E-Talk : Un Protocole de Résolution Distribuée

Philippe Lemoisson

E-talk : un protocole de résolution distribuée

Philippe Lemoisson[1]

[1] LIRMM

juin 2003

Table des matières

1 Motivations **2**

2 sémantique du calcul **3**
2.1 formalisation des énoncés 3
2.1.1 variables, domaines de valorisation et relations 3
2.1.2 assertions 5
2.1.3 requêtes 7
2.2 pilotage du calcul 8
2.2.1 événements du calcul 8
2.2.2 la notion d'influence 9
2.2.3 la métaphore de la conversation 11

3 formalisation du calcul **12**
3.1 termes du calcul 12
3.1.1 la notion de syntagme 13
3.1.2 réduction des syntagmes 14
3.1.3 définition des termes du calcul 15
3.1.4 influence entre termes 16
3.1.5 début de formalisation du jeu de cartes 18
3.2 réécriture des termes 20
3.2.1 la notion de réducteur 20
3.2.2 définition des pentominôs 23
3.2.3 primitives de la réécriture 24
3.3 graphe de calcul et pilotage de la réécriture 24
3.3.1 définition du graphe de calcul 25
3.3.2 fonctionnement du pentominô 27
3.3.3 les règles de réécriture 28

4 propriétés du calcul **30**
4.1 terminaison du calcul 30
4.1.1 une condition suffisante de terminaison 30
4.1.2 un cas de non terminaison 31
4.2 confluence du calcul 31

Chapitre 1

Motivations

Nous cherchons un cadre de calcul adapté à la modélisation des jeux ”de société” et des jeux ”économiques” ; c’est-à-dire des jeux multi-acteurs mettant en oeuvre des prises de décision et des expertises. Prenons tout de suite trois exemples :

> modèle a : Un ensemble fini d’informations économiques est donné. Un groupe d’acteurs a connaissance de ces informations, ce groupe contient des ”décideurs” et des ”experts”. Chaque décideur agit sur l’impulsion d’une information économique particulière, et produit lui-même un ou plusieurs informations économiques, en nombre fini. Pour prendre sa décision, il peut faire appel à un ou plusieurs ”experts”. Chaque expert agit sur l’impulsion d’une demande d’information économique particulière, et produit lui-même une ou plusieurs informations économiques, en nombre fini.
>
> modèle b : Un groupe de personnes joue aux cartes ; chacun porte un numéro et ils jouent à tour de rôle. Chaque joueur, sur l’impulsion ”tour t”, pose la question ”qu’ont joué les joueurs de numéro inférieur ?”, se pose la question ”quelle est l’état de ma main ?”, joue et dit ”j’ai joué le tour t”. D’autre part, un meneur de jeu, sur l’impulsion ”tour t”, pose la question ”qu’ont joué les joueurs au tour t ?”, met en oeuvre une règle de redistribution qui va actualiser les mains individuelles, puis donne l’impulsion ”tour $t + 1$”.
>
> modèle c : Les joueurs du modèle b”, avant de décider du coup à jouer, font appel à un ou plusieurs experts de leur choix.

Notre objectif est de donner à ce type de jeux une représentation dans un langage de ”haut niveau” qui permette de le traiter comme un calcul, et d’exhiber des conditions suffisantes pour que le jeu soit globalement déterministe, lorsque les stratégies de décision et les mécanismes d’expertise sont eux-mêmes déterministes.

Chapitre 2

sémantique du calcul

2.1 formalisation des énoncés

2.1.1 variables, domaines de valorisation et relations

Prenons le jeu de cartes, avec les notations suivantes :

$\mathcal{D}_X$ est l'ensemble des valeurs possibles pour la variable x =joueur
$\mathcal{D}_Y$ est l'ensemble des valeurs possibles pour la variable y =carte
$\mathcal{D}_Z$ est l'ensemble des valeurs possibles pour la variable z =tour
$\mathcal{R}^1{}_{\mathcal{D}_X \times \mathcal{D}_Y \times \mathcal{D}_Z}$ est définie par : $\mathcal{R}^1(x, y, z) \Leftrightarrow$ "x joue y au tour z"

Les premiers énoncés qui viennent à l'esprit sont des prédicats du premier ordre que nous nommerons assertions réelles, et qui auront la forme suivante :

$$Assert_{\{R_1,\, \mathcal{D}_x^1 \times \mathcal{D}_y^1 \times \mathcal{D}_z^1\}} = (\ \forall (x, y, z) \in \mathcal{D}_x^1 \times \mathcal{D}_y^1 \times \mathcal{D}_z^1\ :\ R_1(x, y, z)\ )$$

$$\text{avec : } \mathcal{D}_x^1 \subseteq \mathcal{D}_X\ ;\ \mathcal{D}_y^1 \subseteq \mathcal{D}_Y\ ;\ \mathcal{D}_z^1 \subseteq \mathcal{D}_Z$$

par exemple :

$\mathcal{D}_x^1 = \{Henri\}; \mathcal{D}_y^1 = \{Roi\ de\ Coeur\}; \mathcal{D}_z^1 = \{tour\ 1\}$
Henri joue le roi de coeur au premier tour
$\mathcal{D}_x^1 = \{Henri\}; \mathcal{D}_y^1 = \{\neq Carreau\}; \mathcal{D}_z^1 = \{tour\ 2\}$
Henri joue autre chose que Carreau au second tour
$\mathcal{D}_x^1 = \{Hommes\}; \mathcal{D}_y^1 = \{Carreaux\}; \mathcal{D}_z^1 = \{Tours\}$
Les hommes jouent les "Carreau" à tous les tours

Au cours du calcul, nous nous interesserons à des ensembles d'assertions particuliers définis en intension ; et nous parlerons alors de requêtes :

$$Req_{\{R_1,\, \mathcal{D}_x^1 \times \mathcal{D}_y^1 \times \mathcal{D}_z^1\}} = \left\{ Assert_{\{R_1,\, \mathcal{D}_x \times \mathcal{D}_y \times \mathcal{D}_z\}} \;\middle|\; \begin{array}{l} \mathcal{D}_x \subseteq \mathcal{D}_x^1 \\ \mathcal{D}_y \subseteq \mathcal{D}_y^1 \\ \mathcal{D}_z \subseteq \mathcal{D}_z^1 \end{array} \right\}$$

$\mathcal{D}_x^1 = \{Henri\}; \mathcal{D}_y^1 = \{Cartes\}; \mathcal{D}_z^1 = \{tour1\}$
De quelles informations dispose-t-on sur ce qu'a joué Henri au premier tour ?

$\mathcal{D}_x^1 = \{Joueurs\}; \mathcal{D}_y^1 = \{Carreaux\}; \mathcal{D}_z^1 = \{Tours\}$
De quelles informations dispose-t-on sur les tours où "Carreau" a été joué ?

Lors du processus de réécriture ditribuée , il sera commode également de prendre connaissance d'informations partiellement exprimées, et de nommer assertions des prédicats du premier ordre ayant la forme suivante :

$$\exists \mathcal{D}_x \subseteq \mathcal{D}_x^1,\ \exists \mathcal{D}_y \subseteq \mathcal{D}_y^1,\ \exists \mathcal{D}_z \subseteq \mathcal{D}_z^1 \text{ — } Assert_{\{R_1,\ \mathcal{D}_x \times \mathcal{D}_y \times \mathcal{D}_z\}}$$

$\mathcal{D}_x^1 = \{Hommes\}; \mathcal{D}_y^1 = \{Coeurs\}; \mathcal{D}_z^1 = \{tour\ 1\}$
Certains hommes jouent Coeur au premier tour
$\mathcal{D}_x^1 = \{Henri\}; \mathcal{D}_y^1 = \{\neq Carreau\}; \mathcal{D}_z^1 = \{Tours\}$
Henri joue autre chose que Carreau à certains tours

On appellera alors assertions virtuelles les assertions qui demandent à être précisées ; une des idées structurantes pour le calcul étant que les assertions vont se préciser progressivement par réduction des domaines, jusqu'à devenir des assertions réelles, c'est-à-dire jusqu'à ce que :

(1) : $\exists \mathcal{D}^x \subseteq \mathcal{D}_x^1, \exists \mathcal{D}^y \subseteq \mathcal{D}_y^1, \exists \mathcal{D}^z \subseteq \mathcal{D}_z^1$ — $\forall (x, y, z) \in \mathcal{D}^x \times \mathcal{D}^y \times \mathcal{D}^z : R_1(x, y, z)$

se transforme en :

$$\text{(2)} : \forall (x, y, z) \in \mathcal{D}_x^2 \times \mathcal{D}_y^2 \times \mathcal{D}_z^2 \ :\ R_1(x, y, z)$$

par exemple :

(1) : $\mathcal{D}_x^1 = \{Hommes\}; \mathcal{D}_y^1 = \{Coeurs\}; \mathcal{D}_z^1 = \{Tours\}$
Certains hommes jouent Coeur à certains tours
pourra se transformer en :
(2) : $\mathcal{D}_x^2 = \{Henri\}; \mathcal{D}_y^2 = \{RoideCoeur\}; \mathcal{D}_z^2 = \{Tours\}$
Henri joue Roi de Coeur à tous les tours

De la même façon, nous aurons besoin de distinguer requêtes réelles (au sens où elles sont effectivement posées)et requêtes virtuelles (au sens où elles vont encore être précisées).
Les notations qui suivent vont nous permettre de développer la base de la sémantique opérationnelle du calcul ; nous verrons plus loin que différents formalismes de représentation des connaissances peuvent être vus comme des cas particuliers de syntaxes associées à cette sémantique opérationnelle.

Notations :

$\mathcal{R}_{\mathcal{D}}$ désigne une relation $\mathcal{R}$ définie sur un produit cartésien $\mathcal{D} = \mathcal{D}_{X_1} \times \mathcal{D}_{X_2} \times \mathcal{D}_{X_3} \times \cdots$; les $\mathcal{D}_{X_j}$ sont alors les domaines de valorisation respectifs des variables x_j ; $\overrightarrow{x} = (x_1, x_2, x_3, \cdots)$ désignera un tuple quelconque de $\mathcal{D}$.
$\mathcal{D}^{|i}$ désigne une projection de $\mathcal{D}$ et peut se traduire par : $\exists \mathcal{D}^{|j} \text{ — } \mathcal{D} = \mathcal{D}^{|i} \times \mathcal{D}^{|j}$
Pour $\overrightarrow{x} \in \mathcal{D}$, $\overrightarrow{x}^{|i}$ désigne la projection de $\overrightarrow{x}$ sur $\mathcal{D}^{|i}$.
$\mathcal{R}^{|i}$ est définie sur $\mathcal{D}^{|i}$ par $\mathcal{R}^{|i}(\overrightarrow{x}) \Leftrightarrow \exists\ \overrightarrow{y}\ \text{ — } x = \overrightarrow{y}^{|i} \wedge \mathcal{R}(\overrightarrow{y})$.

2.1.2 assertions

Définitions :

Une *assertion* $Assert_{\{\mathcal{R}^i,\ {}^*\mathcal{X}^j\}}$ où $\mathcal{R}^i = \mathcal{R}^i_{\mathcal{D}^i}$ et $\mathcal{X}^j \subseteq \mathcal{D}^i$ et $\mathcal{X}^j \neq \emptyset$ est définie par :

$$\exists \mathcal{X} \subseteq \mathcal{X}_j \text{ — } \forall\ \overrightarrow{x} \in \mathcal{X}\ :\ \mathcal{R}^i(\overrightarrow{x})$$

Une *assertion réelle* $Assert_{\{\mathcal{R}^i,\ \mathcal{X}^j\}}$ où $\mathcal{R}^i = \mathcal{R}^i_{\mathcal{D}^i}$ et $\mathcal{X}^j \subseteq \mathcal{D}^i$ et $\mathcal{X}^j \neq \emptyset$ est définie par :

$$\forall\ \overrightarrow{x} \in \mathcal{X}_j\ :\ \mathcal{R}^i(\overrightarrow{x})$$

ce qui est équivalent à : $\exists \mathcal{X} = \mathcal{X}_j \text{ — } \forall\ \overrightarrow{x} \in \mathcal{X}\ :\ \mathcal{R}^i(\overrightarrow{x})$

On convient d'effectuer le passage au réel en supprimant "$*$" dès que le domaine est réduit à un élément unique, afin d'éviter lorsque $\mathcal{X} = \{\overrightarrow{x}\}$ le doublon entre assertion et assertion réelle.

$Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_1}\}}$ est une *réduction* de $Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_2}\}}$:

$$Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_1}\}} \preceq Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_2}\}} \text{ ssi :}$$

$Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_1}\}}$ peut-être obtenu à partir de $Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_2}\}}$ en combinant les opérations élémentaires suivantes :

$$\left[\begin{array}{l} Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^i\}} \rightarrow Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^j\}} \text{ avec : } \mathcal{X}^j \subseteq \mathcal{X}^i \wedge \mathcal{X}^j \neq \emptyset \\ Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^i\}} \rightarrow Assert_{\{\mathcal{R}^1,\ \mathcal{X}^i\}} \text{ (passage au réel)} \\ Assert_{\{\mathcal{R}^1,\ \mathcal{X}^i\}} \rightarrow Assert_{\{\mathcal{R}^1,\ \mathcal{X}^i\}} \text{ (identité sur les assertions réelles)} \end{array}\right.$$

$Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_1}\}}$ est une *spécialisation* de $Assert_{\{\mathcal{R}^2,\ {}^*\mathcal{X}^{i_2}\}}$:

$$Assert_{\{\mathcal{R}^1,\ {}^*\mathcal{X}^{i_1}\}} \leq Assert_{\{\mathcal{R}^2,\ {}^*\mathcal{X}^{i_2}\}} \text{ ssi :}$$

$$\exists k \left| \begin{array}{l} \mathcal{R}^2 = \mathcal{R}^{1|k} \\ Assert_{\{\mathcal{R}^{1|k},\ {}^*\mathcal{X}^{i_1|k}\}} \preceq Assert_{\{\mathcal{R}^2,\ {}^*\mathcal{X}^{i_2}\}} \end{array}\right.$$

par exemple :

$\mathcal{R}^1$ défini par : ”Joueur” joue ”carte” à ”tour”
$\mathcal{R}^2$ défini par : ”Joueur” joue ”carte”
$\mathcal{X}^{i_1} = \{Henri\} \times \{Roi\ de\ Coeur\} \times \{tour1\}$
$\mathcal{X}^{i_2} = \{Hommes\} \times \{Coeurs\} \times \{tour1\}$
$\mathcal{X}^{i_3} = \{Hommes\} \times \{Coeurs\}$

vérifient :

$$\mathcal{X}^{i_1} \subseteq \mathcal{X}^{i_2} \quad ; \quad \exists k \text{ — } \mathcal{X}^{i_3} = \mathcal{X}^{i_2|k}$$

nous pouvons construire les énoncés :

$A_1 = Assert_{\{\mathcal{R}^1,\ \mathcal{X}^{i_1}\}}$: *Henri joue le Roi de Coeur au tour 1*
$A_2 = Assert_{\{\mathcal{R}^1,\ *\mathcal{X}^{i_2}\}}$: *Certains hommes jouent Coeur au tour 1*
$A_3 = Assert_{\{\mathcal{R}^1,\ \mathcal{X}^{i_2}\}}$: *Les hommes jouent les ”Coeur” au tour 1*
$A_4 = Assert_{\{\mathcal{R}^2,\ *\mathcal{X}^{i_3}\}}$: *Certains hommes jouent Coeur*
$A_5 = Assert_{\{\mathcal{R}^2,\ \mathcal{X}^{i_3}\}}$: *Les hommes jouent les ”Coeur”*

ce qui donne les réductions :

$$A_1 \preceq A_2 \quad A_3 \preceq A_2 \quad A_5 \preceq A_4$$

et les spécialisations :

$$A_2 \leq A_4 \quad A_3 \leq A_5$$

Propriétés :

”$\preceq$” et ”$\leq$” sont deux ordres partiels sur l’ensemble des assertions tels que :

$$(\preceq) \Rightarrow (\leq)$$

La réduction correspond à une substitution de domaine par un domaine plus petit ; cette propriété sera très largement utilisée durant tout le calcul.

La spécialisation permet de prendre en compte les implications logiques entre prédicats du premier ordre du type :

Henri joue le Roi de Coeur au tour 1 $\Longrightarrow$ *Henri joue le Roi de Coeur.*

2.1.3 requêtes

Définitions :

Une *requête* $Req_{\{\mathcal{R}^i,\, \mathcal{X}^j\}}$ où $\mathcal{R}^i = \mathcal{R}^i_{\mathcal{D}^i}$ et $\mathcal{X}^j \subseteq \mathcal{D}^i$ et $\mathcal{X}^j \neq \emptyset$ est l'ensemble d'assertions réelles défini par :

$$Req_{\{\mathcal{R}^i,\, \mathcal{X}^j\}} = \{Assert_{\{\mathcal{R}^i,\, \mathcal{X}^k\}} — \mathcal{X}^k \subseteq \mathcal{X}^j\}$$

$Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_1}\}}$ est une *réduction* de $Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_2}\}}$:

$$Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_1}\}} \preceq Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_2}\}} \text{ ssi}$$

$Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_1}\}}$ peut-être obtenu à partir de $Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_2}\}}$ par l'opération élémentaire suivante :

$$Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i}\}} \rightarrow Req_{\{\mathcal{R}^1,\, \mathcal{X}^{j}\}} \text{ avec : } \mathcal{X}^j \subseteq \mathcal{X}^i \text{ et } \mathcal{X}^j \neq \emptyset$$

par exemple :

$\mathcal{R}^1$ défini par : "Joueur" joue "carte" à "tour"
$\mathcal{X}^{i_1} = \{Henri\} \times \{Roi\ de\ Coeur\} \times \{tour\ 1\}$
$\mathcal{X}^{i_2} = \{Hommes\} \times \{Coeurs\} \times \{tour\ 1\}$

vérifient :

$$\mathcal{X}^{i_1} \subseteq \mathcal{X}^{i_2}$$

nous pouvons construire les requêtes :

$R_1 = Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_1}\}}$: *Est-ce que Henri joue le Roi de Coeur au premier tour ?*
$R_2 = Req_{\{\mathcal{R}^1,\, \mathcal{X}^{i_2}\}}$: *Est-ce que certains hommes jouent Coeur au premier tour ?*

ce qui donne la réduction :

$$R_1 \preceq R_2$$

Propriétés :

"$\preceq$" est un ordre partiel sur l'ensemble des requêtes.

Nous ne définissons pas de relation analogue à la spécialisation des assertions, car les requêtes sont des ensembles et non pas des prédicats ; mais nous allons maintenant définir une relation entre assertions et requêtes, précisément parce que les requêtes sont des ensembles d'assertions.

2.2 pilotage du calcul

Nous avons déjà défini un ordre partiel ”naturel” sur les assertions et sur les requêtes, ainsi que la relation de spécialisation des assertions ; nous allons maintenant nous intéresser à d'autres relations construites à partir de celles ci afin de pouvoir piloter le calcul. Nous allons en particulier chercher la ”signature syntaxique” des événements suivants :

”une assertion répond à une requête”

”une assertion déclenche une réécriture”

”une requête déclenche une réécriture”

2.2.1 événements du calcul

Nous donnerons au chapitre suivant une définition précise des modules de réécriture ; ils seront basés sur une propriété fondamentale :

Règle 1 : Les modules de réécriture sont définis ”à une réduction près” ; c'est-à-dire qu'ils acceptent toute substitution de leurs ”énoncés-variables” par des énoncés qui en sont des réductions.

Regardons à nouveau quelques exemples qui permettre d'illustrer la suite :

$\mathcal{R}^1$ défini par : ”Joueur” joue ”carte” à ”tour”

$\mathcal{R}^2$ défini par : ”Joueur” joue ”carte”

$\mathcal{X}^{i_1} = \{Henriette\} \times \{RoideCoeur\} \times \{tour1\}$

$\mathcal{X}^{i_2} = \{Joueurs\} \times \{Coeurs\}$

$\mathcal{X}^{i_3} = \{Hommes\} \times \{Coeurs\}$

$\mathcal{X}^{i_4} = \{Joueurs\} \times \{Coeurs\} \times \{tour\ 1\}$

$R_4 = Req_{\{\mathcal{R}^2,\ \mathcal{X}^{i_2}\}}$: *Est-ce que certains joueurs jouent Coeur ?*

$A_5 = Assert_{\{\mathcal{R}^1,\ \mathcal{X}^{i_1}\}}$: *Henriette joue le Roi de Coeur au tour 1*

$A_6 = Assert_{\{\mathcal{R}^1,\ \mathcal{X}^{i_3}\}}$: *Les hommes jouent les ”Coeur”*

$A_7 = Assert_{\{\mathcal{R}^1,\ *\mathcal{X}^{i_3}\}}$: *Certains hommes jouent ”Coeur”*

$R_8 = Req_{\{\mathcal{R}^2,\ \mathcal{X}^{i_2}\}}$: *Est-ce que certains joueurs jouent Coeur ?*

$R_9 = Req_{\{\mathcal{R}^1,\ \mathcal{X}^{i_4}\}}$: *Est-ce que certains joueurs jouent Coeur au tour 1 ?*

une assertion répond à une requête

Nous voudrons pouvoir dire que A_6 mais aussi A_5 répondent à R_4, de façon à obtenir toutes les réponses naturelles à une question ; or nous savons déjà : $A_6 \in R_4$ et par ailleurs $\exists k$ — $\mathcal{R}^2 = \mathcal{R}^{1|k}$

Définition :

Une assertion réelle $Assert_{\{\mathcal{R}^1,\, \mathcal{X}^{i_1}\}}$ *répond à* une requête $Req_{\{\mathcal{R}^2,\, \mathcal{X}^{i_2}\}}$:

$$Assert_{\{\mathcal{R}^1,\, \mathcal{X}^{i_1}\}} \lhd Req_{\{\mathcal{R}^2,\, \mathcal{X}^{i_2}\}} \text{ ssi :}$$

$$\exists k \left| \begin{array}{l} \mathcal{R}^2 = \mathcal{R}^{1|k} \\ Assert_{\{\mathcal{R}^{1|k},\, \mathcal{X}^{i_1|k}\}} \in Req_{\{\mathcal{R}^2,\, \mathcal{X}^{i_2}\}} \end{array} \right.$$

une assertion déclenche une réécriture

L'idée de base est que les modules de réécriture déclenchables par assertions seront l'équivalent de règles : "si ... alors". Or nous disposons déjà de la relation de spécialisation pour prendre en compte des implications logiques "syntaxiquement décelables" entre assertions.

Nous voudrons pouvoir dire que A_5 et A_6 déclenchent le module de réécriture commandé par A_7.

Réécriture déclenchée par assertion : Toutes les assertions "spécialisations de a" (et seulement celles-ci) joueront le rôle de déclencheur pour le module de réécriture commandé par "a".

une requête déclenche une réécriture

L'idée est ici que les modules de réécriture déclenchables par requêtes auront pour mission principale de fournir toutes les réponses à ces requêtes.

Nous voudrons pouvoir dire que R_5 déclenche le module de réécriture commandé par R_8.

Par contre, ce même module, qui travaille "à une réduction près" ne saura pas fournir les réponses à R_9.

Réécriture déclenchée par requête : Toutes les requêtes "réductions de r" (et seulement celles-ci) joueront le rôle de déclencheur pour le module de réécriture commandé par "r".

2.2.2 la notion d'influence

Au cours du calcul, la réécriture sera contrainte par la règle suivante :

Règle 2 : Les seuls termes ajoutés par réécriture sont des réductions de termes déjà présents.

Nous allons donc nous intéresser aux éventualités suivantes :

– Est-ce qu'une assertion peut produire par réduction une spécialisation d'une autre assertion ?
– Est-ce qu'une requête peut produire par réduction une réduction d'une autre requête ?
– Est-ce qu'une assertion peut produire par réduction une réponse à une requête ?

Définitions :

Une assertion *influence* une assertion.

$$Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_1}\}} \lhd\!\!\lhd \; Assert_{\{\mathcal{R}^{i_2},\, *\mathcal{X}^{j_2}\}} \text{ ssi :}$$

$$\exists\, Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_0}\}} \left| \begin{array}{l} Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_0}\}} \preceq Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_1}\}} \\ Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_0}\}} \leq Assert_{\{\mathcal{R}^{i_2},\, *\mathcal{X}^{j_2}\}} \end{array} \right.$$

Une requête *influence* une requête.

$$Req_{\{\mathcal{R}^{i},\, \mathcal{X}^{j_3}\}} \lhd\!:\!\rhd \; Req_{\{\mathcal{R}^{i},\, \mathcal{X}^{j_4}\}} \text{ ssi}$$

$$\exists\, Req_{\{\mathcal{R}^{1},\, \mathcal{X}^{j_0}\}} \left| \begin{array}{l} Req_{\{\mathcal{R}^{i},\, \mathcal{X}^{j_0}\}} \preceq Req_{\{\mathcal{R}^{i},\, \mathcal{X}^{j_3}\}} \\ Req_{\{\mathcal{R}^{i},\, \mathcal{X}^{j_0}\}} \preceq Req_{\{\mathcal{R}^{i},\, \mathcal{X}^{j_4}\}} \end{array} \right.$$

Une assertion *influence* une requête réelle.

$$Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_1}\}} \lhd\!: \; Req_{\{\mathcal{R}^{i_5},\, \mathcal{X}^{j_5}\}} \text{ ssi}$$

$$\exists\, Assert_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_0}\}} \left| \begin{array}{l} Assert_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_0}\}} \preceq Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_1}\}} \\ Assert_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_0}\}} \lhd \; Req_{\{\mathcal{R}^{i_5},\, \mathcal{X}^{j_5}\}} \end{array} \right.$$

2.2.3 la métaphore de la conversation

Comparons les ”modules de réécriture” à un groupe d’agents en conversation ; les règles de la conversation sont alors les suivantes :

> Chaque agent pose en début de calcul des assertions et des requêtes partiellement explicitées.
>
> Il y a deux sortes d’agents :
> – ceux qui interviennent dans la conversation lorsqu’ils sont stimulés par les spécialisations d’une assertion donnée.
> – ceux qui interviennent dans la conversation lorsqu’ils sont stimulés par les réductions d’une requête donnée.
>
> Les agents interviennent simultanément sur des stimulations parallèles.
>
> Chaque agent, lorsqu’il intervient, émet des réductions des assertions et des requêtes qu’il a initialement posées.
>
> Lorsqu’un agent émet une requête, il attend d’en obtenir la réponse avant de poursuivre son intervention.
>
> Lorsqu’un agent sait qu’il n’interviendra plus, c’est-à-dire lorsqu’aucune assertion ou requête n’influence plus son déclencheur, il se retire de la conversation.

L’objet du chapitre suivant est la formalisation de ce principe de calcul

Chapitre 3

formalisation du calcul

3.1 termes du calcul

Tout d'abord, nous allons considérer assertions et requêtes comme éléments d'un même ensemble de "syntagmes"[1] ; en y intégrant les notions de syntagme déclencheur ou "trigger" et de syntagme émis ou "data".

Puis nous allons définir les termes du calcul à partir des syntagmes en intégrant les listes de modules, mécanisme qui permettra de gérer ces termes au fil des réécritures successives.

Ensuite, nous allons revenir sur la notion de réduction pour l'étendre aux syntagmes et sur la notion d'influence pour l'étendre aux termes du calcul.

Enfin, nous allons étudier les propriétés de l'influence du point de vue de la sémantique opérationnelle.

[1] syntagme : groupe de mots ayant un sens (Petit Robert)

3.1.1 la notion de syntagme

Nous supposons donnés un ensemble de variables $\{x_i\}$, et un ensemble de relations $\{\mathcal{R}^i_{\mathcal{D}^i}\}$ entre tuples $\vec{x}$ construits sur ces variables

Définition :

On appelle *syntagme* un quintuplet $(\mathcal{R}, \delta, \alpha, \varepsilon, \mathcal{X})$ tel que :

$\mathcal{R}$ est une relation $\mathcal{R}_{\mathcal{D}}$; c'est la *structure relationnelle* du syntagme.

δ est le booléen $(data/trigger)$
- $\delta = 0$: le syntagme est un *data*, produit lors du calcul
- $\delta = 1$: le syntagme est un *trigger*, déclencheur lors du calcul

α est le booléen $(assert/request)$
- $\alpha = 0$: le syntagme est une assertion
- $\alpha = 1$: le syntagme est une requête

ε est le booléen $(real/virtual)$
- $\varepsilon = 0$: le syntagme est réel
- $\varepsilon = 1$: le syntagme n'est pas réel

$\mathcal{X} \neq \emptyset$ est un domaine de valorisation inclus dans $\mathcal{D}$
- $\alpha = 0$; $\varepsilon = 0$: le syntagme est l'assertion $Assert_{\{\mathcal{R},\, \mathcal{X}\}}$
  définie par $\forall\ \vec{x} \in \mathcal{X}\ :\ \mathcal{R}(\vec{x})$
- $\alpha = 0$; $\varepsilon = 1$: le syntagme représente l'assertion $Assert_{\{\mathcal{R},\, {}^*\mathcal{X}\}}$
  définie par $\exists \mathcal{X}' \subseteq \mathcal{X} \text{ — } \forall\ \vec{x} \in \mathcal{X}'\ :\ \mathcal{R}(\vec{x})$
- $\alpha = 1$: le syntagme est la requête $Req_{\{\mathcal{R},\, \mathcal{X}\}}$
  définie par $\{Assert_{\{\mathcal{R},\, \mathcal{X}^k\}} \text{ — } \mathcal{X}^k \subseteq \mathcal{X}\}$

contrainte : les syntagmes dont le domaine est réduit à un élément sont forcément réels :

$$(Card(\mathcal{X}) = 1) \Rightarrow (\varepsilon = 0)$$

On notera σ_i le syntagme $(\mathcal{R}^i, \delta_i, \alpha_i, \varepsilon_i, \mathcal{X}^i)$ avec $\mathcal{R}^i = \mathcal{R}^i{}_{\mathcal{D}^i}$.

On appellera *domaine* de σ_i le domaine de valorisation $\mathcal{X}^i \subseteq \mathcal{D}^i$

On appellera *valeur* de σ_i et on notera $val(\sigma_i)$ le quadruplet $(\delta_i, \alpha_i, \varepsilon_i, \mathcal{X}^i)$.

On notera parfois $(\mathcal{R}^i, val(\sigma_i))$ le syntagme $(\mathcal{R}^i, \delta_i, \alpha_i, \varepsilon_i, \mathcal{X}^i)$

On notera $\mathcal{S}$ l'ensemble des syntagmes.

La notion de requête réelle ou non réelle étant introduite dans la syntaxe ; nous allons la définir via la relation de réduction.

3.1.2 réduction des syntagmes

Nous allons étendre aux syntagmes la notion de réduction définie précédemment.

Définition :

Soient $\sigma_1 = (\mathcal{R}^1, \delta_1, \alpha_1, \varepsilon_1, \mathcal{X}^1)$ et $\sigma_2 = (\mathcal{R}^2, \delta_2, \alpha_2, \varepsilon_2, \mathcal{X}^2)$ deux syntagmes, "σ_1 est une *réduction* de σ_2" est défini par :

$$\sigma_1 \preceq \sigma_2 \Leftrightarrow \begin{cases} 1. & \mathcal{R}^1 = \mathcal{R}^2 \\ 2. & \delta_1 = 0 \\ 3. & \alpha_1 = \alpha_2 \\ 4. & \mathcal{X}^1 \subseteq \mathcal{X}^2 \\ 5. & \varepsilon_1 \leq \varepsilon_2 \\ 6. & (\varepsilon_2 = 0) \Rightarrow (\mathcal{X}^1 = \mathcal{X}^2) \end{cases}$$

1. exprime que la réduction ne compare que des syntagmes de structure relationnelle indentique
2. exprime que les triggers sont maximaux pour la réduction
3. exprime que les assertions sont comparables aux assertions, et les requêtes aux requêtes
4. exprime qu'un domaine de réduction est une réduction de domaine
5. exprime qu'une réduction de réel est réelle
6. exprime qu'une réduction de réel est de même domaine

On note $\{\preceq \sigma\}$ l'ensemble des syntagmes qui peuvent être obtenus par réduction à partir du syntagme σ.

Si X est un ensemble de syntagmes, on note $\{\preceq X\}$ l'ensemble des syntagmes $\bigcup_{\sigma \in X} \{\preceq \sigma\}$.

Nous sommes maintenant en mesure de distinguer requêtes virtuelles et requêtes réelles, et cette nuance correspond à la possibilité ou à l'impossibilité d'opérer des réductions.

3.1.3 définition des termes du calcul

Définition et notations :

On appelle *terme du calcul* un couple (σ_i, L_i) tel que :

$\sigma_i = (\mathcal{R}^i, \delta_i, \alpha_i, \varepsilon_i, \mathcal{X}^i)$ est le syntagme identifiant le terme ;

L_i est une liste de modules de réécriture.

On notera :

$\mathcal{A}$	:	l'ensemble des termes data assertions
$\mathcal{A}^{Real}$	:	l'ensemble des termes data assertions réelles
$\mathcal{R}$	:	l 'ensemble des termes data requêtes
$\mathcal{R}^{Real}$	:	l'ensemble des termes data requêtes réelles
$\mathcal{R}^{Virtual}$	:	l'ensemble des termes data requêtes non réelles
$\mathcal{T}$	:	l'ensemble des termes
a_i	:	un terme quelconque de $\mathcal{A}$
r_i	:	un terme quelconque de $\mathcal{R}$
x_i	:	un terme quelconque de $\mathcal{T}$

Dans la suite, on confondra parfois les termes du calcul (objets de la réécriture) et les syntagmes qui les identifient et qui seront les arguments de la réécriture.

Nous allons maintenant étendre aux termes du calcul nouvellement définis la notion d'influence qui avaot été introduite au chapitre précédent, puis montrer qu'elle offre une caractérisation syntaxique simple des événements de calcul qui nous intéressent et que nous précisons ainsi :

"une assertion RÉELLE déclenche une réécriture"

"une requête RÉELLE déclenche une réécriture"

"une assertion RÉELLE répond à une requête RÉELLE"

Nous montrerons également qu'elle possède une propriété intéressante vis à vis de la relation de réduction.

3.1.4 influence entre termes

Définitions :

Dans ce qui suit :

$\sigma_1 = (\mathcal{R}^1, 0, 0, \varepsilon_1, \mathcal{X}^1)$ est un data assertion quelconque
$\sigma_2 = (\mathcal{R}^2, 1, 0, \varepsilon_2, \mathcal{X}^2)$ est un trigger assertion quelconque
$\sigma_3 = (\mathcal{R}^3, 0, 1, \varepsilon_3, \mathcal{X}^3)$ est un data requête quelconque
$\sigma_4 = (\mathcal{R}^4, 1, 1, \varepsilon_4, \mathcal{X}^4)$ est un trigger requête quelconque
$\sigma_5 = (\mathcal{R}^5, 0, 1, 0, \mathcal{X}^5)$ est un data requête réel

On dit qu'un data assertion *influence* un trigger assertion,

$$a_1 = (\sigma_1, L_1) \lhd a_2 = (\sigma_2, L_2) \text{ ssi :}$$

$$\begin{cases} \exists\, k \text{ — } \mathcal{R}^2 = \mathcal{R}^{1|k} \\ \exists\, \sigma_0 = (\mathcal{R}^1, 0, 0, \varepsilon_0, \mathcal{X}^{j_0}) \end{cases} \Bigg| \begin{array}{l} \sigma_0 \preceq \sigma_1 \\ (\mathcal{R}^{1|k}, 0, 0, \varepsilon_0, \mathcal{X}^{j_0|k}) \preceq \sigma_2 \end{array}$$

On dit qu'un data requête *influence* un trigger requête,

$$r_3 = (\sigma_3, L_3) \lhd\!\rhd r_4 = (\sigma_4, L_4) \text{ ssi :}$$

$$\begin{cases} \mathcal{R}^3 = \mathcal{R}^4 \\ \exists\, \sigma_0 = (\mathcal{R}^3, 0, 1, \varepsilon_0, \mathcal{X}^{j_0}) \end{cases} \Bigg| \begin{array}{l} \sigma_0 \preceq \sigma_3 \\ \mathcal{X}^{j_0} \subseteq (\mathcal{X}^3 \cap \mathcal{X}^4) \end{array}$$

On dit qu'un data assertion *influence* un data requête,

$$a_1 = (\sigma_1, L_1) \lhd r_5 = (\sigma_5, L_5) \text{ ssi :}$$

$$\begin{cases} \exists\, k \text{ — } \mathcal{R}^5 = \mathcal{R}^{1|k} \\ \exists\, \sigma_0 = (\mathcal{R}^1, 0, 0, \varepsilon_0, \mathcal{X}^{j_0}) \end{cases} \Bigg| \begin{array}{l} \sigma_0 \preceq \sigma_1 \\ (\mathcal{R}^{1|k}, 0, 0, \varepsilon_0, \mathcal{X}^{j_0|k}) \preceq \sigma_5 \end{array}$$

non-propriété :

Les relations "influence" ne sont pas transitives ; cela est du au fait que "$\mathcal{X}_i \cap \mathcal{X}_j \neq \emptyset$" ne définit pas une relation transitive entre domaines.

Le lemme 0 va utiliser la relation "influence" pour nous fournir une nouvelle représentation syntaxique des événements de calcul.

Lemme 0 :

1. $\left\{ \begin{array}{l} a_1 \lhd a_2 \\ a_1 = (\mathcal{R}^1, 0, 0, \varepsilon_1, \mathcal{X}^1) \\ a_2 = (\mathcal{R}^2, 1, 0, \varepsilon_2, \mathcal{X}^2) \end{array} \right\} \Leftrightarrow$ *un terme influence une réécriture par assertion* $\Leftrightarrow (Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_1}\}} \lhd Assert_{\{\mathcal{R}^{i_2},\, *\mathcal{X}^{j_2}\}})$

2. $\left\{ \begin{array}{l} r_3 \lhd\!:\!\rhd\ r_4 \\ r_3 = (\mathcal{R}^3, 0, 1, \varepsilon_3, \mathcal{X}^3) \\ r_4 = (\mathcal{R}^4, 1, 1, \varepsilon_4, \mathcal{X}^4) \end{array} \right\} \Leftrightarrow$ *un terme influence une réécriture par requête* $\Leftrightarrow (Req_{\{\mathcal{R}^i,\, \mathcal{X}^{j_3}\}} \lhd\!:\!\rhd\ Req_{\{\mathcal{R}^i,\, \mathcal{X}^{j_4}\}})$

3. $\left\{ \begin{array}{l} r_5 \in \mathcal{R}^{Real} \\ a_1 \lhd r_5 \\ a_1 = (\mathcal{R}^1, 0, 0, \varepsilon_1, \mathcal{X}^1) \\ r_5 = (\mathcal{R}^5, 0, 1, \varepsilon_5, \mathcal{X}^5) \end{array} \right\} \Leftrightarrow$ *une assertion influence une requête* $\Rightarrow (Assert_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_1}\}} \lhd\!: Req_{\{\mathcal{R}^{i_5},\, \mathcal{X}^{j_5}\}})$

4. $\left\{ \begin{array}{l} a_1 \in \mathcal{A}^{Real} \\ a_1 \lhd a_2 \\ a_1 = (\mathcal{R}^1, 0, 0, \varepsilon_1, \mathcal{X}^1) \\ a_2 = (\mathcal{R}^2, 1, 0, \varepsilon_2, \mathcal{X}^2) \end{array} \right\} \Leftrightarrow$ *une assertion déclenche une réécriture* $\Leftrightarrow (Assert_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_1}\}} \leq Assert_{\{\mathcal{R}^{i_2},\, *\mathcal{X}^{j_2}\}})$

5. $\left\{ \begin{array}{l} r_3 \in \mathcal{R}^{Real} \\ r_3 \lhd\!:\!\rhd\ r_4 \\ r_3 = (\mathcal{R}^3, 0, 1, \varepsilon_3, \mathcal{X}^3) \\ r_4 = (\mathcal{R}^4, 1, 1, \varepsilon_4, \mathcal{X}^4) \end{array} \right\} \Leftrightarrow$ *une requête déclenche une réécriture* $\Rightarrow (Req_{\{\mathcal{R}^i,\, \mathcal{X}^{j_3}\}} \preceq Req_{\{\mathcal{R}^i,\, \mathcal{X}^{j_4}\}})$

6. $\left\{ \begin{array}{l} a_1 \in \mathcal{A}^{Real} \\ r_5 \in \mathcal{R}^{Real} \\ a_1 \lhd r_5 \\ a_1 = (\mathcal{R}^1, 0, 0, \varepsilon_1, \mathcal{X}^1) \\ r_5 = (\mathcal{R}^5, 0, 1, \varepsilon_5, \mathcal{X}^5) \end{array} \right\} \Leftrightarrow$ *une assertion répond à une requête* $\Rightarrow (Assert_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_5}\}} \lhd\ Req_{\{\mathcal{R}^{i_2},\, \mathcal{X}^{j_5}\}})$

démonstration de "Lemme 0" :

Les $\Rightarrow$ qui ne sont pas des $\Leftrightarrow$ proviennent du fait que la nuance "requête réelle" a été introduite.
1 : $\sigma_0 = (\mathcal{R}^i, 0, 0, \varepsilon_0, \mathcal{X}^{j_0})$ dans 2.2.2 correspond à $Assert_{\{\mathcal{R}^{i_1},\, *\mathcal{X}^{j_0}\}}$ dans 3.1.4
2 : $\sigma_0 = (\mathcal{R}^i, 0, 1, \varepsilon_0, \mathcal{X}^{j_0})$ dans 2.2.2 correspond à $Req_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_0}\}}$ dans 3.1.4
3 : $\sigma_0 = (\mathcal{R}^i, 0, 0, 0, \mathcal{X}^{j_0})$ dans 2.2.2 correspond à $Assert_{\{\mathcal{R}^{i_1},\, \mathcal{X}^{j_0}\}}$ dans 3.1.4
4 : $a_1 \in \mathcal{A}^{Real}$ contraint $j_0 = 1$, le reste découle de 1.
5 : $r_3 \in \mathcal{R}^{Real}$ contraint $j_0 = 3$, le reste découle de 2.
6 : $a_1 \in \mathcal{A}^{Real}$ contraint $j_0 = 1$, le reste découle de 3.

Le lemme suivant sera très utile dans la démonstration des propriétés du calcul, il exprime le fait que la réduction ne crée pas de nouvelles "influences", c'est-à-dire que les "zones du graphe de calcul" influençant les déclencheurs et les requêtes vont évoluer de façon sympathique au fil du calcul.

Lemme 1 :

Soient les termes :

$$x_1 = (\sigma_1, L_1)$$
$$x_2 = (\sigma_2, L_2)$$
$$x_{10} = (\sigma_{10}, L_{10})$$
$$x_{20} = (\sigma_{20}, L_{20})$$

L'implication suivante est toujours vraie :

$$\left\{ \begin{array}{l} \sigma_1 \preceq \sigma_{10} \\ \sigma_2 \preceq \sigma_{20} \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} (x_1 \ \lhd\!:\!\rhd \ x_2) \Rightarrow (x_{10} \ \lhd\!:\!\rhd \ x_{20}) \\ (x_1 \ \lhd \ x_2) \Rightarrow (x_{10} \ \lhd \ x_{20}) \end{array} \right\}$$

démonstration de "Lemme 1" :

Cas de la relation $\lhd$

$$\left. \begin{array}{r} \sigma_1 = (\mathcal{R}^1, \delta, \alpha, \varepsilon_i, \mathcal{X}^1) \\ \sigma_2 = (\mathcal{R}^2, \delta, \alpha, \varepsilon_2, \mathcal{X}^2) \\ \sigma_{10} = (\mathcal{R}^1, \delta, \alpha, \varepsilon_{10}, \mathcal{X}^{10}) \\ \sigma_{20} = (\mathcal{R}^2, \delta, \alpha, \varepsilon_{20}, \mathcal{X}^{20}) \\ \sigma_1 \preceq \sigma_{10} \\ \sigma_2 \preceq \sigma_{20} \\ (x_1 \lhd x_2) \end{array} \right\} \Rightarrow \exists\, \sigma_0,\ \exists\, k \left\{ \begin{array}{l} \mathcal{R}^2 = \mathcal{R}^{1|k} \\ \sigma_0 = (\mathcal{R}^{i_1}, 0, 0, \varepsilon_0, \mathcal{X}^{j_0}) \\ \sigma_0 \preceq \sigma_1 \preceq \sigma_{10} \\ (\mathcal{R}^{1|k}, 0, 0, \varepsilon_0, \mathcal{X}^{j_0|k}) \preceq \sigma_2 \preceq \sigma_{20} \end{array} \right.$$

Cas de la relation $\lhd\!:\!\rhd$

$$\left. \begin{array}{r} \sigma_1 = (\mathcal{R}^1, \delta, \alpha, \varepsilon_i, \mathcal{X}^1) \\ \sigma_2 = (\mathcal{R}^1, \delta, \alpha, \varepsilon_2, \mathcal{X}^2) \\ \sigma_{10} = (\mathcal{R}^1, \delta, \alpha, \varepsilon_{10}, \mathcal{X}^{10}) \\ \sigma_{20} = (\mathcal{R}^1, \delta, \alpha, \varepsilon_{20}, \mathcal{X}^{20}) \\ \sigma_1 \preceq \sigma_{10} \\ \sigma_2 \preceq \sigma_{20} \\ (x_1 \lhd\!:\!\rhd \ x_2) \end{array} \right\} \Rightarrow \exists\, \sigma_0,\ \exists\, k \left\{ \begin{array}{l} \sigma_0 = (\mathcal{R}^1, 0, 0, \varepsilon_0, \mathcal{X}^{j_0}) \\ \sigma_0 \preceq \sigma_1 \preceq \sigma_{10} \\ \mathcal{X}^{j_0} \subseteq (\mathcal{X}^1 \cap \mathcal{X}^2) \subseteq (\mathcal{X}^{10} \cap \mathcal{X}^{20}) \end{array} \right.$$

3.1.5 début de formalisation du jeu de cartes

Reprenons l'exemple du jeu de cartes du point de vue du joueur Henri :

- à chaque tour, Henri regarde son jeu issu du tour précédent,
- puis il cherche qui a déjà joué (joueurs de position inférieure),
- puis il regarde ce qui a été joué par ces joueurs,
- enfin il joue.

Nous allons formaliser le jeu à partir des notations suivantes :

$\mathcal{D}_X$ est l'ensemble des valeurs possibles pour la variable x =joueur

$\mathcal{D}_Y$ est l'ensemble des valeurs possibles pour la variable y =carte

$\mathcal{D}_Z$ est l'ensemble des valeurs possibles pour la variable z =tour

$\mathcal{D}_T$ est l'ensemble des valeurs possibles la variable t = position

$\mathcal{X}^{Henri} = \{Henri\} \subseteq \mathcal{D}_X$
$\mathcal{X}^{\neq Henri} = \{\neq Henri\} \subseteq \mathcal{D}_X$
$\mathcal{Z}^{t-1} = \{tour_{t-1}\} \subseteq \mathcal{D}_Z$
$\mathcal{Z}^{t} = \{tour_{t}\} \subseteq \mathcal{D}_Z$
$\mathcal{T}^{<Henri} = \{< position(Henri)\} \subseteq \mathcal{D}_T$
$\mathcal{R}^1{}_{\mathcal{D}_X \times \mathcal{D}_Y \times \mathcal{D}_Z}$ est définie par : $\mathcal{R}^1(x, y, z) \Leftrightarrow$ "x joue y au tour z"
$\mathcal{R}^2{}_{\mathcal{D}_X \times \mathcal{D}_Y \times \mathcal{D}_Z}$ est définie par : $\mathcal{R}^2(x, y, z) \Leftrightarrow$ "x a le jeu y au tour z"
$\mathcal{R}^3{}_{\mathcal{D}_X \times \mathcal{D}_T}$ est définie par : $\mathcal{R}^3(x, t) \Leftrightarrow$ "x a la position t"
$\mathcal{R}^4{}_{\mathcal{D}_Z}$ est définie par : $\mathcal{R}^4(z) \Leftrightarrow$ "tour z"

Henri reçoit à un momment du jeu l'information *tour* t qui correspond au data assertion réel :

$\sigma_1 = (\mathcal{R}^4, 0, 0, 0, \mathcal{Z}^t)$

Il émet alors les data question réels *Quels sont les joueurs placés avant moi ?* et *Quel est mon jeu à l'issue du tour* $t-1$ *?* :

$\sigma_{21} = (\mathcal{R}^3, 0, 1, 0, \mathcal{X}^{\neq Henri} \times \mathcal{T}^{<Henri})$
$\sigma_{22} = (\mathcal{R}^2, 0, 1, 0, \mathcal{X}^{Henri} \times \mathcal{D}_Y \times \mathcal{Z}^{t-1})$

Supposons que la réponse à la première question est que *Julie et Paulette sont placées avant Henri* :

$\sigma_{211} = (\mathcal{R}^3, 0, 0, 0, \mathcal{X}^{Julie} \times \mathcal{T}^{position1})$
$\sigma_{212} = (\mathcal{R}^3, 0, 0, 0, \mathcal{X}^{Paulette} \times \mathcal{T}^{position2})$

Henri émet les data question réels *Qu'a joué Julie au tour* t *? ; Qu'a joué Paulette au tour* t *?* :

$\sigma_{31} = (\mathcal{R}^1, 0, 1, 0, \mathcal{X}^{Julie} \times \mathcal{D}_Y \times \mathcal{Z}^t)$
$\sigma_{32} = (\mathcal{R}^1, 0, 1, 0, \mathcal{X}^{Paulette} \times \mathcal{D}_Y \times \mathcal{Z}^t)$

Lorsqu'il a la réponse ces deux questions, il met en oeuvre sa stratégie personnelle et joue *Henri joue ces cartes au tour* t *?* :

$\sigma_4 = (\mathcal{R}^1, 0, 0, 0, \mathcal{X}^{Henri} \times \mathcal{Y}^{this} \times \mathcal{Z}^t)$

Notons $\triangle_{Henri}$ le module de réécriture représentant Henri ; $\triangle_{Henri}$ aura pour déclencheur :

$\sigma_{10} = (\mathcal{R}^4, 1, 0, 1, \mathcal{D}_Z)$

$\triangle_{Henri}$ émettra des réductions des data virtuels :

$\sigma_{210} = (\mathcal{R}^3, 0, 1, 1, \mathcal{X}^{\neq Henri} \times \mathcal{T}^{<Henri})$
$\sigma_{220} = (\mathcal{R}^2, 0, 1, 1, \mathcal{X}^{Henri} \times \mathcal{D}_Y \times \mathcal{D}_Z)$
$\sigma_{30} = (\mathcal{R}^1, 0, 1, 1, \mathcal{X}^{\neq Henri} \times \mathcal{D}_Y \times \mathcal{D}_Z)$
$\sigma_{40} = (\mathcal{R}^1, 0, 0, 1, \mathcal{X}^{Henri} \times \mathcal{D}_Y \times \mathcal{D}_Z)$

Les relations suivantes apparaissent :

1. $\sigma_1 \in \mathcal{A}^{Real}$ $\quad \sigma_1 \lhd \sigma_{10}$
2. $\sigma_{21} \preceq \sigma_{210}$
3. $\sigma_{211} \in \mathcal{A}^{Real}$ $\quad \sigma_{211} \lhd \sigma_{21}$
4. $\sigma_4 \in \mathcal{A}^{Real}$ $\quad \sigma_4 \preceq \sigma_{40}$

1. exprime que σ_1 déclenche $\triangle_{Henri}$
2. exprime que σ_{21} est une réduction de σ_{210}
3. exprime que σ_{211} répond à σ_{21}
4. exprime que σ_4 est une réduction de σ_{40}

Pour compléter la formalisation du jeu de cartes, il suffit de créer les modules associés aux autres joueurs, puis de créer un module ”maître du jeu” qui donne les impulsions ”tour t” et redistribue les cartes posées sur la table lorsque tout le monde a joué ; nous nous en acquitterons ultérieurement.

3.2 réécriture des termes

La situation initiale d'un calcul sera la donnée d'un ensemble de termes et d'un ensemble de modules de réécriture, appelés ”pentominôs”. Au cours du calcul, les pentominôs ajouteront et retireront des termes ; à chaque instant, l'état du graphe de calcul déterminera le déclenchement ou l'inactivation des pentominôs.

Les pentominôs sont des algorithmes de réécriture qui fonctionnent par étapes, en réduisant à chaque étape les termes du graphe de calcul.

Nous allons d'abord définir les mécanismes de cette réduction, grâce à la notion de réducteur.

Ensuite nous définirons les pentominôs qui sont les modules autonomes de réécriture.

Ensuite nous définirons les primitives de la réécriture.

Puis nous définirons la notion de graphe de calcul et montrerons comment le calcul est piloté par des événements du graphe de calcul.

Enfin nous assemblerons tous ces éléments pour décrire le fonctionnement des pentominôs.

3.2.1 la notion de réducteur

Les réducteurs sont des algorithmes de réécriture particuliers : ils réduisent des assertions et des requêtes, en fonction de paramètres qui sont eux mêmes des termes du graphe de calcul. Deux cas seront mis en pratique :

- la réduction d'un ensemble initial de data, en fonction d'un unique data réel ; la donnée de n data réels produira n réductions des data exécutées en parallèle. On parlera donc de *réduction parallèle*.
- la réduction d'un ensemble de data, en fonction d'un ensemble d'assertions réelles ; la donnée de n assertions réelles produira une unique réduction globale des data, destinée à s'inscrire dans une séquence de réduction successives. On parlera alors de *réduction séquentielle*.

Dans les deux cas, les data réduits, de même que les data paramètres de la réduction, interviendront uniquement à travers la structure relationnelle des syntagmes associés ; ce qui fait qu'un réducteur sera défini "à une réduction près" et pourra intervenir en cascade au cours du calcul.

réduction parallèle

Reprenons l'exemple de Henri : il reçoit à un momment du jeu l'information *tour* t :

$\sigma_1 = (\mathcal{R}^4, 0, 0, 0, \mathcal{Z}^t) \lhd \sigma_{10}$

Il émet alors les data question réels *Quels sont les joueurs placés avant moi ?* et *Quel est mon jeu placés à l'issue du tour* $t-1$ *?* :

$\sigma_{21} = (\mathcal{R}^3, 0, 1, 0, \mathcal{X}^{\neq Henri} \times \mathcal{T}^{<Henri}) \preceq \sigma_{210}$

$\sigma_{22} = (\mathcal{R}^2, 0, 1, 0, \mathcal{X}^{Henri} \times \mathcal{D}_Y \times \mathcal{Z}^{t-1}) \preceq \sigma_{220}$

La donnée d'une assertion réelle a provoqué la réduction de deux questions virtuelles. Durant cette transformation, c'est la valeur de σ_1 qui a joué le rôle d'input et qui a déterminé les valeurs de σ_{21} (inchangée : la seule transformation pour la requête est le passage de virtuel à réel) et σ_{22} (la requête porte sur le tour $t-1$). L'opérateur qui a permis cette transformation des valeurs se définit à partir des structures relationnelles en jeu : $\mathcal{R}^4$ pour la variable input , $\mathcal{R}^3$ et $\mathcal{R}^2$ pour les variables output. Nous pouvons donner une définition générale pour ce type de tranformation :

Paral$_{i_0,\,I}$

Soient :

une structure relationnelle input $\mathcal{R}^{i_0}$

un ensemble de structures relationnelles output $\{\mathcal{R}^i\}_{i\in I}$

On appelle *réducteur parallèle* et on note Paral$_{i_0,\,I}$ une fonction $f_{\mathcal{R}^{i_0},\,\{\mathcal{R}^i\}}$ vérifiant :

$\forall\, J$ ensemble d'indices ;
$\forall\, Y = \{(\mathcal{R}^{i_0}, val(\sigma^j))\}_{j\in J}$ ensemble de syntagmes ;
$\forall\, K$ ensemble d'indices ;
$\forall\, Y_1 = \{(\mathcal{R}^i, val(\sigma^i_k))\}_{i\in I;\, k\in K}$ ensemble de syntagmes :

$$(Y, Y_1) \overset{f_{\mathcal{R}^{i_0},\,\{\mathcal{R}^i\}}}{\longrightarrow} \{Y_2^j = f(val(\sigma^j), \{val(\sigma^i_k)\}_{i\in I;\, k\in K})\}_{j\in J}$$

avec :

Y_2^j ensemble fini de syntagmes $\{\sigma_2^j$ — $\exists\, \sigma_k^i \in Y_1$ tel que : $\sigma_2^j \preceq \sigma_k^i\}$

réduction séquentielle

Lorsque Henri reçoit à un momment du jeu l'information *Julie et Paulette sont placées avant Henri* :

$\sigma_{211} = (\mathcal{R}^3, 0, 0, 0, \mathcal{X}^{Julie} \times \mathcal{T}^{position1}) \preceq \sigma_{210}$
$\sigma_{212} = (\mathcal{R}^3, 0, 0, 0, \mathcal{X}^{Paulette} \times \mathcal{T}^{position2}) \preceq \sigma_{210}$

Il émet les data question réels *Qu'a joué Julie au tour t ? ; Qu'a joué Paulette au tour t ?* :

$\sigma_{31} = (\mathcal{R}^1, 0, 1, 0, \mathcal{X}^{Julie} \times \mathcal{D}_Y \times \mathcal{Z}^t) \preceq \sigma_{30}$
$\sigma_{32} = (\mathcal{R}^1, 0, 1, 0, \mathcal{X}^{Paulette} \times \mathcal{D}_Y \times \mathcal{Z}^t) \preceq \sigma_{30}$

La donnée d'un ensemble d'assertion réelles a provoqué la réduction d'une question virtuelle. Durant cette transformation, ce sont les valeurs de σ_{211} et σ_{212} qui ont joué le rôle d'input et qui ont déterminé les valeurs de σ_{31} (Julie ,tour t) et σ_{220} (Paulette ,tour t). L'opérateur qui a permis cette transformation des valeurs se définit à partir des structures relationnelles en jeu $\mathcal{R}^3$ et $\mathcal{R}^1$. Nous pouvons donner une définition générale pour ce type de tranformation :

Seq$_{J,\,I}$

Soient :

un ensemble de structures relationnelles input $\{\mathcal{R}^j\}_{j\in J}$

un ensemble de structures relationnelles output $\{\mathcal{R}^i\}_{i\in I}$

On appelle *réducteur séquentiel* et on note Seq$_{J,\,I}$ une fonction $f_{\{\mathcal{R}^j\},\,\{\mathcal{R}^i\}}$ vérifiant :

$\forall\, K, \forall\, L$ ensembles d'indices ;
$\forall\, Y = \{(\mathcal{R}^j, val(\sigma_k^j))\}_{j\in J;\, k\in K}$ ensemble de syntagmes ;
$\forall\, Y_1 = \{(\mathcal{R}^i, val(\sigma_l^i))\}_{i\in I;\, l\in L}$ ensemble de syntagmes :

$$(Y, Y_1) \overset{f_{\{\mathcal{R}^j\},\,\{\mathcal{R}^i\}}}{\longrightarrow} Y_2 = f(\cup_{j;\,k}\{val(\sigma_k^j)\}, \cup_{i;\,l}\{val(\sigma_l^i)\})$$

avec :

Y_2 ensemble fini de syntagmes $\{\sigma_2^m$ — $\exists\, \sigma_l^i \in Y_1$ tel que : $\sigma_2^m \preceq \sigma_l^i\}$

liberté de réduction

Les réducteurs ainsi définis permettent d'accueillir une large famille d'algorithmes :

Les réducteurs parallèles pourront implémenter toutes formes de règles "si ... alors" ;

Les réducteurs séquentiels pourront les compléter pour implémenter des algorithmes à étapes, comme celui qui permet à un joueur de jouer un coup après avoir conduit une analyse ;

Il deviendra alors possible d'implémenter par exemple des règles du type "si A et non B, alors ...", au moyen d'un réducteur parallèle déclenché par "A" et qui pose la requête "B".

3.2.2 définition des pentominôs

Nous disposons maintenant de tous les éléments pour définir les "modules autonomes de réécriture".

Définition :

Un pentominô $\triangle\!\!\triangle_i = (\delta_i, Y_i^0, (f_i^n)_{n\in N_i})$ est défini par :

un ensemble de structures relationnelles $\{\mathcal{R}^j\}_{j\in J_i}$

un ensemble Y_i^0 de syntagmes comportant :

un trigger *déclencheur* $\delta_i = (\mathcal{R}^{j_0}, 1, \alpha_{j_0}, \varepsilon_{j_0}, \mathcal{X}^{j_0})$

des data *effecteurs* $(\mathcal{R}^j, 0, \alpha_{j,\,k}, \varepsilon_{j,\,k}, \mathcal{X}^{j,\,k})$ avec $j \in J_i;\ k \in K_i$

une suite de réducteurs $(f^n)_{n\in N_i}$ telle que :

f^0 est un réducteur parallèle $\text{Paral}_{i_0,\,I}$

$\forall\, n > 0$, f^n est un réducteur séquentiel $\text{Seq}_{X_n,\,J_i}$ où $X_n \subseteq J_i$

J_i, K_i, N_i sont des ensembles d'indices

3.2.3 primitives de la réécriture

Au cours du calcul, chaque pentominô va agir sur les termes en s'ajoutant et en se retirant des listes associées aux syntagmes ; précisons ces mécanismes élémentaires :

$\triangle\!\!\triangle_i$ ENTER (S_1)

Pour chaque syntagme $\sigma_j =\in S_1$:

- si le graphe de calcul ne contient aucun terme identifié par le syntagme σ_j, le terme $(\sigma_j, L_j = \emptyset)$ est créé.
- si le pentominô $\triangle\!\!\triangle_i$ n'est pas présent dans la liste L_j du terme (σ_j, L_j), il s'ajoute.

$\triangle\!\!\triangle_i$ EXIT (S_1)

Pour chaque syntagme NON RÉEL $\sigma_j =\in S_1$:

- le pentominô $\triangle\!\!\triangle_i$ est retiré de la liste L_j du terme (σ_j, L_j).
- si $L_j = \emptyset$, le terme (σ_j, L_j) est retiré du graphe de calcul.

3.3 graphe de calcul et pilotage de la réécriture

Dans ce chapitre, nous allons nous intéresser aux réécritures successives d'un ensemble de termes initiaux par un ensemble de pentominôs.

3.3.1 définition du graphe de calcul

Ceci nous permet de prendre la définition suivante :

Définition :

On appelle *calcul* la donnée d'un ensemble fini de pentominôs $\{\triangle\!\!\!\triangle_I\}_{i\in I}$ contenant un pentominô particulier sans déclencheur $\triangle\!\!\!\triangle_0 = (\ , Y_0^0 \in Real,\ )$.

On appelle *graphe de calcul* le graphe orienté dont les sommets sont les termes du calcul à un moment quelconque du processus de réécriture, et dont les arcs sont donnés par la relation "influence".

$$\mathcal{G}_I^t = \text{ état } t \text{ du graphe de calcul lié à } \{\triangle\!\!\!\triangle_I\}_{i\in I}$$

On appelle $\mathcal{G}_I^0$ l'état du graphe de calcul lorsque chaque pentominô $\triangle\!\!\!\triangle_i$ a effectué l'écriture initiale de ses effecteurs : $\triangle\!\!\!\triangle_i$ ENTER (Y_i^0)

$\mathcal{G}_I^0$ est le point de départ du processus de réécriture.

Les règles de réécriture seront toutes du type "si le graphe de calcul présente la propriété p, alors il est transformé par l'application d'un ou plusieurs réducteurs f_i^n". Nous parlerons donc de *prémisses graphiques* pour les prémisses des règles de réécriture, et d'*événements de réécriture* pour leurs conclusions. Le calcul enchaînera automatiquement des règles comme un système expert : les prémisses graphiques étant testées après chaque événement de réécriture.

prémisses graphiques

Voici les trois types de "formes locales" du graphe qui seront testées après chaque événement de réécriture pour déclencher éventuellement d'autres événements d'écriture :

1. un ensemble de data réels influence un trigger.

2. un ensemble de requêtes a une réponse réelle (est tel que les assertions qui l'influencent sont toutes réelles).

3. un trigger est réalisé (est tel que les termes qui l'influencent sont tous réels).

Nous donnerons à chacune de ces prémisses graphiques une définition précise au paragraphe "zones influentes", et montrerons qu'elles correspondent toutes à des changements irréversibles du graphe de calcul, ce qui va permettre de distinguer une chronologie dans les états successifs du graphe de calcul.

événements de réécriture

A chacune des prémisses listées ci-dessus correspondra un événement de réécriture :

1. le(s) pentominô(s) $\triangle_i$ associé(s) au trigger déclenche(nt) leur réducteur parallèle.
2. le(s) réducteur(s) séquentiel(s) associé(s) à l'ensemble de requêtes se déclenche(nt).

3. le(s) pentominô(s) $\triangle_i$ associé(s) au trigger se retire(nt) du calcul.

Nous allons décrire plus en détail ce fonctionnement au paragraphe suivant ; mais il nous faut d'abord préciser la notion de "zone influente" pour un syntagme.

zones influentes

Nous allons nous intéresser à des "zones" du graphe de calcul capables d'influencer des assertions ou des requêtes ; pour cela, nous allons considérer deux applications Φ et Ψ :
Φ est définie de façon différente sur les assertions et sur les requêtes :

Définition de Φ :

Soit $\{\triangle_I\}_{i\in I}$ un calcul, et $\mathcal{G}_I^t$ l'état du graphe de calcul à un moment t :
si a_0 est une assertion : $\Phi(a_0, \mathcal{G}_I^t) = \{\ a \in A \cap \mathcal{G}_I^t \text{ — } a \lhd\ \ a_0\ \}$
On notera $\Phi_{Real}(a_0, \mathcal{G}_I^t) = \Phi(a_0, \mathcal{G}_I^t) \cap Real$
si r_0 est une requête : $\Phi(r_0, \mathcal{G}_I^t) = \{\ r \in R \cap \mathcal{G}_I^t \text{ — } r \lhd\!:\!\rhd\ r_0\ \}$

Ψ est définie sur les seules requêtes, et leur associe les assertions du graphe de calcul qui les influencent :

Définition de Ψ :

Soit $\{\triangle_I\}_{i\in I}$ un calcul, et $\mathcal{G}_I^t$ l'état du graphe de calcul à un moment t :
si r_0 est une requête : $\Psi(r_0, \mathcal{G}_I^t) = \{\ a \in A \cap \mathcal{G}_I^t \text{ — } a \lhd\ \ r_0\ \}$
si R_0 est un ensemble de requêtes : $\Psi(R_0) = \cup_{r\in R_0}\Psi(r, \mathcal{G}_I^t)$
On notera $\Psi_{Real}(R_0, \mathcal{G}_I^t) = \Psi(R_0, \mathcal{G}_I^t) \cap Real$

En début de calcul, un certain nombre (fini) de termes sont présents, qui par réécritures successives vont être remplacés par d'autres (aux syntagmes réduits par rapport aux premiers). Une conséquence importante du Lemme 1 est que les zones de calcul $\Phi(x_i)$ ou $\Psi(r_i)$ évolueront de façon agréable au fil des réécritures successives : le calcul ne produira pas de nouvelles influences et la réduction progressive des termes conduira (peut-être) à des termes réels ; ceci nous amène à poser quelques nouvelles définitions :

formalisation des prémisses graphiques :

Soit x un terme et δ_0 un déclencheur, X *réalise* δ_0
si et seulement si :

$$x \epsilon \Phi_{Real}(\delta_0, \mathcal{G}_I^t)$$

Par ailleurs, "les termes influençant δ_0 sont tous réels" s'écrit :

$$\Phi_{Real}(\delta_0, \mathcal{G}_I^t) = \Phi(\delta_0, \mathcal{G}_I^t)$$

Enfin, si A_1 est un ensemble d'assertions et R_1 un ensemble de requêtes,
A_1 *résoud* R_1 si et seulement si :

$$A_1 = \Psi_{Real}(R_1, \mathcal{G}_I^t) = \Psi(R_1, \mathcal{G}_I^t)$$

Lemme 2 :

Au cours d'un calcul $\{\triangle\!\!\!\triangle_I\}_{i \in I}$ dont les seuls événements de réécriture sont :

$$\triangle\!\!\!\triangle_i \ \text{ENTER}\ (Y) \text{ avec : } Y \subseteq \{\preceq \mathcal{G}_I^t\}$$

$$\triangle\!\!\!\triangle_i \ \text{EXIT}\ (Z)$$

les prémisses graphiques qui apparaissent sont conservées par les réécritures successives.

démonstration de "Lemme 2" :

$x \epsilon \Phi_{Real}(\delta_0, \mathcal{G}_I^t)$ est conservée car les termes réels ne sont jamais effacés.

$\Phi_{Real}(\delta_0, \mathcal{G}_I^t) = \Phi(\delta_0, \mathcal{G}_I^t)$ est conservée car d'après Lemme 1, il ne peut pas apparaître de terme non réel dans $\Phi(\delta_0, \mathcal{G}_I^{t_2})$ s'il n'y en avait pas dans $\Phi(\delta_0, \mathcal{G}_I^{t_1}); t_1 < t_2$

en combinant les deux raisons précédentes, $A_1 = \Psi_{Real}(R_1, \mathcal{G}_I^t) = \Psi(R_1, \mathcal{G}_I^t)$ est également conservée.

3.3.2 fonctionnement du pentominô

Soit $\{\triangle\!\!\!\triangle_I\}_{i \in I}$ un calcul, avec $\mathcal{G}_I^0$ l'état initial du graphe de calcul.

Soit $\triangle\!\!\!\triangle_i = (\delta_i, Y_i^0, (f_i^n)_{n \in N_i})$ un pentominô du calcul :

Avant le début du calcul : $\triangle\!\!\!\triangle_i \ \text{ENTER}\ (Y_i^0)$

A un moment du calcul, l'événement graphique "$\{x_j\}_{j \in J} \in Real \cap \Phi(\delta_i)$" apparaît , le réducteur f_i^0 est appliqué en parallèle pour chaque x_j :

$(x_j, Y_i^0) \overset{f_i^0}{\longmapsto} Y_i^1(x_j)$, et le graphe de calcul est transformé par :

$$\triangle\!\!\!\triangle_i \ \text{ENTER}\ (Y_i^1(x_j))$$

On parlera alors d'*(*invocations) en parallèle du pentominô $\mathbb{\triangle}_i$. En particulier, l'invocation déclenchée par x_j a produit l'ensemble de requêtes réelles $R_i^1(x_j)$.

Lorsque l'événement graphique "$\Psi(R_i^1(x_j)) \subseteq Real$" apparaît, le réducteur f_i^1 est alors appliqué aux assertions $\Psi_{Real}(R_i^1(x_j), \mathcal{G}_I^t)$ répondant aux requêtes et aux $Y_i^1(x_j)$:

$(\Psi_{Real}(R_i^1(x_j), \mathcal{G}_I^t), Y_i^1(x_j)) \stackrel{f_i^1}{\longmapsto} (Y_i^2(x_j))$, et le graphe de calcul est transformé par :

$$\mathbb{\triangle}_i \text{ EXIT } (Y_i^1(x_j)) \quad \mathbb{\triangle}_i \text{ ENTER } (Y_i^2(x_j))$$

...

Lorsque l'événement graphique "$\Psi(R_i^k(x_j)) \subseteq Real$" apparaît, le r éducteur f_i^k est appliqué :

$(\Psi_{Real}(R_i^k(x_j), \mathcal{G}_I^t), Y_i^k(x_j)) \stackrel{f_i^k}{\longmapsto} (Y_i^{k+1}(x_j))$, et le graphe de calcul est transformé par :

$$\mathbb{\triangle}_i \text{ EXIT } (Y_i^k(x_j)) \quad \mathbb{\triangle}_i \text{ ENTER } (Y_i^{k+1}(x_j))$$

...

Lorsque l'événement graphique "$\Psi(R_i^n(x_j)) \subseteq Real$" apparaît, le réducteur f_i^n est appliqué :

$(\Psi_{Real}(R_i^n(x_j), \mathcal{G}_I^t), Y_i^n(x_j)) \stackrel{f_i^n}{\longmapsto} (Y_i^{n+1}(x_j))$, et le graphe de calcul est transformé par :

$$\mathbb{\triangle}_i \text{ EXIT } (Y_i^n(x_j)) \quad \mathbb{\triangle}_i \text{ ENTER } (Y_i^{n+1}(x_j))$$

Enfin, à un moment du calcul, l'événement graphique $\phi(\delta_i) \in Real$ peut apparaitre, ce qui signifie que tous les termes qui influencent le déclencheur de $\mathbb{\triangle}_j$ sont réels $\mathbb{\triangle}_i$ est alors désactivé :

$$\mathbb{\triangle}_i \text{ EXIT } (Y_i^0)$$

3.3.3 les règles de réécriture

Soit un calcul $\{\mathbb{\triangle}_I\}_{i\in I}$, de graphe "instantané" $\mathcal{G}_I^t$, et dont les pentominôs $\mathbb{\triangle}_i = (\delta_i, Y_i^0, (f_i^n)_{n\in N_i})$ sont définis par :

- un ensemble de structures relationnelles $\{\mathcal{R}^j\}_{j\in J_i}$
- un ensemble Y_i^0 de syntagmes comportant :
  - un trigger *déclencheur* $\delta_i = (\mathcal{R}^{j_0}, 1, \alpha_{j_0}, \varepsilon_{j_0}, \mathcal{X}^{j_0})$
  - des data *effecteurs*
- une suite de réducteurs $(f^n)_{n\in N_i}$ telle que :
  - f^0 est un réducteur parallèle $\text{Paral}_{i_0,\, I}$
  - $\forall\, n > 0$, f^n est un réducteur séquentiel $\text{Seq}_{X_n,\, J_i}$

Alors les *événements du calcul* sont les applications des trois règles qui suivent :

RÈGLE START : $[x_j \, \epsilon \, \Phi_{Real}(\delta_i, \mathcal{G}_I^t)] \Rightarrow [start\ (\mathbb{\Delta}_i, 0, x_j)]$

Lorsqu'un terme réel x_j réalise le déclencheur δ_i
du pentominô $\mathbb{\Delta}_i = (\delta_i, Y_i^0, (f_i^n)_{n \in N_i})$, cela produit l'événement de réécriture :

$$start\ (\mathbb{\Delta}_i, 0, x_j) \quad : \quad \triangle\!\!\triangle_i \ \text{ENTER}\ (Y_i^1(x_j) = f_i^0(x_j, Y_i^0))$$

la transformation résultante pour $\mathcal{G}_I^t$ est fonction uniquement des données initiales $\{\mathbb{\Delta}_I\}_{i \in I}$ et de x_j

RÈGLE STEP : $[\Psi_{Real}(R_1^k(x_j), \mathcal{G}_I^t) = \Psi(R_1^k(x_j), \mathcal{G}_I^t)] \Rightarrow [step\ (\mathbb{\Delta}_i, k, x_j)]$

Lorsque l'ensemble des requêtes émises $\triangle\!\!\triangle_i^k\ (j)$ est résolu, cela produit les événements de réécriture :

$$step\ (\mathbb{\Delta}_i, k, x_j) : \begin{array}{l} \triangle\!\!\triangle_i \ \text{ENTER}\ (Y_i^{k+1}(x_j) = f_i^k(\Psi_{Real}(R_i^k(x_j), \mathcal{G}_I^t), Y_i^k(x_j))) \\ \triangle\!\!\triangle_i \ \text{EXIT}\ (Y_i^k(x_j)) \end{array}$$

RÈGLE STOP : $[\Phi_{Real}(\delta_i, \mathcal{G}_I^t) = \Phi(\delta_i, \mathcal{G}_I^t)] \Rightarrow [stop\ (\mathbb{\Delta}_i)]$

Lorsque les termes influençant le trigger δ_i
du pentominô $\mathbb{\Delta}_i = (\delta_i, Y_i^0, (f_i^n)_{n \in N_i})$ sont tous réels, cela produit l'événement de réécriture :

$$stop\ (\mathbb{\Delta}_i) \quad : \quad \triangle\!\!\triangle_i \ \text{EXIT}\ (Y_i^0)$$

la transformation résultante pour $\mathcal{G}_I^t$ est fonction uniquement des données initiales $\{\mathbb{\Delta}_I\}_{i \in I}$

Chapitre 4

propriétés du calcul

4.1 terminaison du calcul

4.1.1 une condition suffisante de terminaison

Définition :

Soient $\triangle\!\!\!\triangle_1 = (\delta_1, Y_1^0, (f_1^n)_{n\in N_1})$ et $\triangle\!\!\!\triangle_2 = (\delta_2, Y_2^0, (f_2^n)_{n\in N_2})$ deux pentominôs ; on dira que $\triangle\!\!\!\triangle_1$ *commande* $\triangle\!\!\!\triangle_2$ si et seulement si :

$$Y_1^0 \cap \Phi(\delta_2) \neq \emptyset$$

Théorème 1 :

Soit un calcul "$\{\triangle\!\!\!\triangle_I\}_{i\in I}$", soit $\mathcal{G}_{Commande}$ le graphe obtenu à partir de l'ensemble des pentominôs du calcul muni de la relation "*commande*".
Si $\mathcal{G}_{Commande}$ est sans circuit, le calcul s'arrête en un nombre de réécritures fini.
On note alors $\mathcal{G}_I^\infty$ l'état final du graphe de calcul.

démonstration de "Théorème 1" :

Si le graphe de commande est sans circuit, nous pouvons attribuer à chaque pentominô un " rang " de telle sorte qu'un pentominô ne commande que des pentominôs de rang strictement supérieur. Auront le rang "0" $\mathbb{\Delta}_0$ ainsi que les pentominôs qui ne sont pas commandés. , ... auront le rang "k" les pentominôs commandées par des pentominôs de rang "$< k$" au maximum.
Si p est le nombre total de pentominôs, le rang maximal est p.

Par récurrence sur "n" : [le nombre de déclenchements de pentominôs de rang "n" est fini].

La propriété est vraie au rang "0" car les pentominôs de rang "0" ne seront jamais déclenchés ; en effet :
- $\mathbb{\Delta}_0 = (\ , Y_0^0 \in Real,\ )$ ne sera jamais déclenché par définition
- soit $\mathbb{\Delta}_{i_0} = (\delta_{i_0}, Y_{i_0}^0, (f_{i_0}^n)_{n \in N_{i_0}})$ un autre pentomino de rang "0"
- soit y_1 un terme vérifiant $y_1 \in \Phi_{Real}(\delta_{i_0}, \mathcal{G}_I^t)$ produit lors du calcul
- alors $\exists j, \exists y_2 \in \mathcal{G}_I^0$ — $\begin{cases} y_2 \in Y_j^0 \\ y_1 \preceq y_2 \end{cases}$
- d'après Lemme 1 : $y_2 \in \Phi\delta_{i_0}$
- et donc $\mathbb{\Delta}_j$ commande $\mathbb{\Delta}_{i_0}$.

Supposons la propriété vraie au rang "n". Soit x_j un terme nécessaire pour déclencher un pentominô $\mathbb{\Delta}_i = (\delta_i, Y_i^0, (f_i^n)_{n \in N_i})$ de rang "$n+1$" :

$$x_j \in \Phi_{Real}(\delta_i, \mathcal{G}_I^t)$$

- si $x_j \in \mathcal{G}_I^0$, il est à prendre dans un ensemble fini
- sinon, x_j a été produit par un pentominô $\mathbb{\Delta}_j$ mais le raisonnement précédent prouve alors que $\mathbb{\Delta}_j$ commande $\mathbb{\Delta}_i$ et donc $\mathbb{\Delta}_i$ est de rang "$\leq n$" ; $x_j \in \mathcal{G}_I^0$ est donc à prendre dans un ensemble fini.

4.1.2 un cas de non terminaison

L'exemple qui suit montre que dans certains cas, une boucle dans le graphe de commande engendre un calcul qui ne s'arrête jamais.

Prenons $S = \mathcal{P}(N)$. Les deux ordres sont identiques : il s'agit de l'inclusion. Les éléments terminaux sont les entiers naturels considérés comme ensembles à un élément Soit l'assertion $a = (\alpha, N)$ dont les réalisations sont de la forme $a_i = (\alpha, n_i)$. Considérons le pentominô défini par :

déclencheur : (α, N)
$Y^0 = \{(\alpha, N)\}$: l'unique élément est le déclencheur
f^0 défini par $(a_i, Y^0) \xmapsto{f^0} (Y^1)$ avec $Y^1 = \{(\alpha, N), a_i\}$
la suite de réducteurs séquentiels est vide

Si le terme réel a_0 est présent au départ du calcul, alors a_0 déclenche une invocation qui produit a_1, qui déclenche une invocation qui produit a_2 ...

4.2 confluence du calcul

Dans cette section, nous allons montrer une propriété essentielle du calcul : lorsqu'un calcul $\{\mathbb{\Delta}_I\}_{i \in I}$ s'arrête, l'état final $\mathcal{G}_I^\infty$ du graphe de calcul ne dépend que de son état initial $\mathcal{G}_I^0$.

Définitions :

Soit un calcul $\{\mathbin{\triangle\!\!\!\triangle}_I\}_{i\in I}$, nous appellerons *transition* une transformation du graphe de calcul $\mathcal{G}_I^t$ identifiée par :
- une règle : START, STEP ou STOP
- son contexte d'application formé d'un pentominô et des paramètres éventuels x_j et k

Bien que les règles ne soient pas "consommatrices" des termes du graphe de calcul qui les déclenchent, chaque transition aura lieu une seule fois, lorsqu'un changement d'état du graphe de calcul fait apparaître une prémisse graphique pour le couple (règle, contexte).

D'après Lemme 2, ces prémisses sont conservées une fois qu'elles apparaissent ; nous n'avons donc pas à imposer a priori un ordre pour les transitions.

Nous reprendrons pour les transitions les notations utilisées précédemment :
- $start\ (\mathbin{\triangle\!\!\!\triangle}_i, 0, x_j)$
- $step\ (\mathbin{\triangle\!\!\!\triangle}_i, k, x_j)$
- $stop\ (\mathbin{\triangle\!\!\!\triangle}_i)$

Une transition fait passer le graphe de calcul d'un état $\mathcal{G}_I^{avant}$ à l'état $\mathcal{G}_I^{après}$.

Nous noterons $step\ (\mathbin{\triangle\!\!\!\triangle}_i, k, x_j) = \mathcal{G}_I^{t_1 \to t_2}$ pour exprimer que l'application de la règle STEP au pentominô $\mathbin{\triangle\!\!\!\triangle}_i$ pour l'étape k de son invocation due à x_j fait passer le graphe de calcul de l'état $\mathcal{G}_I^{t_1}$ à l'état $\mathcal{G}_I^{t_2}$.

Un calcul $\{\mathbin{\triangle\!\!\!\triangle}_I\}_{i\in I}$ qui s'arrête a pour *journal* une suite finie de transitions reliant $\mathcal{G}_I^0$ à $\mathcal{G}_I^\infty$.

Par définition des mécanismes de réécriture, les transitions de type START ou STEP relatives à une même invocation de pentominô sont naturellement ordonnées chronologiquement par l'exposant du réducteur qui est appliqué ; par contre ces mécanismes ne disent rien a priori sur l'ordre d'apparition de transitions correspondant à des invocations différentes. Nous allons pourtant dégager des "points de rendez-vous" obligés qui vont garantir la confluence du calcul.

Dans la suite de ce paragraphe, nous prenons les notations suivantes :
- $\forall i : \mathbin{\triangle\!\!\!\triangle}_i = (\delta_i, Y_i^0, (f_i^n)_{n\in N_i})$
- $start\ (\mathbin{\triangle\!\!\!\triangle}_i, 0, x_j) = \mathcal{G}_I^{t_1 \to t_2}$ produit $\mathbin{\triangle\!\!\triangle}_i$ ENTER $(Y_i^1(x_j))$ qui contient les requêtes réelles $R_i^1(x_j)$; cette transition est déclenchée par $x_j \in \Phi_{Real}(\delta_i, \mathcal{G}_I^{t_1})$
- $step\ (\mathbin{\triangle\!\!\!\triangle}_i, k, x_j) = \mathcal{G}_I^{t_1 \to t_2}$ produit $\mathbin{\triangle\!\!\triangle}_i$ ENTER $(Y_i^{k+1}(x_j))$ qui contient les requêtes réelles $R_i^{k+1}(x_j)$ et $\mathbin{\triangle\!\!\triangle}_i$ EXIT $(Y_i^k(x_j))$; cette transition est déclenchée par $\Psi_{Real}(R_i^k(x_j), \mathcal{G}_I^{t_1}) = \Psi(R_i^k(x_j), \mathcal{G}_I^{t_1})$
- $stop(\mathbin{\triangle\!\!\!\triangle}_i) = \mathcal{G}_I^{t_1 \to t_2}$ produit $\mathbin{\triangle\!\!\!\triangle}_i$ EXIT (Y_i^0) ; cette transition est déclenchée par $\Phi_{Real}(\delta_i, \mathcal{G}_I^{t_1}) = \Phi(\delta_i, \mathcal{G}_I^{t_1})$

Lemme 3 :

Soit $\{\mathbb{A}_I\}_{i\in I}$ un calcul d'état initial $\mathcal{G}_I^0$ qui s'achève sur l'état final $\mathcal{G}_I^\infty$.

Il est possible d'attribuer un rang aux différentes transitions présentes dans le journal de ce calcul en procédant de la façon suivante :

- les transitions $start\ (\mathbb{A}_i, 0, x_j)$ telles que $x_j \in \mathcal{G}_I^0$ ont le rang "0"
- les transitions $stop\ (\mathbb{A}_i)$ telles que $\Phi(\delta_i, \mathcal{G}_I^0) \cap Virtual = \emptyset$ ont le rang "0"
- les $start\ (\mathbb{A}_i, 0, x_j)$ dont la prémisse graphique est apparue au rang "0" ont le rang "1"
- les $stop\ (\mathbb{A}_i)$ et les $step\ (\mathbb{A}_i, j, x_k)$ dont la prémisse graphique est réalisée lorsque toutes les transitions de rang "0" ont eu lieu, tout en étant *nécessairement postérieure* à l'une d'entre elles au moins, ont le rang "1"
- ...
- les $start\ (\mathbb{A}_i, 0, x_j)$ dont la prémisse graphique est apparue au rang "k" ont le rang "$k+1$"
- les $stop\ (\mathbb{A}_i)$ et les $step\ (\mathbb{A}_i, j, x_k)$ dont la prémisse graphique est réalisée lorsque toutes les transitions de rang "k" ont eu lieu, tout en étant *nécessairement postérieure* à l'une d'entre elles au moins, ont le rang "$k+1$"
- ... [a]

De plus chaque transition ainsi classée applique son réducteur à des syntagmes tous issus de transitions de rang inférieur.

[a] il est sous-entendu que deux étapes correspondant à deux réductions successives au sein d'un même pentominô auront des rangs différents ; en effet la prémisse de la seconde n'existe qu'à partir du moment où la première est achevée.

démonstration de "Lemme 3" :

La relation *nécessairement postérieure* a une définition syntaxique sur la base suivante :

soit $\mathcal{G}_I^{t_1 \rightarrow t_2} = stop\,(\triangle\!\!\!\triangle_{i_1})$, $stop\,(\triangle\!\!\!\triangle_{i_2})$ et $step\,(\triangle\!\!\!\triangle_{i_3}, k, x_j) : k > 0$ trois transitions :

$stop(\triangle\!\!\!\triangle_{i_2})$ est *nécessairement postérieure* à $stop(\triangle\!\!\!\triangle_{i_1})$ si et seulement si il existe un syntagme non réel σ_2 dans $\Phi(\delta_i, \mathcal{G}_I^{t_1}) \cap Y_{i_1}^0$; en effet $\triangle\!\!\!\triangle_{i_1}$ EXIT (σ_2) devra nécessairement précéder $stop\,(\triangle\!\!\!\triangle_{i_2})$ pour que la prémisse $\Phi_{Real}(\delta_i, \mathcal{G}_I^t) = \Phi(\delta_i, \mathcal{G}_I^t)$ apparaisse.

$step\,(\triangle\!\!\!\triangle_{i_3}, k, x_j)$ est *nécessairement postérieure* à $stop\,(\triangle\!\!\!\triangle_{i_1})$ si et seulement si il existe un syntagme non réel σ_3 dans $\Psi(R_{i_3}^{k-1}(x_j), \mathcal{G}_I^{t_1}) \cap Y_{i_1}^0$; en effet $\triangle\!\!\!\triangle_{i_1}$ EXIT (σ_3) devra nécessairement précéder $step\,(\triangle\!\!\!\triangle_{i_3}, k, x_j)$ pour que la prémisse $\Psi_{Real}(R_{i_3}^k(x_j), \mathcal{G}_I^t) = \Psi(R_{i_3}^k(x_j), \mathcal{G}_I^t)$ apparaisse.

Soit $\mathcal{G}_I^{t_1 \rightarrow t_2} = step\,(\triangle\!\!\!\triangle_{i_1}, k_1, x_{j1}); k_1 > 0$, $stop\,(\triangle\!\!\!\triangle_{i_2})$ et $step\,(\triangle\!\!\!\triangle_{i_3}, k_3, x_{j3}) : k_3 > 0$ trois transitions :

$stop\,(\triangle\!\!\!\triangle_{i_2})$ est *nécessairement postérieure* à $step\,(\triangle\!\!\!\triangle_{i_1}, j_1, x_{k_1})$ si et seulement si il existe un syntagme non réel σ_2 dans $\Psi(\delta_i, \mathcal{G}_I^{t_1}) \cap Y_{i_1}^{k_1}$; en effet $\triangle\!\!\!\triangle_{i_1}$ EXIT (σ_2) devra nécessairement précéder $stop\,(\triangle\!\!\!\triangle_{i_2})$ pour que la prémisse $\Phi_{Real}(\delta_i, \mathcal{G}_I^t) = \Phi(\delta_i, \mathcal{G}_I^t)$ apparaisse.

$step\,(\triangle\!\!\!\triangle_{i_3}, k_3, x_{j3})$ est *nécessairement postérieure* à $step\,(\triangle\!\!\!\triangle_{i_1}, k_1, x_{j1})$ si et seulement si il existe un syntagme non réel σ_3 dans $\Psi(R_{i_3}^{k-1}(x_{j3}), \mathcal{G}_I^{t_1}) \cap Y_{i_1}^{k_1}$; en effet $\triangle\!\!\!\triangle_{i_1}$ EXIT (σ_3) devra nécessairement précéder $step\,(\triangle\!\!\!\triangle_{i_3}, k_3, x_{j3})$ pour que la prémisse $\Psi_{Real}(R_{i_3}^{k_3}(x_{j3}), \mathcal{G}_I^t) = \Psi(R_{i_3}^{k_3}(x_{j3}), \mathcal{G}_I^t)$ apparaisse.

Examinons maintenant le fonctionnement des réducteurs :

dans le cas d'une transition $start\,(\triangle\!\!\!\triangle_i, 0, x_j)$, la définition des rangs impose que x_j ait été produit par une transition de rang inférieur (peut importe que d'autres transitions le produisent ultérieurement)

dans le cas d'une transition $step\,(\triangle\!\!\!\triangle_i, k, x_j); k > 0$, dont la prémisse graphique est $\Psi_{Real}(R_i^k(x_j), \mathcal{G}_I^t) = \Psi(R_i^k(x_j), \mathcal{G}_I^t)$; prenons une assertion réelle quelconque α dans $\Psi_{Real}(R_i^k(x_j), \mathcal{G}_I^t)$ et supposons que toutes les transitions productrices de α soient de rang ¿= "p", alors il y avait à l'issue du rang "$p-1$" un syntagme non réel α_0 qui a produit α par réduction, et qui empêchait la réalisation de la prémisse ; le rang de $step\,(\triangle\!\!\!\triangle_i, k, x_j); k > 0$ est donc ¿ "p".

Théorème 2 :

Soit un calcul $\{\triangle\!\!\!\triangle_I\}_{i \in I}$ qui s'arrête, l'état final $\mathcal{G}_I^\infty$ est fonction du seul état initial $\mathcal{G}_I^0$.

démonstration de "Théorème 2" :

Supposons qu'un même calcul $\{\mathbb{A}_I\}_{i\in I}$ puisse produire deux journaux finis différents, l'un contenant les $\mathcal{G}_I^{t_{a_i}\to t_{a_j}}$, l'autre contenant les $\mathcal{G}_I^{t_{b_i}\to t_{b_j}}$. Et appliquons à chaque journal le classement des transitions par rang ; nous allons montrer par récurrence sur "n" : [les transitions de rang "$<= n$" sont les mêmes dans les deux journaux.]

La propriété est évidente au rang "0".

Supposons la propriété vraie au rang "n" et soit une transition $\mathcal{G}_I^{t_{a_i}\to t_{a_j}}$ de rang "$n+1$", appartenant au premier journal.

S'il s'agit d'une $start\ (\mathbb{A}_i, 0, x_j)$, x_j a été produit par une transition de rang "n" dans les deux journaux, et donc $start\ (\mathbb{A}_i, 0, x_j)$ est également de rang "$n+1$" dans le second journal.

S'il s'agit d'une $stop\ (\mathbb{A}_i)$, la prémisse (indépendante du journal) a été réalisée lorsque toutes les transitions de rang "n" ont eu lieu, tout en étant nécessairement postérieure à l'une d'entre elles ; et ceci est vrai dans les deux journaux.

S'il s'agit d'une $step\ (\mathbb{A}_i, j, x_k)$, la prémisse est issue d'une transition de rang inférieur ; elle est donc identique dans les deux journaux. Par ailleurs le réducteur s'applique à des syntagmes issus de transitions de rang "$<= n$" qui sont les mêmes dans les deux journaux.