ tels que $P_m = P + [a_m, -b_m]$

. a_M , $b_M \in N$ tels que $P_M = P + [-a_M, b_M]$

. card_min : taille minimale de l'intervalle P

```
Procédure encadrement (s,p,s',p',a_m,b_m,a_M,b_M,card_min);
Début
|   Si s= s'
|   |   Alors % G'= G[s <-(s,⌊p-p'⌉)] %
|   |       substitution (s,p,s',p',a_m,b_m,a_M,b_M,card_min);
|   |   Sinsi estvar(s) Alors % G'= G[s <-(s',p'-p)] %
|   |   |       substitution(s',p',s,p,a_m,b_m,a_M,b_M,card_min)
|   |   Sinsi estvar(s') Alors % G'= G[s' <-(s,p-p')] %
|   |   |       substitution(s,p,s,'p',a_m,b_m,a_M,b_M,card_min)
|   |   Sinsi Lab(s) = Lab(s')
|   |   |   Alors  % G'= G[s' <-(s,p-p')] %
|   |   |       substitution(s,p,s',p',a_m1,b_m1,a_M1,b_M1,card_min);
|   |   |       q := p-p';
|   |   |       a_m := a_m1 + Sup(-q,0); b_m := b_m1 - Inf(-q,0);
|   |   |       a_M := a_M1 - Inf(q,0) ; b_M := b_M1 + Sup(q,0);
|   |   |       Pour i=1 A Nsucc(s) Faire
|   |   |       |   encadrement(s_i,p_i+q ,s'_i,p'_i,a_m2,b_m2,a_M2',b_M2',card_min_2) ;
|   |   |       |   a_m := a_m + a_m2 + Sup(-q,0);
|   |   |       |   b_m := b_m + b_m2 - Inf(-q,0);
|   |   |       |   a_M := a_M + a_M2 - Inf(q,0);
|   |   |       |   b_M := b_M + b_M2 + Sup(q,0);
|   |   |       |   card_min := card_min+card_min_2;
|   |   |       Fait;
|   |   |   Sinon  échec (pas d'encadrement de l'unifié);
|   |   Fsi;Fsi;Fsi;
|   Fsi;
Fin;
```

Cet algorithme est identique aux congruences près à celui obtenu sur les Gops, seul un calcul de constantes (am,bm,aM,bM) y est intégré.

==> 1) Pondérations et indirections identiques

==> 2) l'encadrement de l'unifié de deux Igops existe ssi l'unifié au sens des les Gops existe.

4-4-4) ENCADREMENT DE L'UNIFICATION DE 2 IGOPS POINTES.

Avant d'encadrer l'unification de deux IGops, mettons un peu en forme nos résultats.

PROPRIETE.- On peut construire Gu et deux intervalles P_m et P_M tels que
$S = S_P(G) \cup \{(y_{i+p'}=x_{i+p}) / \forall i \in P\}$ vérifie l'inéquation suivante :

$$S_{Pm}(Gu) \cup Ind_{Pm}(Gu) \leq S \leq S_{PM}(Gu) \cup Ind_{PM}(Gu) .$$

avec . Gu homogène
. toute variable indirigée est de congruence nulle
. les indirections sont élémentaires : x -k-> z (z non indirigée)

Démonstration :

a) Représentant d'une variable indirigée.

Posons, comme pour les Gops, la fonction Représentant :

Repr(x) = (x,0) si x n'est pas indirigée
= (z,k) si x est indirigée et k est la somme des pondérations des indirections

$x_i = y_{i+p0}$ $\forall i \in P.$ $x \text{ -}p_0\text{-> } y$
$y_i = y'_{i+p1}$ $\forall i \in P.$ $y \text{ -}p_1\text{-> } y'$
.. ..
$y''_i = z_{i+pn}$ $\forall i \in P.$ $y'' \text{ -}p_n\text{-> } z$

$x = z_{i+k}$ $\forall i \in P'$ $x \text{ --k--> } z$ $k = p_0+p_1+ ..+p_n$

On voit qu'il existe un intervalle P' défini à une constante de bornes par rapport à P tel que : $\forall i \in P', x_i = z_{i+k}$

On minimise ou maximise notre système interprété sur P
$S_{P.}(Gu) \cup Ind_{P.}(Gu)$ en prenant $P_m = P \cap P'$ et $P_M = P \cup P'$
et notre nouveau système d'indirection est composée des indirections cumulées " x --k--> z " avec z non indirigée.

c) Congruence et indirection.

Dans certains cas, les indirections peuvent permettre d'étendre les congruences de certaines variables : $x \text{ --k--> } z$ et $Clas(x) = c$, $Clas(z) = c'$

On sait, par construction des indirections, que c' est un multiple de c.

On peut remplacer les équations suivantes définies pour tout i de P :

$x_i = x_{i0} \pmod c$, $x_i = z_{i+k}$, $z_i = z_{i0} \pmod{c'}$

par $\forall i \in P , x_i = z_{i+k}$ et $\forall i \in P+[k,k] \cup P , z_i = z_{i0} \pmod{c'}$

cad la congruence d'indice sur x est devenue inutile.

THEOREME. - **Soient g et g' deux Igops unifiables, on peut construire un Igop pointé h et deux intervalles P_m et P_M, égaux à P à une constante de bornes près (constante indépendante de P) tels que :** $I_{Pm}(h) \leq I_P(g) \vee I_P(g') \leq I_{PM}(h)$

Par extension, on notera h comme unifié de g et de g' : $h = g \vee g'$.

Principe : Nous savons que l'unification de deux Igops peut être représenté en terme de systèmes d'équations, sous la forme de l'unification des deux systèmes d'équations suivants :

$S = S_P(G) \vee \{ (y_{i+p'} = x_{i+p}) / \forall i \in P \}$

avec s et s' sommets de G , $Lab2(s) = u$ et $Lab2(s') = v$.

Appliquons le Théorème précédent : $S_{Pm}(Gu) \cup Ind_{Pm}(Gu) \leq S \leq S_{PM}(Gu) \cup Ind_{PM}(Gu)$

avec . Gu homogène

. toute variable indirigée est de congruence nulle et non accessible dans Gu,

. les indirections sont élémentaires : x <-k- z (z non indirigée)

Soient σ_m et σ_M les substitutions associées aux systèmes minorant et majorant

$\Rightarrow \{(i,x_i) / \forall i \in P \}.\sigma \leq I_P(g) \vee I_P(g') \leq \{(i,y_i) / \forall i \in P \}.\sigma$

a) Si la variable x n'est pas indirigée, toutes les contraintes sur cette variable et les variables dont elles dépendant sont contenues dans le système $S_P(G)$

$\Rightarrow h = G/(s,p)$

b) Si la variable et indirigée , alors la seule équation portant sur elle est cette indirection :

$Repr(x) = (z,k)$ c-à-d $\forall i \in P_m \; x_i = z_{i+k}$, on choisira donc $h = G/(z,k+p)$

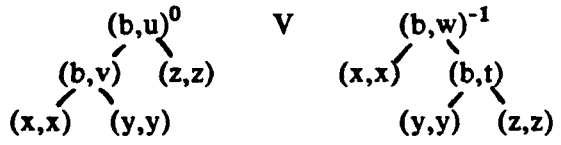
Exemple : Soient $(b,u)^0$ et $(b,w)^{-1}$ à unifier sur P

$S_P(G) = \{ u_i = b(v_i,z_i)\ ,\ v_i = b(x_i,y_i)\ ,\ w_i = b(x_i,t_i)\ ,\ t_i = b(y_i,z_i)\ /\ \forall\ i \in P \}$

L'unification des Igops :

. Etape 0 : $S = S_P(G)\ V\ \{ u_i = w_{i-1}\ /\ \forall\ i \in P \}$

. Etape 1 : On peut appliquer le théorème 1, mais u et v n'apparaissent pas en partie gauche des équations du 1er système (G'=G).



Sur les successeurs :

$S_{P.}(G)\ V\ \{ v_i = x_{i-1}\ /\ \forall\ i \in P. \}\ V\ \{ z_i = t_{i-1}\ /\ \forall\ i \in P. \}$

avec $P\# = P_m = P+[1,1] \cap P = P+[1,0]$

ou $P_M = P+[-1,-1] \cup P = P+[-1,0]$

. Etape 2 : Sur les fils gauches , $S_P(G')\ V\ \{ u_i = w_{i-1}\ /\forall\ i \in P \}$ se réécrit sur v et x :

$S_{P\#}(G')\ V\ \{ v_i = x_{i-1}\ /\ \forall\ i \in P\# \}$

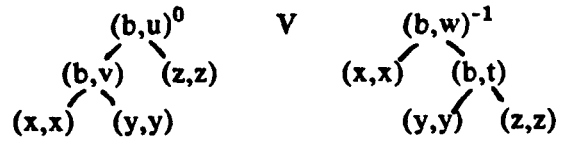
avec $P\# = P_m = P+[1,1] \cap P = P+[1,0]$

ou $P_M = P+[-1,-1] \cup P = P+[-1,0]$

On applique le théorème 1 : $G' = G[x <- (v,1)]$

$P_m <- P_m \cap P_m+[-1,-1] = P+[1,-1]$

$P_M <- P_M \cup P_M+[1,1] = P+[-1,1]$



$S_{P\#}(G') = \{ u_i = b(v_i,z_i)\ ,\ v_i = b(v_{i+1},y_i)\ ,\ w_i = b(v_{i+1},t_i)\ ,\ t_i = b(y_i,z_i)\ /\ \forall\ i \in P\# \}$

. Etape 3 : Sur les fils droits ,, $S_{P\#}(G')$ V { $u_i = w_{i-1}$ / ∀ i ∈ P# } se réécrit sur z et t :

$S_{P\#}(G')$ V { $z_i = t_{i-1}$ / ∀ i ∈ P# }

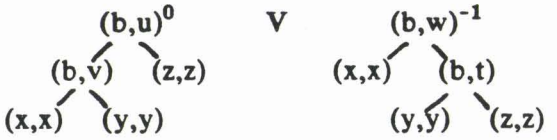
avec P# = P_m <- P_m+[1,1] ∩ P_m = P+[2,-1]

ou P_M <- P_M+[-1,-1] ∪ P_M = P+[-2,1]

On applique le théorème 1 : Gu = G'[z <- (t,-1)]

P_m <- P_m ∩ P_m+[0,0] = P+[2,-1]

P_M <- P_M ∪ P_M+[0,0] = P+[-2,1]



$S_{P\#}(Gu)$ = { u_i = $b(v_i,t_{i-1})$, v_i = $b(v_{i+1},y_i)$, w_i = $b(v_{i+1},t_i)$, t_i = $b(y_i,t_{i-1})$ / ∀ i ∈ P# }

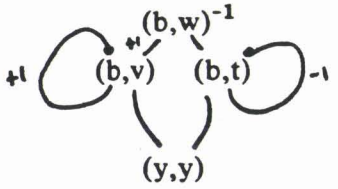
Le résultat est donc : $S_P(G)$ V { $u_i = w_{i-1}$ / ∀ i ∈ P } est encadré par

. $S_{Pm}(Gu)$ ∪ {($x_i = v_{i+1}$)/ ∀ i ∈ P_m } ∪ {($z_i = t_{i-1}$)/ ∀ i ∈ P_m } et P_m = P+[2,-1]

. $S_{PM}(Gu)$ ∪ {($x_i = v_{i+1}$)/ ∀ i ∈ P_M } ∪ {($z_i = t_{i-1}$)/ ∀ i ∈ P_M } et P_M = P+[-2,1]

L'unification des deux Igops pointés est aussi encadré par le Igop h :

h =



sur P_m = P+[2,-1]

P_M = P+[-2,1]

4-5) Test d'occurrence de l'interprété d'un Igop homogène.

Nous allons donner ici, un critère nécessaire et suffisant pour qu'un système $S_P(G)$ vérifie le test d'occurrence, ceci pour un Igop G homogène et un intervalle P d'interprétation assez grand.

Un système d'équations S quelconque satisfait le test d'occurrence ssi on ne peut pas déduire des équations de S, une équation de la forme : $x = t$ avec $t <> x$ et $x \in Var(t)$.

4-5-1) BOUCLES DE PONDERATIONS NULLES DANS UN IGOP.

NOTATION.- Un chemin m sera appelée une boucle sur le sommet s dans le Igop G si m va de s à lui-même via $s_1, s_2, \ldots, s_k$ $(k \geq 0)$.
La pondération d'une boucle étant, bien-sûr, la somme des pondérations élémentaires rencontrées le long de ce chemin.
Une boucle sera dite élémentaire si les sommets traversés sont différents : $s_i <> s_j <> s$.

Exemple : Soit le Igop suivant :

Il y a une infinité de boucles sur s et s'
par exemple sur s : 1^n.nil , $(1.2.3)^n$.nil , ..
mais un nombre fini de boucles élémentaires :
sur s : 1.nil, 2.3.nil
sur s' : 2.nil, 3.2.nil

DEFINITION.- Un Igop G homogène vérifie la Restriction sur les Pondérations de Boucles (notée R.P.B.) ssi il n'existe pas de boucle, sur un sommet s , de pondération nulle modulo la congruence de s : soit m, une boucle sur s, Pondération(m) mod Clas(s) <> 0 .

PROPRIETE.- Un Igop G vérifie la Restriction sur les Pondérations de Boucles (notée R.P.B.) ssi les trois conditions suivantes sont valides :

. Il n'existe pas de boucles élémentaires dont la pondération est nulle.

. Il n'existe pas m_1 et m_2, deux boucles élémentaires sur le même sommet dont les pondérations k_1 et k_2 sont de signes opposés.

. Il n'existe pas de boucles élémentaires sur un sommet de congruence non nulle.

Démonstration :

1) La R.P.B n'est pas vérifiée => Il existe une boucle de pondération nulle (modulo Clas). L'une des trois conditions n'est pas vérifiée :

1ère Condition fausse : immédiat

2ème Condition fausse : il existe m_1 et m_2, boucles sur le même sommet s de pondérations k_1 et k_2 de signes opposés . Le chemin $m_1^{|k2|}.m_2^{|k1|}$ est un chemin bouclant sur s et de pondération nulle.

3ème Condition fausse : il existe m, boucle sur s, sommet de congruence non nulle. Posons k, la pondération de m et c est la congruence de s, alors le chemin $m' = m^c$ est de pondération $c*k$, c-à-d nulle modulo c.

2) La R.P.B est vérifiée => Il n'existe pas de boucle de pondération nulle (modulo Clas).

Soit m, une boucle sur le sommet s, montrons que la pondération de m est non nulle modulo la congruence de s. Il existe bien-sûr des boucles élémentaires sur ce sommet et la congruence de s nulle car la 3ème condition de la R.P.B. est vérifiée.

Il nous suffit donc de montrer que la pondération de m est non nulle.

Par hypothèse, les boucles élémentaires sur le sommet s sont de pondérations, soit toutes strictement positives, soit toutes strictement négatives.

Soit s' un sommet traversé par une boucle élémentaire m de s :

m = m1.m2 avec s --m1--> s' --m2--> s

=> m2.m1 est une boucle élémentaire de s' et elle est de même pondération que m.

Récursivement, définissons l'ensemble des sommets accessibles directement ou non par des boucles élémentaires :

. s ∈ Dépend(s)

. si s' ∈ Dépend(s), alors tous les sommets accessibles par une boucle élémentaire à partir de s' appartiennent aussi à Dépend(s).

Notons Elem(s) = { m boucle élémentaire de s' / s' ∈ Dépend(s) }

Grâce à la remarque précédente, toutes les pondérations de Elem(s) s sont de même signe. Or toute boucle m sur s est une composition d'éléments de Elem(s), c-à-d que la pondération de m est une somme de pondérations d'éléments de Elem(s) :

$Pond(m) = a_1*Pond(m_1) + a_2*Pond(m_2) + ... + a_k*Pond(m_k)$

avec $a_i \in N$ dont au moins un est non nul (car m non vide) et $m_i \in Elem(s)$.

Comme les Pondérations des m_i sont toutes non nulles et de même signe, m est de pondération non nulle.

REMARQUE.- Si la R.P.B. n'est pas vérifiée, il existe une boucle sur un sommet s , boucle de longueur bornée et de pondération nulle modulo la congruence du sommet.
R.P.B. non satisfaite => il existe m , boucle sur s tel que :

. Pondération(m) modulo Clas(s) = 0

. Longueur(m) $\leq$ Card(G) * [2*Card(G)*MaxPond(G) + MaxClas(G) + 1]

(Card(G) désigne le nombre de sommets de G)

Démonstration : En effet par construction, dans la démonstration précédente, cette longueur est , pour chacune des conditions non satisfaites :

1) inférieure à la longueur de la plus grande boucle élémentaire.

2) inférieure à deux fois le produit de la plus grande boucle élémentaire et de la plus grande pondération de boucle élémentaire.

3) inférieure au produit de la longueur de la plus grande boucle élémentaire et de la plus grande congruence.

On peut majorer, grossièrement la longueur de ces chemins par :

1) Card(G)
2) 2*Card(G) * [Card(G) * MaxPond(G)]
3) Card(G) * MaxClas(G)

En faisant la somme, on obtient la majoration suivante :

Card(G) * [2*Card(G)*MaxPond(G) + MaxClas(G) + 1]

CONVENTION : Si la R.P.B. est vérifiée, tout sommet bouclant de G admet des boucles qui sont de pondérations soient toutes strictement positives soient toutes strictement négatives.

On parlera d'un sommet bouclant *positif*, respectivement *négatif*.

4-5-2) TEST D'OCCURRENCE DU SYSTEME $S_P(G)$.

Montrons que pour P assez grand, la condition nécessaire et suffisante pour que $S_P(G)$ satisfasse le test d'occurrence est que G vérifie la R.P.B.

PROPRIETE FONDAMENTALE.- Pour P assez grand, le système $S_P(G)$ satisfait le test d'occurrence ssi il n'existe pas de boucles, dans G, de pondération nulle (modulo Clas).

Démonstration :

A) Le système $S_P(G)$ étant non "déterministe", nous allons, pour faciliter la démonstration, l'encadrer par deux systèmes "déterministes".

Soient les systèmes $Sm_P(G)$ et $SM_P(G)$ définis sur $S_P(G)$:

. Equations non congruentes :

$(x_i = t) \in S_P(G) \Rightarrow (x_i = t) \in Sm_P(G)$ et $SM_P(G)$ pour $i \in P_{/Clas(x)}$

. Equations congruentes, i mod Clas(x) = i0

$(x_i = x_{i0}) \in Sm_P(G)$, $\forall i \in P$

$(x_i = x_{i0}) \in SM_P(G)$, $\forall i \in P+[-MaxPond(G),MaxPond(G)]$

Montrons qu'ils vérifient l'inégalité : $Sm_P(G) \leq S_P(G) \leq SM_P(G)$.

. $Sm_P(G) \leq S_P(G)$, car $Sm_P(G)$ est inclus syntaxiquement dans $S_P(G)$.

. $S_P(G) \leq SM_P(G)$. Tous les indices des variables apparaissant dans $S_P(G)$ appartiennent à P + [-MaxPond(G),MaxPond(G)].

Or nous savons que le non-déterministe de $S_P(G)$ peut générer de nouvelles égalités entre des variables de même symbole.

ex : $(a,x):2 \xrightarrow{-1} (y,y):2$ $\quad \forall i \in [j,k]$ $\quad . x_i = a(y_{i-1})$; $. x_i = x_{i \bmod 2}$; $. y_i = y_{i \bmod 2}$

En effet y est de congruence 2 sur [j-1,k], car

$x_j = x_{j \bmod 2}$, $x_j = a(y_{j-1})$, $x_{j \bmod 2} = a(y_{(j \bmod 2)-1}) \Rightarrow y_{j-1} = y_{(j-1) \bmod 2}$

Comme G est homogène, on maximise ces égalités en étendant l'interprétation des congruences sur P+[-MaxPond(G),MaxPond(G)]

On peut alors épurer les équations congruentes en ne conservant que celles d'indices , représentants canoniques de la congruence :

$Sm_P(G) \leq S_P(G) \leq SM_P(G) = Sm_{PM}(G)$ avec PM = P+[-MaxPond(G),MaxPond(G)]

B) Construire une équation $x_i = t(x_i)$ à partir des équations de $Sm_P(G)$ ou revient à utiliser un certain nombre de ses équations :

$x_{i0} = x_{j0}$ (mod Clas(x)) et $x_{j0} = t_0(y_{i1})$,
$y_{i1} = y_{j1}$ (mod Clas(y)) et $y_{j1} = t_1(z_{i2})$
.. ..
$z'_{in-1} = z'_{jn-1}$ (mod Clas(z')) et $z'_{jn-1} = t_{n-1}(x_{in})$
$x_{in} = x_{jn}$ (mod Clas(x))
où . $n \geq 1$
. $i = i_0 = j_n$
. $i_0, .., i_n$ et $j_0, .., j_n \in P$

Ce qui correspond au chemin $m = m_1.m_2. .. .m_n$ dans les équations de $Sm_P(G)$:
$x_{j0} = \#(... , y_{i-1} , ...)$, $y_{j1} = \#(... , z_{i2} , ...)$, etc ...
$m_1^{ième}$ fils $m_2^{ième}$ fils

La somme des pondérations sur ce chemin est :

$$Pond(m) = (i_1 - j_0) + (i_2 - j_1) + .. + (i_n - j_{n-1})$$

Comme G est homogène, les variables x,y,z, ..,z' sont toutes de même congruence :

$$Clas(x) = Clas(y) = ... = Clas(z') .$$

$$\Rightarrow Pond(m) \equiv (i_1 - i_0) + (i_2 - i_1) + .. + (i_n - i_{n-1}) \pmod{Clas(x)}$$
$$\equiv (i_n - i_0) \equiv (i - i) = 0$$

Donc construire une équation $x_i = t(x_i)$ à partir du système $Sm_P(G)$, revient à rechercher une boucle de pondération nulle modulo la congruence dans le graphe G.

C) Nous savons que la recherche d'une boucle de pondération (modulo Clas) peut se limiter aux boucles de longueur inférieure à :

$$Card(G) * [2*Card(G)*MaxPond(G) + MaxClas(G) + 1]$$

Montrons que l'on peut construire un intervalle P, tel que une boucle m est effective, c-à-d que les contraintes de la construction précédente peuvent être respectées (les indices $\in$ P).

Or pour tout chemin m, il existe un intervalle P de taille linéairement dépendante de la longueur du chemin m tel que dans la construction précédente :

$$i_0, .., i_n \in P \quad et \quad j_0, .., j_n \in P$$

Il suffit de prendre par exemple un intervalle P contenant au moins 2*MaxPond(G)*longueur(m) éléments. En effet, à partir de la pondération milieu de P, la pondération maximale du chemin m ou de l'un des facteurs gauches de m sera toujours en valeur absolue inférieur à MaxPond(G)*Longueur(m).

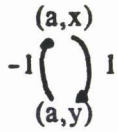
En conclusion, pour P assez grand, $Sm_P(G)$ satisfait le test d'occurrence ssi il n'y a pas dans G, de boucles de pondération nulle (mod Clas). Or $Sm_P(G)$ satisfait le test d'occurrence ssi $Sm_{PM}(G)$ le satisfait, pour P de taille assez grande.

Donc $S_P(G)$ satisfait le test d'occurrence ssi il n'y a pas de boucles de pondération nulle (modulo Clas). P doit être de taille supérieure à une certaine constante :

$2 * MaxPond(G) * [longueur(m) + 1]$

avec $longueur(m) \leq Card(G)*[2*Card(G)*MaxPond(G) + MaxClas(G) + 1]$

Exemples :

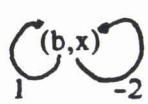
1) Soit G_1, le Igop suivant : 

$a \in F$, x et $y \in V$

G_1 ne satisfait pas la première condition de la R.P.B.

et $S_P(G_1)$ ne satisfait le test d'occurrence pour $Card(P) \geq 2$:

$S_{[1,2]}(G_1) = \{ x_1 = a(y_2) , x_2 = a(y_3) , y_1 = a(x_0) , y_2 = a(x_1) \} \Rightarrow x_1 = a(a(x_1))$

2) Soit G_2, le Igop suivant : 

$b \in F$, $x \in V$

G_2 ne satisfait pas la seconde condition de la R.P.B.

et $S_P(G_2)$ ne satisfait le test d'occurrence pour $Card(P) \geq 3$.

$S_{[1,3]}(G_2) = \{ x_1 = b(x_2,x_{-1}), x_2 = b(x_3,x_0) , x_3 = b(x_4,x_1) \}$

$x_1 = b(x_2,x_{-1}) \Rightarrow x_1 = b(b(x_3,x_0),x_{-1})$ car $x_2 = b(x_3,x_0)$

$\Rightarrow x_1 = b(b(b(x_4,x_1),x_0),x_{-1})$ car $x_3 = b(x_4,x_1)$

3) Soit G_3, le Igop suivant : 

sommet de congruence 1, $a \in F$, $x \in V$

G_3 ne satisfait pas la troisième condition de la R.P.B.

et $S_P(G_3)$ ne satisfait le test d'occurrence pour $Card(P) \geq 2$.

$S_{[1,2]}(G_3) = \{ x_1 = a(x_2) , x_2 = a(x_3) , x_2 = x_{2 \bmod 1} \text{ (c-à-d } x_1) \} \Rightarrow x_1 = a(x_1)$

4-6) LES IGOPS PARFAITS : interprétation finie des Gops.

DEFINITION.- Un Igop G sera dit *parfait* si tous ses sommets de label fonction sont de congruence nulle.

Remarque : Si G est un Igop parfait, $S_P(G)$ est un système déterministe , c-à-d avec au plus une équation par variable indicée en partie droite.

PROPRIETE.- Soit G un Igop parfait, la substitution $\sigma_P(G)$ associée à $S_P(G)$ peut être construite grâce à la fonction Descendant suivante :

$Desc_P((s,p), nil) = (s,p)$

$Desc_P((s,p),i.m) = Desc_P((s_i,p+p_i),m)$ si $p \in P$ et $Succ(s,i) = (s_i,p_i)$

La substitution $\sigma_P(G)$ est composée des équations suivantes : $(x_q = t)$

ssi $\forall m \in Dom(t)$, $Desc_P((s_x,q),m) = (s,p)$, $Lab_1(s) = u$ et $Lab_2(s) = y$

et $t(m) = u$ si $u \in F$ et $p \in P$

$t(m) = y_{p \bmod P Clas(s)}$ sinon

Démonstration : Par récurrence sur le longueur du chemin m de t.

Posons $Desc((s_x,i),m) = (s,p)$, $Lab_1(s) = u$, $Lab_2(s) = y$.

. $m = nil$: $Desc((s_x,q),nil) = (s_x,q) \Rightarrow y = x$

Si $u \in F$, $Clas(x) = 0$ (par hyp.) et sur le symbole y, les seules équations de $S_P(G)$ sont :

$\forall j \in P$, $x_j = u(...) \Rightarrow t(nil) = u$ (si $q \in P$) , $= x_q$ (sinon) .

Si $u \in V$, alors $u = x$ et les seules équations sur ce symbole sont les équations congruentes :

$x_q = x_{q0}$ où $q_0 = q \bmod_P Clas(x)$ et $t(nil) = x_{q0}$

. $m = i.m_1 \Rightarrow$ il existe une équation $x_q = u(a_1 , .. , a_i , .. , a_n)$ dans $S_P(G)$

$\Rightarrow u \in F_n$ et $q \in P$.

Posons $Succ(s,i) = (s_i,p_i)$, $Lab2(s_i) = v$, le sous-arbre t.i correspond à la valeur de la variable v_{p+pi}, dans $\sigma_P(G)$ car $x_p = u(... , v_{p+pi} , ...) \in S_P(G)$.

Or $Desc((s,q),i.m_1) = Desc((s_i,q+p_i),m_1)$ si $q \in P$.

Il nous suffit d'appliquer donc l'hypothèse de récurrence sur m_1.

CORROLAIRE.- Soit g un Igop parfait, pointé de racine (s,p) alors
$\forall$ m $\in$ Dom(A_P(g,k)), posons Desc((s,p+k),m) = (s',q), Lab1(s') = u et Lab2(s') = x :

$A_P(g,k)(m) = u$ si $u \in F$ et $q \in P$

$= x_{q0}$ sinon avec $q_0 = q \bmod_P Clas(s')$

REMARQUE - <u>Les Igops sont donc une généralisation des Gops pointés.</u>
Si l'on compare les Gops pointés et les Igops pointés parfaits, on peut remarquer deux différences : . La présence d'un second label caractéristique pour chaque sommet,
. Une condition d'appartenance pour la pondération cumulée à un intervalle P d'interprétation,
le reste étant totalement similaire, en particulier seules les variables possèdent une congruence.
Si on interprète un Igop pointé parfait sur Z tout entier, on obtient l'interprété du Gop pointé correspondant (c-à-d sans second label).

Les propriétés particulièrement intéressantes des Igops parfaits seront largement utilisées dans la suite, en particulier grâce à la propriété suivante :

PROPRIETE.- Soit G un Igop homogène satisfaisant la Restriction sur les Pondérations de Boucles, on peut construire un Igop parfait G' tel que $\forall$ P intervalle de Z, on peut construire un intervalle P_M défini à une constante de bornes près sur P satisfaisant l'inéquation suivante :

$$S_P(G') \leq S_P(G) \leq S_P(G').$$

Si G = (X,Lab,Succ,Clas) , G' = (X,Lab,Succ,Clas') avec Clas' définie par :
Clas'(s) = 0 si $Lab_1(s) \in F$ et Clas'(s) = Clas(s) sinon.

<u>Démonstration</u> : La seule condition de la R.P.B. que nous utiliserons est celle concernant la congruence nulle des sommets bouclants.
Soit G =(X,Lab,Succ,Clas) un Igop et X', l'ensemble des sommets de label fonction de congruence non nulle : $X' = \{ s \in X / Lab_1(s) \in F \text{ et } Clas(s) <> 0 \}$.
On peut se définir une relation d'ordre partiel entre ces éléments de X', car ceux ne sont pas des sommets bouclants (R.P.B.): $s < s'$ ssi il existe un chemin non vide allant de s à s'.

Soit s un sommet minimal de X', et posons $Lab_1(s) = u$ et $Lab_2(s) = x$,
dans $S_P(G)$, les seules égalités sur les x_i déductibles sont les équations congruentes :

$$x_i = x_j \text{ avec } i \equiv j \pmod{Clas(s)}$$

car tous les sommets ancêtres de s sont de congruence nulle (sommet minimal).
Posons $G_s = (X,Lab,Succ,Clas')$ où $Clas'(s) = 0$ et $Clas'(s') = Clas(s')$ si $s' <> s$
Deux possibilités :

a) u est d'arité nulle, alors $\forall P$, $S_P(G_s) = S_P(G)$

b) u est d'arité n non nulle et soit s_i, le $i^{ème}$ sommet successeur de s :

$Succ(s,i) = (s_i,p_i)$ et $Lab_2(s_i) = v$.

La congruence de s engendre des égalités sur les v_i :

$v_i = v_j$ si $i, j \in P+[p_i,p_i]$ et $i \equiv j \bmod Clas(s)$

Or G est homogène c-à-d Clas(s) et un multiple de $Clas(s_i)$.

Cette double congruence est donc encadrer par v de congruence Clas(v) sur :

. P pour la minoration.

. $P \cup P+[p_i,p_i]$ pour la majoration.

Si l'on pose $P_M = P \cup P+[p_1,p_1] \cup P+[p_2,p_2] \cup \ldots \cup P+[p_n,p_n]$

$= P + [\min(0,p_1, .., p_n) , \max(0,p_1, .. ,p_n)]$

alors $S_P(G_s) \leq S_P(G) \leq S_{PM}(G_s)$.

Nous pouvons ainsi itérer le procédé en annulant "par couches" la congruence de ces sommets et construire un intervalle P_M défini à une constante près par rapport à P, tel que :

$$S_P(G') \leq S_P(G) \leq S_{PM}(G') \quad \text{et} \quad \forall s \in X', Clas_{G'}(s) = 0.$$

Toutes les propriétés et théorèmes relatifs à l'encadrement de l'unifié des Igops pointés peuvent être adaptés avec comme hypothèse supplémentaire , le Igop Gu unifié est parfait.

CORROLAIRE.- Soient $S_p(G)$ et $\{(y_{i+p'}=x_{i+p})/\ \forall\ i \in P\ \}$, deux systèmes unifiables avec test d'occurrence et soit S leur unifié, on peut construire un Igop Gu parfait et deux intervalles P_m et P_M tels que : $S_{Pm}(Gu) \cup Ind_{Pm}(Gu) \leq S \leq S_{PM}(Gu) \cup Ind_{PM}(Gu)$.

avec . toute variable indirigée est de congruence nulle

. les indirections sont élémentaires : x -k-> z (z non indirigée)

CORROLAIRE.- Soient g et g' deux Igops unifiables avec test d'occurrence, on peut construire un Igop pointé parfait h et deux intervalles P_m et P_M, égaux à P à une constante de bornes près (constante indépendante de P) tels que : $I_{Pm}(h) \leq I_P(g) \vee I_P(g') \leq I_{PM}(h)$

REMARQUE : Cette propriété est une généralisation sur les Igops de résultats démontrés dans le cadre des Gops. Car si P = Z, le seul intervalle défini à une constante près sur Z est Z , donc PM = Pm = Z. L'unification est bien interne sur les Gops.

Chapitre 5

ETUDE DE L'APPLICATION FINIE D'UNE REGLE
(décidabilité de sa terminaison)
ETUDE DU TQ PROLOG

5-1) Etude de l'application finie d'une règle récursive.

Soit donc P(ß) -> P(ϑ), une règle récursive, nous savons qu'elle peut engendrer une infinité de réécritures (sans test d'occurrence) ssi l'unifié de ß et ϑ existe .

Dans ce chapitre, nous étudions l'application finie d'une régle récursive simple, P(ß)->P(ϑ) dans le programme :

P(α) -> ;
P(ß) -> P(ϑ) ;
P(T) ?

On supposera que les trois règles n'ont aucune variable commune :

Var(T) ∩ (Var(ß)∪Var(ϑ)) = Var(α) ∩ (Var(ß)∪Var(ϑ)) = Var(T) ∩ Var(α) = ∅.

5-1-1) FORMALISATION EN TERMES D'IGOPS.

A un arbre ß ∈ M(F,V) , nous avons associé les arbres indicés :

$ß_1$, $ß_2$, ... , $ß_n$ ∈ M(F,VxZ), de même pour ϑ : $ϑ_1$, .. , $ϑ_n$.

La recherche d'une solution du TQ Prolog avec n applications de la règle récursive, peut se caractériser par la plus petite substitution σ(n) vérifiant :

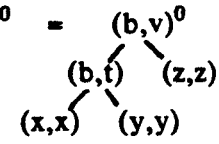
. T.σ(n) = $ß_1$.σ(n)

. $ß_i$.σ(n) = $ϑ_{i-1}$.σ(n) pour tout i ∈ [2,n]

. α.σ(n) = $ϑ_n$.σ(n)

Nous pouvons construire à partir de ß , un Igop pointé dont l'interprété est la séquence d'arbres indicés $ß_1$, $ß_2$, .. , $ß_n$.

Exemple : ß = b(b(x,y) , z)

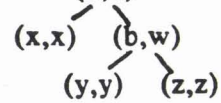


u,v,x,y,z ∈ V et b ∈ F, toutes les pondérations et les congruences sont nulles dans $ß^0$

Ce Igop ainsi défini, nous pouvons donc écrire, pour E1 = P = [1,n]

$I_{E1,P}(ß^0) = \{ (1,ß_1) , (1,ß_1) , .. , (n,ß_n) \}$

De même : $\gamma = b(x, b(y,z))$ $\gamma^{-1} = (b,u)^{-1}$



Pour E2 = [2,n+1] et P = [1,n] : $I_{E2,P}(\gamma^{-1}) = \{ (2,\gamma_1), (3,\gamma_2), \ldots, (n+1,\gamma_n) \}$

La résolution du TQ s'écrit donc: $\{ (1,T) \} \vee I_{E1,P}(\beta^0) \vee I_{E2,P}(\gamma^{-1}) \vee \{ (n,\alpha) \}$

CARACTERISATION.- L'existence et la forme de la solution obtenue par n applications de la règle récursive est pleinement caractérisée par l'unification suivante d'ensembles ordonnés :

$\{ (1,T) \} \vee I_{E1,P}(\beta^0) \vee I_{E2,P}(\gamma^{-1}) \vee \{ (n,\alpha) \}$

avec E1 = P = [1,n] et E2 = [2,n+1]

CARACTERISATION EQUIVALENTE.- La recherche d'une solution au programme TQ Prolog peut être exprimer sous la forme de l'unification des quatre systèmes d'équations suivants :

$S(n) = S_{[1,n]}(G) \vee \{ (u_i = v_{i-1}) / i \in [2,n] \} \vee \{ (u_1 = T) \} \vee \{ (v_n = \alpha) \}$

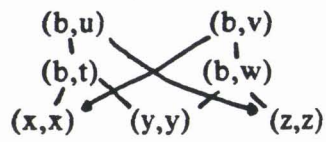
où G est le Igop contenant ß et γ , u et v ,les labels variables des sommets de ß et γ dans G .

Si l'on note $\sigma(n)$, la substitution associée à S(n), alors $\sigma(n)$ est la plus petite substitution vérifiant : $T.\sigma(n) = \beta_1.\sigma(n) \rightarrow \gamma_1.\sigma(n) = \beta_2.\sigma(n) \ldots \gamma_{n-1}.\sigma(n) = \beta_n.\sigma(n) \rightarrow \gamma_n.\sigma(n) = \alpha.\sigma(n)$

On posera dans la suite : $\beta_i.\sigma(n) = t_{i,n}$ **pour** $i \in [1,n]$ **et** $\gamma_n.\sigma(n) = t_{n+1,n}$

Exemple : $\beta = b(b(x,y),z)$ et $\gamma = b(x, b(y,z))$

G :



Toutes les pondérations et les congruences sont nulles et chaque sommet est étiqueté d'une variable (2ème Label) qui lui est propre.

. Le système $S_{[1,n]}(G)$ est égal à :

$\{ u_i = b(t_i,z_i)$, $t_i = b(x_i,y_i)$, $v_i = b(x_i,w_i)$, $w_i = b(y_i,z_i)$ / $i \in [1,n-1]\}$

Dans la substitution $\sigma_{[1,n]}(G)$ associée : $u_i = \beta_i = b(b(x_i,y_i),z_i)$ et $v_i = \gamma_i = b(x_i,b(y_i,z_i)$

. Contraintes pour n réécritures : $\forall i \in [2,n]$, $\beta_i \vee \gamma_{i-1}$ c-à-d $\{ u_i = v_{i-1} / \forall i \in [2,n] \}$

. Unification avec la question : $\beta_1 \vee T$ c-à-d $\{ u_1 = T \}$

. Unification avec le fait : $\gamma_n \vee \alpha$ c-à-d $\{ v_n = \alpha \}$

5-1-2) CONTRAINTES POUR N ITERATIONS D'UNE REGLE RECURSIVE BOUCLANTE.

Nous allons d'abord limiter notre étude aux deux premiers systèmes d'équations :

$\Sigma(n) = S_{[1,n]}(G) \vee \{ (u_i = v_{i-1}) / i \in [2,n] \}$.

RAPPEL.- En application des résultats du chapitre précédent,il existe n_0 tel que $\forall n \geq n_0$:

$S_{Pm}(Gu) \cup Ind_{Pm}(Gu) \leq S_{[1,n]}(G) \vee \{u_i = v_{i-1} / i \in [2,n] \} \leq S_{PM}(Gu) \cup Ind_{PM}(Gu)$

où Gu est le Igop unifié, Ind exprime les indirections de l'unification

P_m ,P_M non vides, sont définis à une constante de bornes près sur [1,n]

On notera dans la suite Sm(n), le système minorant : $Sm(n) = S_{Pm}(Gu) \cup Ind_{Pm}(Gu)$

PROPRIETE.- Il existe $n_1 \in N$ et Sg, $Sd_{/n}$ deux systèmes d'équations constants, respectivement par rapport à 1 et n, tels que $\forall n \geq n_1$: $\Sigma(n) = Sg \cup Sm(n) \cup Sd_{/n}$

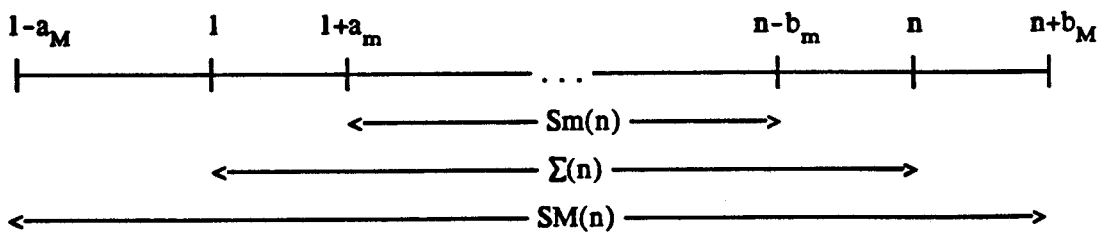
où Sg représente les effets de bords à gauche

$Sd_{/n}$ .. . . à droite.

Démonstration : Nous savons que pour tout $n \geq n_0$, on peut construire les intervalles de minoration et majoration, restriction ou extension constante par rapport à l'intervalle [1,n].

Soient Sm(n) et SM(n) les systèmes d'équations minorant et majorant correspondant et $\Sigma(n)$ une forme de la solution exacte : $Sm(n) \leq \Sigma(n) \leq SM(n)$

sachant que $P_m = [1,n]+[a_m,-b_m]$ et $P_M = [1,n]+[-a_M,b_M]$.



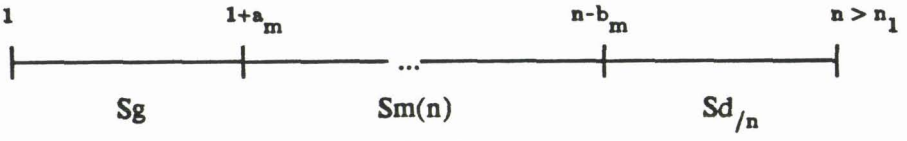
Toutes les variables de VxP_m sont connues dans Sm(n) et de la même façon dans SM(n) et donc aussi dans la solution exacte. Reste donc à définir les variables de $Vx([1,n]-P_m)$ présentes dans le système majorant, et par la force des choses, absentes du système minorant.

On peut donc exprimer la solution exacte sous la forme : $\Sigma(n) = Sg(n) \cup Sm(n) \cup Sd(n)$
avec Sg(n) représentant les contraintes sur les variables d'indices $\in [1,a_m]$ (à gauche)
et Sd(n) , les contraintes sur les variables d'indices $\in [n-b_m+1,n]$ (à droite)

Montrons qu'à partir d'un certain rang n_1 :

1) Sg(n) est constant, c-à-d Sg(n+1) = Sg

2) Sd(n) est constant par rapport à n, c-à-d $Sd(n) = Sd_{/n} = Sd(n-1)->1$, le système Sd(n-1) dont tous les indices de variables ont été incrémentés de 1.



Sg porte sur les variables d'indices $\in [1,a_m]$

Sm(n) .. .. $\in [1+a_m,n-b_m] = [1,n]+[a_m,-b_m]$

$Sd_{/n}$ .. .. $\in [n-b_m+1,n]$

On peut noter que, si l'on pose $Kg = a_M + a_m$ et $Kd = b_M + b_m$ alors :

$$SM(n)->Kg \equiv Sm(n+Kd+Kg)$$

Nous supposerons, pour simplifier la démonstration que Kg = Kd = K, il suffit en effet de maximiser encore SM(n), c-à-d a_M ou b_M.

Or $Sg(n) \cup Sd(n) \leq \Sigma(n) \leq SM(n)$

$\Rightarrow Sg(n)->K \cup Sd(n)->K \leq SM(n)->K = Sm(n+2K)$ (1)

Supposons maintenant que les systèmes Sg(n) et Sd(n) restent constants pendant K réécritures supplémentaires, c-à-d Sg(n+K) = Sg(n) et Sd(n+K) = Sd(n)->K alors :

$\Sigma(n+2K) = \Sigma(n+K) \vee \Sigma(n+K)->K$

= (Sg(n+K) ∪ Sm(n+K) ∪ Sd(n+K)) ∨ (Sg(n+K)->K ∪ Sm(n+K)->K ∪ Sd(n+K)->K)

= (Sg(n+K) ∪ Sm(n+2K) ∪ Sd(n+K)->K) ∨ (Sg(n+K)->K ∪ Sd(n+K)

car Sm(n+K) ∨ Sm(n+K)->K = Sm(n+2K)

= (Sg(n) ∪ Sm(n+2K) ∪ Sd(n)->2K) ∨ (Sg(n)->K ∪ Sd(n)->K)

car Sg(n+K) = Sg(n) et Sd(n+K) = Sd(n)->K

= Sg(n) ∪ Sm(n+2K) ∪ Sd(n)->2K car (1) : Sg(n)->K ∪ Sd(n)->K ≤ Sm(n+2K)

D'où le résultat :

si $Sg(n+K) = Sg(n)$ et $Sd(n+K) = Sd(n)->K$ alors $Sg(n+2K) = Sg(n)$ et $Sd(n+2K) = Sd(n)->2K$

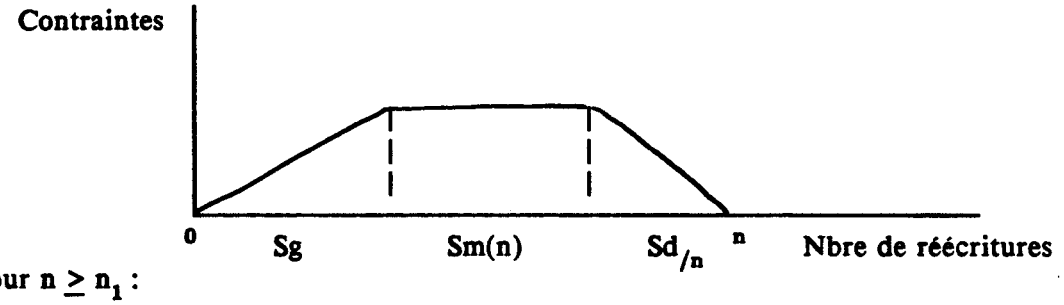
Il existe une constante $n_1 > n_0$ tel qu'au delà les systèmes d'équations gauche Sg(n) et droit Sd(n) sont constants $n_0 \leq n_1 \leq n_0 + K*Card(G)*Card([1,n]-P_m)$ car :

. $Sg(n+1) \geq Sg(n)$ et $Sd(n+1) \geq Sd(n)->1$ (suites croissantes)

. le nombre des équations qu'ils peuvent contenir est bornée : $Card(G)*Card([1,n]-P_m)$

. leur croissance stricte implique une contrainte sur une nouvelle variable toutes les K réécritures.

Schématisation de la réécriture : On peut représenter l'évolution des contraintes sur les variables indicées de Gu sous la forme d'un graphe :



Pour $n \geq n_1$:

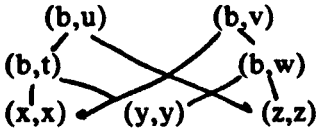
. Sg est la phase d'accélération

. Sm représente les contraintes de croisière (défini itérativement)

. $Sd_{/n}$, la phase de décélération

(Dans une stratégie de résolution en chaînage avant, les rôles de Sg et $Sd_{/n}$ sont inversées)

Exemple : Dans toutes les règles récursives classiques, ces effets de bord sont de taille presque nulle. Reprenons le problème précédent :

G = 

La simplification de $S_{[1,n]}(G)$ V { $u_i = v_{i-1}$ / $i \in [2,n]$ } nous fournit le système Sm(n) suivant :

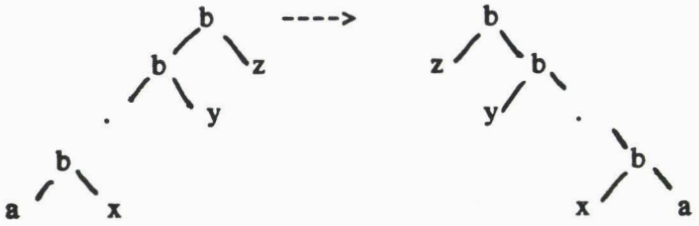
{ $u_i = b(t_i,w_{i-1})$, $t_i = b(t_{i+1},y_i)$, $v_i = b(t_{i+1},w_i)$, $w_i = b(y_i,w_{i-1})$, $x_i = t_{i+1}$, $z_i = w_{i-1}$ / $i \in [2,n-1]$ }

avec . $Sg = \{ x_1 = t_2 , u_1 = b(t_1,z_1) , t_1 = b(x_1,y_1) , v_1 = b(x_1,w_1) , w_1 = b(y_1,z_1) \}$

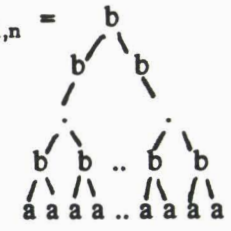
. $Sd = \{ z_n = w_{n-1} , u_n = b(t_n,z_n) , t_n = b(x_n,y_n) , v_n = b(x_n,w_n) , w_n = b(y_n,z_n) \}$

La solution exacte pour n réécritures : $Sg \cup S_{[2,n]}(Gu) \cup Sd_{/n}$.

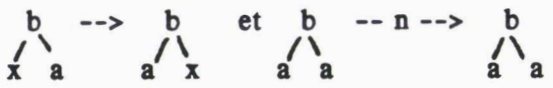
Dans certains cas extrêmes, les effets de bord peuvent être de tailles importantes, environ 2^k réécritures (si il y a k variables) :



Ici $t_{1,n}$ et $t_{n+1,n}$ sont égaux, fermés et constants pour $n \geq 2^k$

$t_{1,n} = t_{n+1,n}$ =  =  (DAG de hauteur k)

Par exemple pour k = 1, la règle est donc :



CONJECTURE.- Les effets de bord Sg et $Sd_{/n}$ sont toujours constants c-à-d : $n_1 = n_0$.

5-2) Décidabilité de l'arrêt d'un règle récursive.

THEOREME.- Une règle récursive simple engendre une infinité de réécritures avec application du test d'occurrence ssi :

1) g = β^0 V γ^{-1} existe (c-à-d β V γ existe)

2) g satisfait la Restriction sur les Pondérations de Boucles.

Démonstration : Nous avons étudié le cas sans test d'occurrence dans le chapitre 3 et nous savons qu'une règle récursive peut boucler ssi l'unifié au sens des Gops existe. Etudions donc le cas avec test d'occurrence.

1) L'étude peut se limiter à celle du système S_p(Gu).

En effet , nous savons que : $Sm(n) \leq \Sigma(n) \leq SM(n) = Sm(n+Kg+Kd)\text{->}Kg$

Donc $\forall\, n \geq n_1$, $\Sigma(n)$ satisfait le test d'occurrence ssi $\forall\, n \geq n_1$, Sm(n) le vérifie aussi.

Les équations du système Ind_{Pm}(Gu) peuvent être négligées, ce sont des égalités entre variables de la forme ($y_i = x_j$) où un seul de ces symboles de variable apparait dans Gu.

2) Nous savons enfin que pour P assez grand, S_p(Gu) satisfait le test d'occurrence ssi il n'existe pas de boucle de pondération nulle (mod Clas), c-à-d ssi Gu satisfait la Restriction sur les Pondérations de Boucles (R.P.B.).

Exemple : Soit la règle suivante : b (x , x) -> b (a(x) , x) ;

La séquence A d'arbres obtenus par n itérations de la règle est de la forme (sans test d'occurrence):

b(a^∞,a^∞) --> b(a^∞,a^∞) ... --> b(a^∞,a^∞)

Si l'on applique le test d'occurrence, la règle ne peut être utilisée qu'au plus une fois :

b(x_0,x_0) --> b(a(x_0),x_0) V b(x_1,x_1) n'existe pas

Cette séquence d'arbres peut s'exprimer sous la forme de l'unifié de deux Igops :

$I_{[1,n],[1,n]}(\beta^0)$ V $I_{[2,n+1],[1,n]}(\gamma^{-1}) = \{ (i,t_{i,n}) \;/\; i \in [1,n+1] \}$

avec β^0 = (b,r)^0 (x,x) et γ^{-1} = (b,u)$^{-1}$ (a,v) (x,x) (x,x)

β^0 V γ^{-1} = (b,r)^0 -1 -1 -1 (a,v) :1 qui ne vérifie pas le R.P.B. (sommet bouclant et congruent)

Cette règle ne boucle pas si l'on applique le test d'occurrence.

Sans test d'occurrence $I_{[1,n+1]}(\beta^0 \text{ V } \gamma^{-1}) = \Sigma(n) = \{ (1,b(a^\infty,a^\infty)),\ ..\ ,(n+1,b(a^\infty,a^\infty)) \}$

5-3) Forme et croissance des arbres $t_{1,n}$ et $t_{n+1,n}$.

On supposera, dans la suite, que l'on applique le test d'occurrence, c-à-d que le Igop Gu satisfait la R.P.B., il sera donc choisi parfait.
Nous savons que la solution exacte des contraintes, $\Sigma(n)$, peut s'exprimer sous la forme :

$$\Sigma(n) = Sg \cup Sm(n) \cup Sd_{/n}.$$

où Sm(n) , donc aussi $\Sigma(n)$ sont déterministes car le Igop unifié Gu est parfait.

PROPRIETE.- Soit $\sigma(n)$ la substitution caractérisant la plus petite séquence d'arbre $t_{i,n}$ vérifiant
$t_{1,n} = \beta_1.\sigma(n) \rightarrow t_{2,n} = \delta_1.\sigma(n) = \beta_2.\sigma(n) \rightarrow \ldots \quad \beta_n.\sigma(n) \rightarrow t_{n+1,n} = \delta_n.\sigma(n)$.
Pour tout $n \geq n_1$, $\sigma(n)$ peut être construite par application infinie de $\Sigma(n)$: $\sigma(n) = S(n)^{\infty}$.

Démonstration : Il nous faut pour cela de vérifier que les équations d'égalités de $\Sigma(n)$, pour éviter tout cycle , comme par exemple : x_i substituée par y_j et y_j par x_i :

1) Dans Sm(n), deux types d'équations d'égalités :

- les indirections,dans le sens, $y_i = x_{i+p}$ si y -p-> x , y indirigée et $x \in Var(Gu)$
c-à-d qu'on privilégie les variables de Gu dans $t_{i,j}$
- les équations congruentes, de même, on choisira le représentant canonique, c-à-d la variable indicée de plus faible indice $\in Pm$ (c-à-d $\geq 1+a_m$).

On peut remarquer que ces équations sont donc, dans Sm(n), exactement sous la forme souhaitée.

2) Les systèmes Sg et $Sd_{/n}$ sont constants donc , on peut supposer que :

$$Dom(Sg) \cap Var(Sd_{/n}) \ .$$

on peut privilégier dans Var(Sg) et $Var(Sd_{/n})$ les variables non indirigées. On peut d'ailleurs remarquer, pour les équations congruentes, $x_i = x_j$, on peut toujours choisir $j \in P_m$, car Sg->1 $\cup$ $Sd_{/n}$->-1 $\leq \Sigma(n)$, c-à-d si $x_i = x_k \in Sg$ avec i, $b \in [1,a_m]$ alors $(x_{i+1} = x_{k+1}) \leq Sg$

REMARQUE.- Les variables apparaissant dans $Var(\sigma(n))$ sont donc ici, de préférence, le variables indicées de Gu, représentants canoniques de leurs congruences .

Etudions maintenant l'évolution des arbres $t_{1,n}$ et $t_{n+1,n}$, et plus précisément la croissance de leurs branches.

REMARQUE : $\forall n \in N^*$, $t_{1,n} \leq t_{1,n+1}$ et $t_{n+1,n} \leq t_{n+2,n+1}$.

PROPRIETE.- Pour $n \geq n_1$, toute branche de $t_{1,n}$ étiquetée d'une variable d'indice $\leq n-b_m$ reste inchangée dans $t_{1,n+k}$ ($\forall k \in N$).

Démonstration : Montrons que cette branche est inchangée dans $t_{1,n+1}$.

$\Sigma(n) \leq \Sigma(n+1) = Sg \cup Sm(n+1) \cup Sd_{/n+1} = \Sigma(n) \cup S_{/n+1}$

avec $S_{/n+1} = S_{[n+1-bm,n+1-bm]}(Gu) \cup Ind_{[n+1-bm,n+1-bm]}(Gu) \cup Sd_{/n+1}$: constant par rapport à n c-à-d $Dom(S_{/n+1})$ est inclus dans $Vx[n+1-b_m,n+1]$.

Les modifications éventuelles de $t_{1,n+1}$ par rapport à $t_{1,n}$ dépendent du système constant $S_{/n+1}$. Toute branche étiquetée d'un symbole de fonction ou d'une variable d'indice $\in P_m$ n'est bien-sûr pas touchée par ses nouvelles contraintes.

CORROLAIRE.- Pour tout $n \geq n_1$, toute branche de $t_{1,n}$ n'ayant pas poussée strictement (en hauteur) dans $t_{1,n+2bm}$ est constante.
Toute branche de $t_{1,n}$ de longueur inférieure à $(n-n_1)/b_m$ est constante.

Démonstration.- Montrons que si une variable x_i de $t_{1,n}$ n'est pas modifiée pendant b_m réécritures supplémentaires, elle ne peut plus l'être après .

Soit x_i une variable de $t_{1,n}$:

. si $i \in [1,n-b_m]$, cette variable est non modifiée dans $t_{1,n+k}$ car aucune nouvelle contrainte sur cette variable indicée n'existe dans $\Sigma(n+k)$.

. si $i \in [n-b_m+1,n]$, elle peut être modifiée dans $t_{1,n+k}$:

si x est une variable de symbole non indirigée , il suffit d'attendre au pire b_m réécritures pour pouvoir utiliser l'équation $(x_i = z_j)$ avec z non indirigée,

si x n'est pas indirigée , alors soit s_x, le sommet tel que $Lab_2(s_x) = x$:

- soit $Lab_1(s_x) \in F$, $(x_i = t) \in S_{Pm}(Gu)$ et t non réduit à une variable : la branche "pousse".
- soit $Lab_1(s_x) \in V$ alors les seules équations possibles portant sur x_i sont des équations congruentes, x_i est substituée alors une fois pour toutes par son représentant canonique $x_{i \bmod Clas(x)}$ dans P_m. La branche ne poussera jamais plus dans $t_{1,n+k}$.

=> Pour tout $n \geq n_1$, toute branche de $t_{1,n}$ n'ayant pas poussée strictement (en hauteur) dans $t_{1,n+2bm}$ est constante

Toute branche croissante dépend bien-sûr d'une variable non indirigée. Soient x un symbole non indirigé et x_i, une variable de $t_{1,n}$, substituée par un arbre dans $t_{1,n+bm}$.

L'équation qui force cette croissance et bien-sûr $(x_i = t) \in S_{Pm}(Gu)$ où t ne contient que des symboles de variables non indirigés. La croissance des branches de t dépend toutes des symboles non indirigés.

=> Toute branche de $t_{1,n}$ n'ayant pas poussé en moyenne de 1 toutes les b_m réécritures est constante.

PROPRIETE.- Pour $n \geq n_1$, toute branche de $t_{n+1,n}$ étiquetée d'une variable d'indice $\geq 1+a_m$ reste inchangée par rapport à n et modulo la congruence dans $t_{n+1+k,n+k}$ ($\forall k \in N$).
Pour une variable indicée x_i de $t_{n+1,n}$ ($i \geq 1+a_m$), elle est remplacée dans $t_{n+1+k,n+k}$ par

- x_{i+k} **si** $i > n-b_m$
- $x_{i+k \bmod Clas(x)}$ **sinon**

Démonstration : De façon symétrique :

$\Sigma(n+1) = S_0 \cup \Sigma(n)->1$ où $S_0 = Sg \cup S_{[1+am,1+am]}(Gu) \cup Ind_{[1+am,1+am]}(Gu)$ système constant

Le passage de $t_{n+1,n}$ à $t_{n+2,n+1}$ peut donc se faire par trois opérations :

1) Incrémenter de 1 de tous les indices des variables de $t_{n+1,n}$ pour tenir compte de $\Sigma(n)->1$

2) Remplacer les variables indicées périodiques par leur représentant canonique.

3) Tenir compte des contraintes supplémentaires du système S_0 portant sur les variables d'indices appartenant à $[1,1+a_m]$.

Soit une feuille de $t_{n+1,n}$ d'étiquette x_i.

a) $i \in [1+a_m,n]$ et Clas(x) = 0. Aucune contrainte n'existe sur cette variable dans $\Sigma(n+1)$.

Dans $t_{n+2,n+1}$, cette même feuille sera étiquetée x_{i+1} $(=x_{i+1 \bmod Clas(x)})$.

b) $i \in [1+a_m,n-b_m]$ et Clas(x) <> 0. Les seules contraintes portant sur cette variable, sont les équations congruentes. Dans $t_{n+2,n+1}$, cette même feuille sera étiquetée $x_{i \bmod Clas(x)}$.

c) $i \in [n-b_m+1,n]$ et Clas(x) <> 0. Les équations congruentes sur x ne s'étendent pas à la variable x_i sinon elle aurait été substituée par x_j avec $j \in [1+a_m,n-b_m]$, en effet $Sd_{/n}->-1 \leq \Sigma(n)$, c-à-d si $(x_i = x_k) \in Sd_{/n}$ avec $i, k \in [n+1-b_m,n]$ alors $(x_{i-1} = x_{k-1}) \leq \Sigma(n)$, nous avons privilégié l'utilisation des variables d'indices $\in P_m$ dans $t_{n+1,n}$.

CORROLAIRE.- Soit c, le PPCM des congruences de Gu, alors toute variable indicée x_i de $t_{n+1,n}$ ($i \geq 1+a_m$), elle est remplacée dans $t_{n+1+c,n+c}$ par

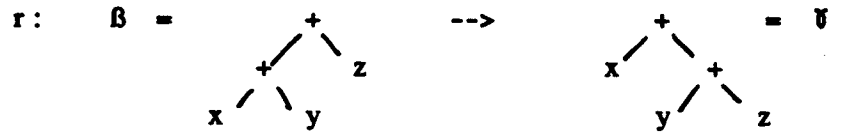
- x_{i+c} **si** $i > n-b_m$ **ou Clas(x) nul**
- x_i **sinon**

CORROLAIRE.- Pour tout $n \geq n_1$, toute branche de $t_{n+1,n}$ n'ayant pas poussée strictement (en hauteur) dans $t_{n+2am+1,n+2am}$ est constante (par rapport à n , modulo la congruence).
Toute branche de $t_{n+1,n}$ de longueur inférieure à $(n-n_1)/a_m$ est constante (. . .) .

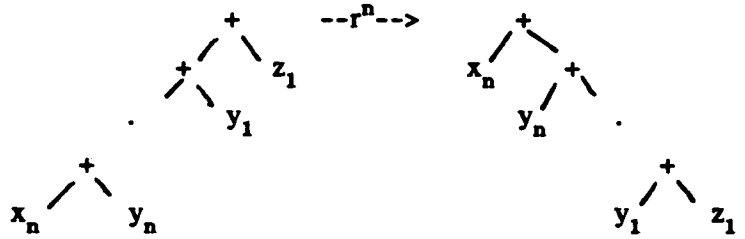
Démonstration : Identique à celle sur $t_{1,n}$

5-4) Règles consommatrices et productrices.

Pour obtenir des résultats fins sur l'arrêt d'une règle récursive et de sa complexité en nombre de réécritures, il est indispensable de pouvoir reconnaître les branches qui pousseront arbitrairement loin dans les arbres $t_{1,n}$ et de $t_{n+1,n}$. Ces branches sont associées en général aux boucles du Gop caractéristique de la règle . Reprenons l'exemple de l'associativité de l'addition :



Appliquer n fois cette règle, revient à appliquer la règle suivante :

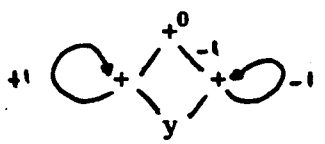


On voit dans ici que les seules branches arbitrairement longues dans $t_{1,n}$ et $t_{n+1,n}$ sont sur :

. les chemins 1^k dans $t_{1,n}$

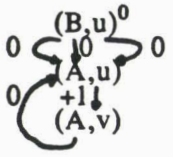
. les chemins 2^k dans $t_{n+1,n}$

ce qui correspond aux chemins de pondérations croissantes, respectivement décroissantes dans notre Gop caractéristique (boucles positives et négatives) :



Malheureusement dans le cas général, les choses sont beaucoup plus complexes. En effet, les effets de bord altèrent la lecture du Gop et certains boucles existant dans celui-ci ne sont pas effectives. Considérons par exemple la règle suivante :

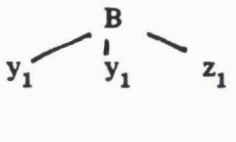
Son Gop caractéristique est :



Cette règle a la particularité d'être invariante sur les variables x_1 et y_n :

le m.g.u de

V

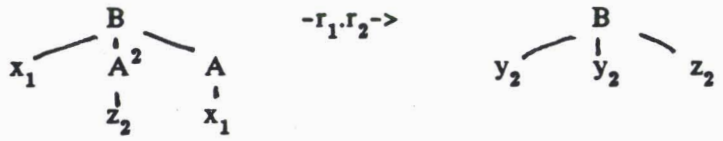


est $\quad . y_1 = A(z_2)$

$\quad . z_1 = A(z_2)$

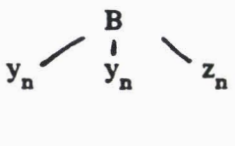
$\quad . x_2 = A(z_2)$

c-à-d



$\forall n$, $t_{1,n}$ =

et $t_{n+1,n}$ =



L'arbre $t_{1,n}$ ne possède qu'une branche poussante, or le Gop caractéristique contient une boucle positive qui n'est donc pas effective sur les branches 1 et 3 de $t_{1,n}$.

DEFINITION.- Une variable est dite positive ssi elle étiquette comme second Label un sommet positif.De même, une variable est dite négative ssi elle étiquette comme second Label un sommet négatif. On rappelle qu'un sommet s est positif (resp. négatif) ssi il existe dans le Gop une boucle élémentaire sur s de pondération positive (resp. négative).

DEFINITION.- Soit le système $\Sigma(n)$ des contraintes relatives à n applications de la règle récursive r, nous dirons que :

. $x_i \leq y_j$ si $\exists (x_i = t) \leq \Sigma(n)$ telle que $y_j \in Var(t)$.

. $x_i < y_j$ si $\exists (x_i = t) \leq \Sigma(n)$ telle que $y_j \in Var(t)$ et t non réduit à y_j.

Remarque : La relation < est une relation d'ordre si le système $\Sigma(n)$ satisfait le test d'occurrence.

DEFINITION.- Soit Mg un mot infini de N^{∞} , Mg est dit chemin gauche infini de la règle r ssi pour tout facteur gauche m (fini) de Mg, il existe n tel que m est un chemin de l'arbre $t_{1,n}$:

Mg chemin gauche infini si $\forall$ m facteur gauche de Mg , $\exists$ n tel que $m \in Dom(t_{1,n})$

De même, soit Md un mot infini de N^{∞} , Md est dit chemin droit infini de r ssi pour tout facteur gauche m (fini) de Md, il existe n tel que m soit un chemin de l'arbre $t_{n+1,n}$:

Md chemin droit infini si $\forall$ m facteur gauche de Md , $\exists$ n tel que $m \in Dom(t_{n+1,n})$

REMARQUE. - La croissance linéaire des branches poussantes des arbres $t_{1,n}$ et $t_{n+1,n}$ nous assure que le n vérifiant $m \in Dom(t_{1,n})$ ou $Dom(t_{n+1,n})$ est borné linéairement par la longueur de m

$\exists$ f = ax+b , telle que , $\forall$ m , et $n \geq a$*longueur(m)+b , $m \in Dom(t_{1,n})$ ou $Dom(t_{n+1,n})$.

PROPRIETE.- Tout chemin infini gauche ou droit d'une règle est un chemin infini de $A_Z(G,k)$ où k est quelconque et G est le Igop caractéristique de r. L'inverse est faux (Exemple précédent).

Démonstration : De façon immédiate, $A_Z(G,k)$ est le plus petit arbre se réécrivant en un arbre équivalent. Donc $A_Z(G,k) \geq t_{1,n}$ et $t_{n+1,n}$.

DEFINITION. - Une règle récursive est dite *consommatrice*, respectivement *productrice* si elle possède des chemins infinis gauches, respectivement droits.

Exemple : La règle de l'associativité de l'addition est consommatrice et productrice.

PROPRIETE.- Il est décidable de savoir si une règle est consommatrice, respectivement productrice.

Démonstration : L'idée est d'observer à travers les effets de bords constants les variables positives ou négatives "accessibles".

Nous savons que pour tout $n \geq n_1$, toute branche de $t_{1,n}$ étiquetée d'une variable d'indice inférieure ou égale à $n-b_m$ est constante. Plaçons nous dans le cas d'une branche étiquetée d'une variable d'indice supérieure à $n-b_m$.

Peut-on dire si cette branche peut devenir arbitrairement longue quand n croit ?

Oui, on peut le dire , si $n \geq 1 + a_m + b_m + K*Maxpond + MaxInd$. K désigne ici le nombre de noeuds du Gop caractéristique $\beta^0 \; V \; \nu^{-1}$, MaxPond est le Max des valeurs absolues de ses pondérations, enfin MaxInd est le Max des valeurs absolues des indirections dans l'unification. Nous poserons p_b, la constante égale à K*Maxpond + MaxInd.

Soit donc une branche de $t_{1,n}$ étiquetée en feuille par une variable x_i avec $i \in]n-b_m,n]$. Cette variable x_i sera substituée par un arbre arbitrairement grand en hauteur, ssi on peut atteindre à partir du sommet s_x un sommet positif s_y dans le Gop caractéristique.

. Si un tel chemin existe, on peut construire un chemin de longueur inférieure ou égale à K qui inclus une boucle élémentaire positive sur le sommet s_y . Comme $n \geq 1 + a_m + b_m + p_b$, on peut construire aussi à partir du système $\Sigma(n+p_b)$, une inéquation de la forme :

$$x_i < \; .. < y_j < .. < y_k \quad \text{et} \quad j < k$$

car tous les sommes des pondérations intermédiaires appartiennent à $[1+a_m , n+p_b -b_m]$.

De même, dans $\Sigma(n+p_b+k-j)$, on a l'inégalité :

$$x_i < \; .. < y_j < .. < y_k < y_{2k-j}$$

Par récurrence sur n, la variable x_i est donc substituée par un arbre arbitrairement grand en hauteur quand n croit.

. Inversement, si il n'existe aucun chemin permettant d'atteindre un sommet positif à partir du sommet s_x, tout chemin dans le Gop de racine s_x a une pondération sommante qui est majorée par p_b, c-à-d que les variables z_k indicées telles que " $x_i < z_k$ " sont en nombre fini indépendamment du nombre de réécritures. Les inégalités de la forme :

$$x_i < \; .. < v_j < .. < z_k$$

sont de longueur bornée. La variable x_i sera substituée par un arbre constant .

De la même façon, on montre que dans $t_{n+1,n}$, une variable pourra être substituée par un arbre arbitrairement grand ssi :

. $i \in [1,a_m]$

. un sommet négatif est accessible à partir du sommet s_x dans le Gop caractéristique.

REMARQUE.- Une condition nécessaire pour qu'une règle soit

. consommatrice est que son Gop caractéristique possède des sommets positifs

. productrice est qu'il contienne des sommets négatifs

5-5) Décidabilité de la propriété de terminaison pour les termes clos.

RAPPEL.- Une règle r satisfait la propriété de terminaison ssi tout terme fini et clos ne peut être réécrit que finiment pas la règle r : $\forall$ T clos fini , T --r--> finiment

THEOREME.- Une règle r satisfait la propriété de terminaison pour tout terme clos fini, ssi elle est consommatrice (ce qui est décidable) .

Démonstration. : En effet si une règle est consommatrice, alors quelque soit la hauteur du terme clos T, il existe un n linéairement dépendant de cette hauteur telle que l'une des branches de $t_{1,n}$ soit de longueur supérieure de celle correspondant dans le terme T. En d'autres termes, pour n assez grand, l'unification de T et $t_{1,n}$ n'existe pas.

Inversement si la règle n'est pas consommatrice, il existe un nombre de réécritures minimal et calculable tel que pour tout n supérieur à ce nombre, l'arbre $t_{1,n}$ est constant, tout instanciation close de cet arbre peut alors être réécrit infiniment par la règle r.

Exemples : La règle de l'associativité de l'addition est consommatrice, elle contient un chemin infini gauche 1^{∞}. Tout terme clos ne peut être réécrit que finiment pas cette règle, au plus la longueur de la branche la plus à gauche.

De même, la règle de définition des entiers est consommatrice :

Succ (x) -> x ; Igop associé : $(Succ,x)^0$ +1

$t_{1,n} = Succ^n (x_{n+1})$ et $t_{n+1,n} = x_{n+1}$

Tout terme clos ne pourra être réécrit que finiment par cette règle.

Si l'on inverser cette règle de réécriture :

x -> Succ (x) ;

on obtient une règle productrice, mais non consommatrice, d'ailleurs n'importe quel terme clos se réécrit infiniment par cette règle.

De même, le prédicat Frère peut engendrer une réécriture infinie de certains termes clos :

FRERE (x , y) -> FRERE (y , x) ;

Cette règle n'est pas consommatrice, son Igop ne contient d'ailleurs aucune boucle.

5-6) Résolution du TQ pour T et α linéaires.

Etudions le programme PROLOG que nous avons appelé le TANT QUE, dans le cas simple où T et α sont linéaires :

$P(\alpha)$ ->;

$P(\beta)$ -> $P(\gamma)$;

$P(T)$?

où α, β, γ et T sont des arbres finis de M(F,V), et T, α sont linéaires (une occurrence de chaque variable dans T et α).

CONVENTION : On note . σ_n (T,α) et $(t_{1,n}, t_{n+1,n})$.
. r_n , **la solution à prés n réécritures**

CARACTERISATION 1.- La recherche d'une solution avec n utilisations de la régle récursive peut s'exprimer sous la forme l'unification de trois systèmes :

$$\text{Solution}(n) = \Sigma(n)\ V\ (u_1 = T)\ V\ (v_n = \alpha)$$
$$= (\ Sg \cup Sm(n) \cup Sd_{/n}\ )\ V\ \{\ (u_1 = T)\ ,\ (v_n = \alpha)\ \}$$

CARACTERISATION 2.- Si l'on note σ_n , la substitution vérifiant :

$$(T\ ,\ \alpha)\ V\ (t_{1,n}\ ,\ t_{n+1,n}) = (t_{1,n}\ ,\ t_{n+1,n})\ .\ \sigma_n$$

La solution r_n après n utilisations de la règle récursive est caractérisée par σ_n :

r_n **existe ssi** σ_n **existe**

et $r_n = t_{1,n}\ .\ \sigma_n$ $\quad (\ T \leq r_n = t_{1,n}.\sigma_n \ \text{--n-->}\ t_{n+1,n}.\sigma_n \geq \alpha\ )$

CARACTERISATION 3.- Il existe $n_2 \in N$, dépendant linéairement de la hauteur de T et α, telle que pour tout $n \geq n_2$, l'existence et la forme de σ_n est caractérisée par celles de trois substitutions "constantes" : $\sigma_n = \sigma_g\ V\ (\sigma_{d/Clas} \cup \sigma_{d/n}\ )$

où σ_g est définie par $t_{1,n}\ V\ T = t_{1,n}\ .\ \sigma_g$ avec $Dom(\sigma_g) \subset Vx[1,n_2]$, $Var(\sigma_g) \subset Var(T)$

$\sigma_{d/Clas}$ et $\sigma_{d/n}$ sont définies par $t_{n+1,n}\ V\ \alpha = t_{n+1,n}\ .\ (\sigma_{d/Clas} \cup \sigma_{d/n})$

$\sigma_{d/Clas}$ est constante modulo les congruences de Gu et $\sigma_{d/n}$ est constante par rapport à n.

Ces substitutions portent sur des variables différentes : $Dom(\sigma_{d/Clas}) \cap Dom(\sigma_{d/n}) = \emptyset$.

$Dom(\sigma_{d/Clas}) \subset Vx[1+a_m, 1a_m+c[$, $Var(\sigma_{d/Clas}) \subset Var(\alpha)$

$Dom(\sigma_{d/n}) \subset Vx[n-n_2, n]$, $Var(\sigma_{d/n}) \subset Var(\alpha)$

Démonstration : La caractérisation 2 repose sur l'unification de $(t_{1,n} , t_{n+1,n})$ et (T , α).
Or il existe $n_2 \in N$, dépendant linéairement de la hauteur de T et α telle que pour $n \geq n_2$, les branches de $t_{1,n}$ (resp. $t_{n+1,n}$) plus petites que celles correspondant dans T (resp. α), sont constantes : $\exists$ a, b, c constantes telles que : $\forall$ T et α , $n_2 = a^*Hauteur(T) + b^*Hauteur(\alpha) + c$.
(Conséquence immédiate des propriétés sur la croissance de $t_{1,n}$ et $t_{n+1,n}$)
Comme T est linéaire, l'existence d'une solution nécessite celle d'une substitution σ_g constante telle que : $\forall n \geq n_2$, $t_{1,n} \vee T = t_{1,n}.\sigma_g$
σ_g porte sur les feuilles des branches constantes de $t_{1,n}$ plus courtes que celles de T.

$\sigma_g = \{ (x_i = t) / x$ est une variable libre de $t_{1,n}$ et t est un sous-arbre de T $\}$

c-à-d $Dom(\sigma_g) \subset Vx[1,n_2]$, $Var(\sigma_g) \subset Var(T)$

De même, α est linéaire, l'existence d'une solution nécessite celle de deux substitutions $\sigma_{d/Clas}$ et $\sigma_{d/n}$ "constantes" telles que : $\forall n \geq n_2$, $t_{n+1,n} \vee \alpha = t_{n+1,n}.(\sigma_{d/Clas} \cup \sigma_{d/n})$

- $\sigma_{d/Clas}$ porte sur les feuilles des branches constantes (modulo les congruences) de $t_{n+1,n}$ plus courtes que celles de α, la feuille est étiquetée d'une variable périodique $\in Vx[1+a_m,1+a_m+c[$:

$\sigma_d = \{ (x_i = t)$ / x est une variable congruente de $t_{n+1,n}$, x_i l'un de ses représentants canoniques et t est un sous-arbre de $\alpha \}$

c-à-d $Dom(\sigma_{d/Clas}) \subset Vx[1+a_m,1a_m+c[$, $Var(\sigma_{d/Clas}) \subset Var(\alpha)$

- $\sigma_{d/n}$ porte, par contre, sur le même type de feuilles des branches constantes par rapport à n :

$\sigma_{d/n} = \{ (x_{n-i} = t) / x_{n-i}$ est une variable libre de $t_{n+1,n}$ et t est un sous-arbre de $\alpha \}$

c-à-d $Dom(\sigma_{d/n}) \subset Vx[n-n_2,n]$, $Var(\sigma_{d/n}) \subset Var(\alpha)$

L'existence et la forme d'une solution r_n dépend de : $\sigma_n = \sigma_g \vee (\sigma_{d/Clas} \cup \sigma_{d/n})$ et $r_n = t_{1,n}.\sigma_n$.

CARACTERISATION 4.- Il existe n_3 linéairement dépendant de la hauteur de T, telle que pour tous $n \geq n_3$ et équivalents (modulo PPCM des congruences de Gu), l'existence et la forme d'une solution est liée celles de deux substitutions

- **$\sigma_{/1}$ constante**
- **$\sigma_{/n}$ constante par rapport à n**

telle que $\sigma_n = (\sigma_{/1} \cup \sigma_{/n})$ où $Dom(\sigma_{/1}) \cap Dom(\sigma_{/n}) = \emptyset$ et $Dom(\sigma_n) \cap Var(\sigma_n) = \emptyset$.

Démonstration : La caractérisation 3 repose sur les systèmes σ_g , $\sigma_{d/Clas}$ et $\sigma_{d/n}$.
Pour les $n \geq n_2$ et l'existence d'une solution est dépendant de l'unification des substitutions σ_g et $(\sigma_{d/Clas} \cup \sigma_{d/n})$.
Pour tout $n > 2^*n_2$, l'existence de r_n dépendant de $\sigma_g \vee \sigma_{d/Clas}$ car la troisième substitution porte sur des variables différentes de celles de σ_g et $\sigma_{d/Clas}$.

Comme $\sigma_{d/Clas}$ est une substitution périodique (au pire modulo PPCM des congruences de Gu), pour tous les $n > 2^*n_2$ et équivalents (mod c), la forme et l'existence de σ_n dépend de l'unification de deux systèmes constants σ_g et σ_d .
Posons donc $(\sigma_g \vee \sigma_d) = (\sigma_g \cup \sigma_d \cup \mu)$ où $Dom(\mu) \cup Var(\mu) \subset Var(T) \cup Var(\alpha)$.
$\Rightarrow \sigma_n = (\sigma_g \vee \sigma_d) \cup \sigma_{d/n}.\mu$ $\Rightarrow \sigma_{/1} = \sigma_g \vee \sigma_d$ constant et $\sigma_{/n} = \sigma_{d/n}.\mu$ constant / n

CORROLAIRE.- $\forall n \geq n_3$, r_n **existe ssi** r_{n+c} **existe** **(c : PPCM des congruences de Gu) .**

CONVENTIONS.- On notera

- $R_{k,n}$ **: l'ensemble des solutions ou réponses que l'on obtient après i réécritures** $\forall i \in [k,n]$

$$R_{k,n} = \{ r_i \text{ (s'il existe) } / \forall i \in [k,n] \}$$

- **R : l'ensemble de toutes les solutions ,** $R = R_{1,+\infty}$
- R_{min} **: le plus petit thermal de R, c-à-d l'ensemble des réponses les plus générales.**

$\forall r_j \in R$, $\exists r_i \in R_{min}$ **telle que** $r_i \geq r_j$.
$\nexists r_i$ **et** $r_j \in R_{min}$, **telles que** $r_i \geq r_j$

CORROLAIRE.- L'existence d'une solution est décidable en un nombre de réécritures dépendant linéairement de la hauteur de la question.
R est non vide ssi $R_{1,n3+c-1}$ **l'est (c : PPCM des congruences de Gu).**

Démonstration : En application du Corollaire : $\forall n \geq n_3$, r_n existe ssi r_{n+c} existe.
Il nous suffit donc de rechercher toutes les solutions r_n , avec $n \leq n_3+c-1$

CORROLAIRE.- L'existence d'un nombre infini de réponses est décidable en un nombre de réécritures dépendant linéairement de la hauteur de la question :
R est infini ssi $R_{n3,n3+c-1}$ **est non vide (c : PPCM des congruences)**

Démonstration : En application du Corollaire, il existe donc une infinité de réponses à la question P(T) si une solution est détectée entre n_3 et n_3+c-1 utilisations de la régle récursive ; deux réponses obtenues en un nombre de réécritures différents étant, ici , considérées comme différentes , même si elles sont comparables, équivalentes , voire égales.

THEOREME : Sous l'hypothèse où α et T sont linéaires, deux propriétés suivantes :

1) l'existence de solutions,

2) l'existence d'un nombre fini de solutions,

sont décidables en un nombre de réécritures linéairement dépendant de la hauteur de la question.

PROPRIETE.- Pour une question T close, si la règle est consommatrice, le programme Tant Que n'admet qu'un ensemble fini de solutions, c-à-d s'arrête après un nombre de réécritures borné linéairement par la hauteur de la question T.

Démonstration : Découle directement de la propriété de terminaison sur les termes clos.

PROPRIETE. - Si la règle récursive est ni consommatrice, ni productrice, elle est périodique au bout d'un nombre borné de réécritures. Le programme Tant Que est de complexité constante, c-à-d qu'il existe un programme, ne comportant que des faits, qui lui est équivalent .

Exemple :

```
FRERE (Jean , Jacques )  ->  ;
FRERE ( x , y )  ->  FRERE ( y , x )  ;
```

équivalent à

```
FRERE ( Jean , Jacques )  ->  ;
FRERE ( Jacques , Jean )  ->  ;
```

Démonstration : Nous savons que si la règle est ni consommatrice, ni productrice, cela signifie qu'il existe une constante n à partir laquelle les arbres $t_{1,n}$ et $t_{n+1,n}$ sont de hauteur constante. De plus . $t_{1,n}$ est totalement invariant quand n croit ,

. les variables de $t_{n+1,n}$ sont soient périodiques, soient d'indice constant par rapport à n

En d'autres termes, à partir d'un certain rang cette règle devient purement périodique, l'application de n réécritures étant équivalent à celle de n+c réécritures (c : période de la règle).
Il suffit de déterminer une fois pour toutes les solutions que l'on peut obtenir avant ce rang à le question la plus générale.

DEFINITION.- Un arbre t sera dit clos sur un chemin infini m si la feuille dans t correspondant à l'un des facteurs gauches de m n'est pas étiquetée d'une variable.

PROPRIETE.- Si le fait est clos sur l'un des chemins infinis droits de la règle r, le Tant Que est de complexité constante.

Exemple : INF-QUATRE ($Succ^3$(Zéro)) -> ; [3 < 4]

INF-QUATRE (x) -> INF-QUATRE (Succ(x)) ; [x+1 < 4 SI x < 4]

Le chemin 1^∞ est un chemin droit infini : x --r^n--> $Succ^n$(x).

Quelque soit la question posée, le nombre de réécritures possible pour l'obtention d'une solution est bornée ; ce programme est équivalent à :

INF-QUATRE ($Succ^3$(Zéro)) -> ; [3 < 4]

INF-QUATRE ($Succ^2$(Zéro)) -> ; [2 < 4]

INF-QUATRE ($Succ^1$(Zéro)) -> ; [1 < 4]

INF-QUATRE (Zéro) -> ; [0 < 4]

PROPRIETE.- Si la question est close sur l'un des chemins infinis gauches de la règle r, la résolution classique PROLOG s'arrête ; les chemins de preuves sont en nombre fini et obtenus après un nombre de réécritures linéairement dépendant de la longueur de la branche close.

Exemple : INF (Zéro, Succ(x)) -> ; [3 < 4]

INF (Succ(x) , Succ(y)) -> INF (x , y) ; [x < y SI x+1 < y+1]

Les chemins 1^∞ et 2.1^∞ sont des chemins infinis gauches de r :

INF($Succ^n$(x) , $Succ^n$(y)) --r^n--> INF (x , y)

Pour toute question dont le premier ou le second argument est connue, le programme s'arrête, il n'existe qu'un nombre fini de chemins de preuve dans l'arbre SLD à une profondeur linéairement dépendant de la longueur de la branche close .

5-7) EXEMPLES.

5-7-1) DEFINITION DES ENTIERS.

Soit le programme PROLOG suivant :

Entier(Zéro) ->;

Entier(Succ (x)) -> Entier (x)

Nous savons que la séquence A d'arbres obtenus par n itérations de la règle est de la forme :

$Succ^n(x)$ -1^e-> $Succ^{n-1}(x)$ ... -(n-1)^e-> Succ(x) -n^e-> x

On peut exprimer A sous forme d'unifié de 2 Igops :

$\beta^0 = (Succ,u)^0_{(x,x)}$ et $\gamma^{-1} = (x,x)^{-1}$

avec $\{ (i,t_{i,n}) / i \in [1,n+1] \} = I_{[1,n+1],[1,n]} (\beta^0) \vee I_{[1,n+1],[1,n]} (\gamma^{-1})$

$= \{ (1,Succ^n(x)), (2,Succ^{n-1}(x)), .. ,(n-1 ,Succ(x)), (n,x) \}$

Le Igop unifié est : $\beta^0 \vee \gamma^{-1} = (Succ,u)^0_{+1}$

Le Igop Unifié satisfait la Restriction sur les Pondérations de Boucles, cette règle récursive peut donc boucler (vrai ou sans test d'occurrence).

A est obtenu ici, de façon exacte, par interprétation du Igop unifié sur [1,n+1] et [1,n] :

$A = I_{[1,n+1],[1,n]} (\beta^0 \vee \gamma^{-1})$

En termes de systèmes d'équations, la résolution du TQ s'exprime :

$\{ (u_i = Succ(x_i)) / i \in [1,n] \} \vee \{ (u_i = x_{i-1}) / i \in [2,n] \} \vee (u_1 = T) \vee (x_n = Zéro)$.

1) On peut minorer l'unifié de deux premiers systèmes par :

$Sm(n) = \{ (u_i = Succ(u_{i+1}) / i \in [2,n-1] \} \cup \{ (x_i = u_{i+1}) / i \in [2,n-1] \}$

$\Sigma(n) = Sg \cup Sm(n) \cup Sd/n$ avec $Sg = \{ (u_1 = Succ(u_2) , (x_1 = u_2) \}$

et $Sd = \{ (u_n = Succ(x_n) \}$

Pour $n \geq 3$ (= n_1), on peut donc exprimer de façon exacte les contraintes liées à n itérations de la régle récursive . La résolution du TQ s'exprime : $\Sigma(n) \vee (u_1 = T) \vee (v_n = Frère(Jean,Jacques))$.

2) $t_{1,n} = Succ^n(x_n)$ --n--> $t_{n+1,n} = x_n$

La seule branche de $t_{1,n}$ est une branche poussante et pousse de hauteur 1 à chaque réécriture, et celle de $t_{n+1,n}$ est constante dés que $n \geq 1$.

Ici toutes les congruences sont nulles : $c = 1$ (PPCM des congruences).

Caractérisation 4 : $\forall n \geq n_s = Hauteur(T)+1$, $\sigma_n = \sigma_{/1} \vee \sigma_{/n}$

l'existence de $\sigma_{/1}$ dépend de T et $\sigma_{/n} = \{ (x_n = Zéro) \}$.

Après Hauteur(T)+1 réécritures, on sait s'il existe des solutions et si elle sont en nombre fini.

Soient les questions suivantes :

- T1 = y ($\in$ V) -- arbre de hauteur 0

On calcule donc r_0 et r_1 , r_0 = Zéro et r_1 = Succ(Zéro)

Il existe une infinité de réponses car r_1 existe (=> $r_{1+k*c} = r_n$ existe)

- T2 = $Succ^5$(Zéro) -- arbre de hauteur 5

On recherche les solutions en moins de 6 réécritures : seule $r_5 = Succ^5$(Zéro) existe

C'est la seule solution car r_6 n'existe pas (=> r_n n'existe pas pour n > 6).

- T3 = $Succ^2$(Nil) -- arbre de hauteur 2

Aucune solution n'existe, car aucune n'est obtenue avant 3 réécritures.

5-7-2) COMMUTATIVITE.

Soit le programme PROLOG suivant :

Frère (Jean,Jacques) ->;

Frère (x,y) -> Frère (y,x);

Nous savons que la séquence A d'arbres obtenus par n itérations de la règle est de la forme :

Frère(x,y) -1^e-> Frère(y,x) .. $-2p^e$-> Frère(x,y) $-(2p+1)^e$-> Frère(y,x) ..

c-à-d $\forall$ 2p ou 2p+1 $\leq$ n , $t_{2p,n}$ = Frère(x,y) et $t_{2p+1,n}$ = Frère(y,x) .

Cette séquence d'arbres peut s'exprimer sous la forme de l'unifié de deux Igops : $I_{[1,n+1],[1,n]}(\beta^0)$ V $I_{[1,n+1],[1,n]}(\gamma^{-1})$

L'unifié β^0 V γ^{-1} : (Frère,u)^0 V (Frère,v)$^{-1}$ = (Frère,u)^0 x de congruence 2

(x,x) (y,y) (y,y) (x,x) $0\{(x,x)\}+1$

Le Igop Unifié satisfait la Restriction sur les Pondérations de Boucles, cette règle récursive peut donc boucler (vrai ou sans test d'occurrence).

A = $I_{[1,n+1],[1,n]}(\beta^0 \text{ V } \gamma^{-1})$

= { (1,Frère (x_1,x_2)), .. ,(n+1,Frère(x_1,x_2)) } si n pair

= { (1,Frère (x_1,x_2)), .. ,(n+1,Frère(x_2,x_1)) } si n impair

En termes de systèmes d'équations, la résolution du TQ s'exprime sous la forme :

$\{ (u_i = \text{Frère}(x_i,y_i)), (v_i = \text{Frère}(y_i,x_i)) / i \in [1,n] \}$ V $\{ (u_i = v_{i-1}) / i \in [2,n] \}$

V $(u_1 = T)$ V $(v_n = \text{Frère(Jean,Jacques)})$.

1) On peut minorer l'unifié de deux premiers systèmes par :

$Sm(n) = \{ (u_i = \text{Frère}(x_i,x_{i+1})) , (v_i = \text{Frère}(x_{i+1},x_i)) , (x_i = x_{i \bmod 2}) / i \in [2,n-1] \}$

$\cup \{ (y_i = x_{i+1}) / i \in [2,n-1] \}$

$\Sigma(n) = Sg \cup Sm(n) \cup Sd/n$

avec $Sg = \{ (u_1 = \text{Frère}(x_3,x_2) , (v_1 = \text{Frère}(x_2,x_3)) , (x_1 = x_3) , (y_1 = x_2) \}$

et $Sd = \{ (u_n = \text{Frère}(x_3,x_2) , v_n = \text{Frère}(x_2,x_3) , (x_n = x_3) , (y_n = x_2) \}$ (n pair)

ou $Sd = \{ (u_n = \text{Frère}(x_2,x_3) , v_n = \text{Frère}(x_3,x_2) , (x_n = x_2) , (y_n = x_3) \}$ (n impair)

Pour $n \geq 3$ ($= n_1$), on peut donc exprimer de façon exacte les contraintes liées à n itérations de la régle récursive . La résolution du TQ s'exprime : $\Sigma(n)$ V $(u_1 = T)$ V $(v_n = \text{Frère(Jean,Jacques)})$.

2) $t_{1,n}$ = Frère(x_3,x_2) --n--> $t_{n+1,n}$ = soit Frère(x_3,x_2) (n pair) , soit Frère(x_2,x_3) (n impair)
Les branches de $t_{1,n}$ et de $t_{n+1,n}$ sont constantes modulo la congruence dés que $n \geq 1$.
Une seule congruence, ici de valeur 2 : c = 2 (PPCM des congruences).
Caractérisation 3 : $\forall n \geq 1$, $\sigma_n = \sigma_g \vee \sigma_{d/Clas}$
σ_g dépend de T et $\sigma_{d/Clas}$ = { ($x_{n \bmod 2}$ = Jean) , ($x_{n+1 \bmod 2}$ = Jacques) }
Après une réécriture, on sait s'il existe des solutions et elles sont en nombre fini .

Soient les questions :

- Frère(w,z) ? -- $r_0 = r_2$ = Frère(Jean,Jacques) et $r_1 = r_3$ = Frère(Jacques,Jean)

σ_{2p} = { (x_2 = Jacques) , (x_3 = Jean) } , σ_{2p+1} = { (x_2 = Jean) , (x_3 = Jacques) }
Il existe une infinité de réponses car r_0 existe , donc r_{2p} aussi , de même r_1 et r_{2p1+} existent)

- Frère(Jacques,z) ? -- $r_1 = r_3$ = Frère(Jacques,Jean) ;

Il existe une infinité de solutions car r_1 existe donc aussi r_{2p+1} .

- Frère(Alain,z) ? -- r_0, r_1 n'existent pas .

Il n'existe donc aucune solution.

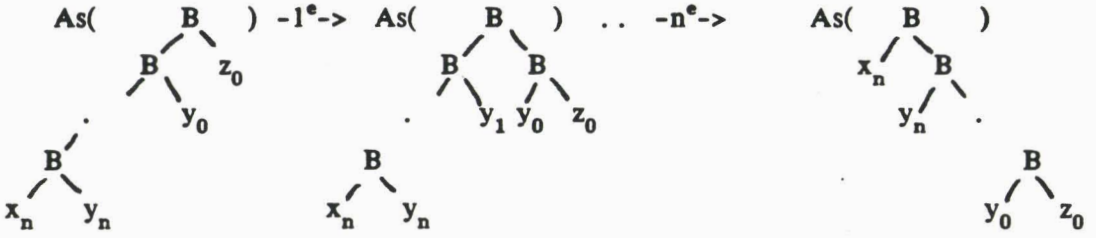
Cet exemple met en relief une règle périodique , le nombre de réécritures nécessaires est indépendant de la question, ce programme est équivalent à un programme non récursif.

5-7-3) ASSOCIATIVITE.

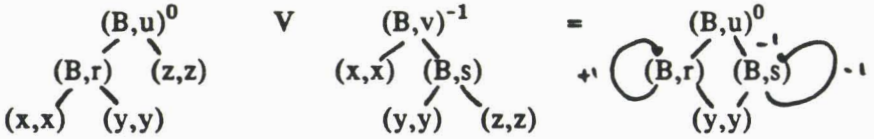
As (B(C,B(C,D))) ->;

As (B(B(x,y),z)) -> As(B(x,B(y,z))) ;

Nous savons que la séquence A d'arbres obtenus par n itérations de la règle est de la forme :



L'unifié β^0 V γ^{-1} :



Le Igop Unifié satisfait la Restriction sur les Pondérations de Boucles, cette règle récursive peut donc boucler (vrai ou sans test d'occurrence).

$A = I_{[1,n],[1,n]}(\beta^0) \text{ V } I_{[2,n+1],[1,n]}(\gamma^{-1})$ minoré par $I_{[1,n+1],[1,n]}(\beta^0 \text{ V } \gamma^{-1})$

majoré par $I_{[1,n+1],[1,n+1]}(\beta^0 \text{ V } \gamma^{-1})$

$I_{[1,n+1],[1,n]}(\beta^0 \text{ V } \gamma^{-1}) =$ { (1, B) , . . . , (n, B) (n+1, u_{n+1}) }

$I_{[1,n+1],[1,n+1]}(\beta^0 \text{ V } \gamma^{-1}) =$ { (1, B) , . . . , (n+1, B) }

En termes de systèmes d'équations, la résolution du TQ s'exprime sous la forme :

$S_{[1,n]}(G) \text{ V } \{ (u_i = v_{i-1}) / \forall i \in [2,n] \} \text{ V } (u_1 = T) \text{ V } (v_n = B(C,B(C,D)))$

où $S_{[1,n]}(G) = \{ (u_i = B(r_i,z_i)) , (r_i = B(x_i,y_i)) , (v_i = B(x_i,s_i)) , (s_i = B(y_i,z_i) / \forall i \in [1,n] \}$

1) On peut minorer l'unifié de deux premiers systèmes par : $Sm(n) = S_{[2,n-1]}(Gu) \cup Ind_{[2,n-1]}(Gu)$

où $S_{[2,n-1]}(Gu) = \{ (u_i = B(r_i,s_{i-1})) , (r_i = B(r_{i+1},y_i)) , (s_i = B(y_i,s_{i-1}) \quad / \forall i \in [2,n-1] \}$

et $Ind_{[2,n-1]}(Gu) = \{ (v_i = u_{i+1}) , (z_i = s_{i-1}) , (x_i = r_{i+1}) \quad / \forall i \in [2,n-1] \}$

$\Sigma(n) = Sg \cup Sm(n) \cup Sd/n$

avec $Sg = \{ (u_1 = B(r_1,z_1)) , (r_1 = B(r_2,y_1)) , (s_1 = B(y_1,z_1) , (v_1 = u_2) \}$

et $Sd = \{ (u_n = B(r_n,s_{n-1})) , (r_n = B(x_n,y_n)) , (s_n = B(y_n,s_{n-1}) , (v_n = B(x_n,s_n) \}$

Pour $n \geq 3$ ($= n_1$), on peut donc exprimer de façon exacte les contraintes liées à n itérations de la régle récursive . La résolution du TQ s'exprime : $\Sigma(n) \; V \; (u_1 = T) \; V \; (v_n = B(C,B(C,D)))$.

2) $t_{1,n} = B(B(.. B(B(x_n,y_n),y_{n-1}), .. ,y_1),z_1)$ --n--> $t_{n+1,n} = B(x_n,B(y_n,B(y_{n-1}, .. B(y_1,z_1))))$

La seule branche croissante de $t_{1,n}$ est celle le plus à gauche , elle croit d'un à chaque réécriture.

De même , la seule branche poussante de $t_{n+1,n}$ est celle le plus à droite, elle croit aussi d'un à chaque réécriture. Les autres, dans $t_{1,n}$ (resp. $t_{n+1,n}$) sont constantes.

Toutes les congruences sont nulles : $c = 1$ (PPCM des congruences).

Caractérisation 4 : il n'existe pas pour $n \geq 2$, $\sigma_{/n}$ tel que $\sigma_n = \sigma_{/1} \; V \; \sigma_{/n}$.

En effet , $t_{n+1,n} \; V \; \alpha$ n'existe pas pour $n \geq 2$.

Quelque soit la question, après deux réécritures, on sait s'il existe des solutions ou non , seule r_0 et r_1 peuvent exister.

Soient les questions suivantes :

- T1 = w ($\in$ V) : $r_0 = B(C,B(C,D))$, $r_1 = B(B(C,C),D)$.
- T2 = B(B(x,y),z)) : $r_1 = B(B(C,C),D)$
- T3 = B(B(C,C),C) : aucune solution.

Cet exemple met en relief que la forme du fait (branche poussante de $t_{n+1,n}$ fermée dans α) peut borner le nombre de solutions , indépendamment de la question. Il existe un programme non récursif qui lui est équivalent.

5-7-4) REGLE PERIODIQUE PLUS COMPLEXE.

Soit le programme PROLOG suivant :

P(B(B(C,D),B(E,F))) ->;

P(B(B(x,y),z)) -> P(B(z,B(y,x))) ;

Si l'on itère le processus de résolution ; nous obtenons :

. $B(B(x_1,y_1),z_1)$ -1-> $B(z_1,B(y_1,x_1))$

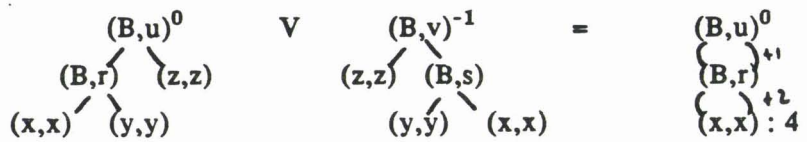
. $B(B(x_1,y_1),B(x_2,y_2))$ -2-> $B(B(y_1,x_1),B(y_2,x_2))$

. $B(B(x_1,x_3),B(x_2,y_2))$ -3-> $B(B(y_2,x_2),B(x_1,x_3))$

. $B(B(x_1,x_3),B(x_2,x_4))$ -4-> $B(B(x_1,x_3),B(x_2,x_4))$

$\forall n \geq 2$, $P(B(B(x_1,x_3),B(x_2,x_4)))$ -n-> $P(B(B(x_{n+1 \bmod 4}, x_{n+3 \bmod 4}),B(x_{n+2 \bmod 4},x_{n \bmod 4})))$

L'unifié β^0 V γ^{-1} :



Le Igop Unifié satisfait la Restriction sur les Pondérations de Boucles, cette règle récursive peut donc boucler (vrai ou sans test d'occurrence).

Pour $n \geq 2$, $A = I_{[1,n],[1,n]}(\beta^0) \text{ V } I_{[2,n+1],[1,n]}(\gamma^{-1}) = I_{[1,n+1],[1,n+4]}(\beta^0 \text{ V } \gamma^{-1})$

$A = \{(1 , B(B(x_1,x_3),B(x_2,x_4))), .. , (i , B(B(x_{i \bmod 4},x_{i+2 \bmod 4}),B(x_{i+1 \bmod 4},x_{i+3 \bmod 4}))),$

$(n+1 , B(B(x_{(n+1) \bmod 4},x_{(n+1)+2 \bmod 4}),B(x_{(n+1)+1 \bmod 4},x_{(n+1)+3 \bmod 4})))\ \}$

En termes de systèmes d'équations, la résolution du TQ s'exprime sous la forme :

$S_{[1,n]}(G) \text{ V } \{ (u_i = v_{i-1}) / \forall i \in [2,n] \} \text{ V } (u_1 = T) \text{ V } (v_n = \alpha)$

où $S_{[1,n]}(G) = \{ (u_i = B(r_i,z_i)) , (r_i = B(x_i,y_i)) , (v_i = B(z_i,s_i) , (s_i = B(y_i,x_i) / \forall i \in [1,n] \}$

1) Pour $n \geq 4$, on peut minorer les deux premiers systèmes : $Sm(n) = S_{[1,n-2]}(Gu) \cup Ind_{[1,n-1]}(Gu)$

où $S_{[1,n-2]}(Gu) = \{ (u_i = B(r_i,r_{i+1})) , (r_i = B(x_i,x_{i+2})) (x_i = x_{i \bmod 4}) \ / \forall i \in [1,n-2] \}$

et $Ind_{[1,n-2]}(Gu) = \{ (v_i = u_{i+1}) , (z_i = r_{i+1}) , (s_i = r_{i+2}) , (y_i = x_{i+2} \ / \ \forall i \in [1,n-2] \}$

$\Sigma(n) = Sg \cup Sm(n) \cup Sd/n$ avec $Sg = \emptyset$

et $Sd = \{ (u_{n-1} = B(r_{n-1},r_n) , (u_n = B(r_n,z_n) , (v_{n-1} = u_n) , (v_n = B(z_n,s_n) , (x_{n-1} = y_{n-3})$

$, (x_n = y_{n-2}) , (y_{n-1} = x_{n-3}) , (y_n = x_{n-2}) , (z_{n-1} = r_n) , (z_n = s_{n-1}) \}$

La résolution du TQ s'exprime : $\Sigma(n) \text{ V } (u_1 = T) \text{ V } (v_n = \alpha)$.

2) $\forall n \geq 2$, P (B(B(x_1,x_3),B(x_2,x_4))) -n-> P(B(B($x_{n+1 \bmod 4}$, $x_{n+3 \bmod 4}$),B($x_{n+2 \bmod 4}$,$x_{n \bmod 4}$)))
Toutes les branches de $t_{1,n}$ et $t_{n+1,n}$ sont constantes (modulo la congruence).
Ici la seule congruence non nulle est 4 : c = 4 (PPCM des congruences).
Carac. 3 : $\forall n \geq n_3 = 2$, $\sigma_n = \sigma_g \vee \sigma_{d/Clas}$ où $t_{1,n} \vee T = t_{1,n}.\sigma_g$ et $t_{n+1,n} = t_{n+1,n}.\sigma_{d/Clas}$.
Après n_3+c-1 (=5) réécritures, on sait s'il existe des solutions et si elle sont en nombre fini, on peut dire aussi s'il existe un nombre fini de solutions plus générales.
Soient les questions suivantes :

- T1 = x ($\in$ V) -- arbre de hauteur 0

On calcule de r_0 à r_5 : $r_0 = r_4$ = B(B(C,D),B(E,F)) , $r_1 = r_5$ = B(B(F,E),B(C,D)) ,
r_2 = B(B(D,C),B(F,E)) , r_3 = B(B(D,C),B(F,E)) ,
Pour tout n $\in$ N , la solution r_n existe , car r_2, r_3, r_4 et r_5 existent.

- T2 = B(x,B(C,C)) -- Aucune solution car r_i n'existe pas par i $\in$ [1,5] .

- T3 = B(B(D,C),B(F,E)) -- Entre r_0 et r_5 , r_2 existe. De plus r_6 existe ,donc r_{2+4n} existe quelque soit n $\in$ N.

Cet exemple met en relief, combien la forme du Igop unifié clarifie, la compréhension d'une règle récursive. Celle-ci est, de façon inattendue, périodique (4) et se comporte comme une simple "permutation de branches".
Avec au moins 2 réécritures, elle est équivalente à la régle :

P (B(B(w,x),B(y,z))) -> P(B(B(y,z),B(x,w)));

qui a, d'ailleurs, un Igop unifié équivalent .

Un exemple similaire est B(B(B(x,y),z),w) -> B(w,B(z,B(y,x))) ; (période 8)

Conclusion

La finalité de ces recherches sur l'arrêt et la complexité de règles récursives simples est liée à l'étude d'une classe plus large de programmes récursifs tels que :

ADD (Zéro , x , x) -> ;
ADD (Succ(x) , y , Succ(z)) -> ADD (x , y , z) ;
MUL (Zéro , x , Zéro) -> ;
MUL (Succ(x) , y , z) -> MUL (x , y , z') ADD (y , z' , z) ;

ou encore

FIBONNACCI (Succ(Zéro) , Succ (Zéro) , Zéro) -> ;
FIBONNACCI (Succ(n) , f_{n+1} , f_n) -> FIBONNACCI (n , f_n , f_{n-1})
ADD (f_n , f_{n-1} , f_{n+1}) ;

programmes qui peuvent être vus comme des programmes TQ imbriqués (ici 2).

De manière plus générale, soient P un programme Prolog et T un but composé d'un littéral, la complexité du but T sur l'ensemble de clauses P peut se formuler :
Existe-t-il une fonction F_P telle qu'une solution existe pour le but T ssi elle est obtenue en moins de F_P (T) inférences ?

Si les programmes non récursifs sont évidemment de complexité constante, nous avons montré que pour des questions et des faits linéaires (une occurrence par variable), le Tant Que est de complexité linéaire. Dès que l'on élargit la classe de programmes à ceux contenant une règle récursive, leur arrêt (i.e. l'existence de solutions) est indécidable (Codification du Problème de Post [Lebègue]).

Pourtant si l'on accepte de parler de programmation *naturelle* en programmation logique, cette codification de POST ne l'est pas du tout. En d'autres termes , ce programme est "mal écrit". Dans le cadre d'une bonne méthodologie en programmation logique, des résultats sur la décidabilité et la complexité peuvent être espérés sur des schémas de programmes plus complexes.

Sur le plus simple des programmes récursifs, nous avons pu répondre aux questions suivantes :

1) Pour tout T, la règle satisfait la propriété de terminaison :

T quelconque --r--> finiment

ssi . pour une résolution sans test d'occurrence : $\beta \vee \gamma$ n' existe pas

. pour une résolution avec test d'occurrence :

soit $\beta \vee \gamma$ n' existe pas

soit $\beta^0 \vee \gamma^{-1}$ contient des boucles de pondération nulle modulo la congruence.

Remarque : Pour savoir donc qu'une règle récursive peut engendrer une infinité de réécritures pour certaines questions, il suffit d'une unification et un test sur les pondérations de boucles.

2) Si l'on applique une résolution avec test d'occurrence, il est toujours possible de transformer le processus récursif d'utilisation de $\beta \rightarrow \gamma$ par sa caractérisation itérative. Plus précisément, on peut exprimer les contraintes liées à n utilisations de la règle sous la forme du système d'équations:

Sg U Système Itératif U Sd (une seule équation par variable indicée)

où Sd et Sd sont des systèmes constants

et Système Itératif contient des équations de la forme $x_i = t_i$, c-à-d identiques à un décalage d'indices près.

3) En se basant sur cette caractérisation itérative de la récursivité, on peut exprimer la propriété de terminaison d'une règle

r : $\beta \rightarrow \gamma$;

T ?

. Il est décidable de savoir si la règle r satisfait quelque soit T terme clos la propriété de terminaison.

. Il est décidable de savoir si un terme <u>linéaire</u> T donné et la règle satisfait la propriété de terminaison : T donné --r--> finiment

. Il existe une fonction linéaire f indépendante de T telle que T linéaire et R satisfont la propriété de terminaison ssi T ne peut être réécrit plus de f(Hauteur(T)) par r .

4) Soit notre programme TQ Prolog où le fait et le but sont linéaires, appliquons une résolution avec test d'occurrence :

f : α -> ;

r : β -> γ ;

T ?

. l'existence de solutions au TQ Prolog est décidable après un nombre de réécritures linéairement dépendant du but T.

. l'existence d'un ensemble infini de solutions (i.e. de chemins de preuve) est décidable après un nombre de réécritures linéairement dépendant du but T. Dans le cadre d"une recherche classqique de toutes les solutions, l'arrêt de la résolution est décidable en un nombre de réécritures linéairement dépendant de la hauteur de la question.

Ces résultats semblent pouvoir être généraliser, sans difficultés majeures, à des termes, but et fait, non nécessairement linéaires. Il n'en reste pas moins que ces résultats peuvent être utilisés dynamiquement pour n'importe quel programme PROLOG : dans l'arbre SLD de résolution d'un programme, pour un chemin m de dérivation de la forme $m_1.m_2^n$, le programme se comporte à cet instant comme un TQ Prolog :

m_2 : P(.) -> P(.) . . . ;

et donc utiliser les résultats de décidabilité du TQ pour contrôler l'effacement du prédicat P .

Exemple :

P (F(G(F(A)))) -> ;

P (x) -> P (F(x)) ;

P (x) -> P (G(x)) ;

P (A) ?

Bibliographie

A. Colmerauer, *PROLOG II, Manuels de référence, théorique et pratique*, 1979, GIA Marseille.

G. Comyn, *Objets Infinis Calculables*, 1982, Thèse d'Etat, Lille.

B. Courcelle, *Fundamental Properties of infinite trees*, 1983, Theor. Comp. Sci. 17, pp. 95-169 .

B. Courcelle, *Equivalence and transformations of regular systems. Applications to recursive program schemes and grammars* , 1986, Theor. Comp. Sci. , Vol 42 , pp. 1-122 .

M. Dauchet, *Termination of rewriting is undecidable in the one rule case*, 1987, Internal Report IT 110, Laboratoire Informatique Fondamentale de Lille.

N. Dershowitz, *Termination*, 1985, First International Conference on Rewriting Techniques and Applications, Dijon France, pp. 180-224.

N. Dershowitz, *Termination of Rewriting*, 1987, J. Symbolic Computation 3, pp. 69-116.

P. Devienne, P. Lebègue, *Weighted Graphs, a tool for logic programming*, 1986, 11th Colloquium on Trees in Algebra and Programming, Nice, pp 100-111 .

F. Fages, *Notes sur l'unification des termes de premier ordre finis ou infinis*, Internal Report, INRIA-LIPT .

H. Gaifman, H.Mairson, *Undecidable Optimisation Problems for Database Logic Programs* , 1987, Symposium on Logic in Computer Science, New York, pp. 106-115 .

G. Huet, D.S. Lankford, *On the Uniform Halting Problem for Term-rewriting Systems*, 1978, Rapport Laboria 283, INRIA Le Chesnay, France.

G. Huet, *Confluent reductions : Abstract properties and applications to term rewriting systems*, 1980, JACM 27, pp. 797-821.

Y.E. Ioannidis, *A time bound on the materialisation of some recursively defined views*, 1985, Proc. 11th Int'l Conf. on Very Large Data Bases, Stockholm, pp. 219-226 .

D. Kozen, *Indexings of subrecursive classes*, 1980, Theor. Comp. Sci. 11, pp. 227-301 .

P. Lebègue, Thèse à paraître, Lille .

R. Lipton, L. Snyder, *On the halting of tree replacement systems*, 1977, Conference on Theorical Computer Science, Waterloo Canada, pp. 43-46.

J.W. Lloyd, *Foundations of logic programming*, 1984, Springer Verlag .

J.F. Naughton, *Data independent recursion in deductive databases*, 1986, Proc. 5th ACM Symp. on Principles of DataBase Systems, Cambridge, pp. 267-279 .

S. Pelhat, *Analyse des inférences récursives en Prolog. Système d'aide à la detection et au controle des boucles*, 1987 , Thèse , Orsay.

J.M. Pereira, *Processus communicants : un langage formel et ses modèles. Problèmes d'analyse*, 1984, Thèse de 3ème cycle, Grenoble .

Les Graphes Orientés Pondérés :
un outil pour l'étude de la terminaison et de la complexité
dans les systèmes de réécritures et en programmation logique .

Cette étude a pour objet le problème de l'arrêt et de la complexité en temps pour une classe simple de programmes genre PROLOG. Nous avons analysé statiquement le plus simple des programmes récursifs, composé d'un fait f "P(a)" , d'une règle récursive "P(g) -> P(d)" et d'un but P(T) .

Sa modélisation est basée sur un nouvel objet syntaxique, le *Graphe Orienté Pondéré*, graphe dont les arcs sont pondérés par des entiers relatifs et dont les variables peuvent être périodiques. Lors du dépliage, les pondérations le long d'une branche sont sommées et cette somme, modulo la période, indice la feuille variable de la branche. Les arbres ainsi obtenus peuvent être irrationnels. Nous étudions leurs propriétés formelles dont l'unification pour laquelle nous proposons un algorithme (avec ou sans test d'occurrence) généralisant celui des Graphes Orientés.

Tout règle récursive simple PROLOG possède son Graphe Orienté Pondéré qui caractérise le comportement et la complexité de la règle (linéaire ou constante).

Dans le cadre des systèmes de réécritures en tête, la terminaison d'une règle de réécriture en tête pour tout terme clos est démontrée décidable (indécidable pour une règle de réécriture quelconque), elle reste décidable pour un terme clos.

Dans le cadre de la programmation logique, pour le plus simple programme récursif contenant un fait et un but linéaires (une occurrence de chaque variable) et une règle récursive "P(g) -> P(d)" , on démontre la décidabilité de l'existence de solutions et de l'arrêt du programme. ceci en un nombre de réécritures linéairement dépendant de la hauteur de la question T.

Ces résultats permettent pour une plus large classe de programmes récursifs, de vérifier facilement leur terminaison (*aide aux programmeurs*), de les compiler (*rapidité*), d'améliorer la stratégie de résolution PROLOG dans ces appels récursifs (*complétude*).

MOTS-CLES :

Système de Réécriture
Narrowing
Graphe
Programmation logique,
Récursivité
Terminaison
Complexité .