Modélisation algébrique du dîner des philosophes

Anne Dicky, David Janin

LaBRI, CNRS UMR 5800
Laboratoire Bordelais de Recherche en Informatique

Rapport de recherche RR-1474-13

Modélisation algébrique du dîner des philosophes

September 2, 2013

Anne Dicky, David Janin,
LaBRI, IPB, Université de Bordeaux

Contents

Modélisation algébrique du dîner des philosophes

Anne Dicky, David Janin
Université de Bordeaux,
LaBRI, UMR 5800,
351, cours de la Libération,
F-33405 Talence, FRANCE

September 2, 2013

Abstract

Dans cet article, nous présentons une étude expérimentale de modélisation de systèmes distribués à l'aide des outils de la théorie des monoïdes inversifs. Pour ce faire, nous nous plaçons dans les monoïdes (inversifs) des bi-arbres étiquetés. Ces monoïdes dérivent de la présentation des monoïdes inversifs libres de Scheiblich et Munn. Ils peuvent aussi être vus comme un cas particulier des monoïdes de tuilage de Kellendonk et Lawson. Ces outils sont alors mis en œuvre pour modéliser le problème du dîner des philosophes de Dijkstra.

1 Introduction

Les méthodes formelles sont des techniques, mathématiquement bien fondées, dédiées à la spécification, au développement, et à la vérification des systèmes informatisés. Elles visent à augmenter la fiabilité et la robustesse de ces systèmes. A l'intersection des mathématiques et de l'informatique, elles doivent permettre de démontrer formellement la correction des systèmes auxquels elles s'appliquent.

En s'appuyant sur des domaines de l'informatique théorique aussi variés que la logique, la théorie des types, la théorie des automates et la théorie des langages, l'usage effectif des méthodes formelles repose cependant sur l'intégration, dans ces méthodes, de métaphores et de paradigmes de modélisation adaptés à l'ingénierie des systèmes. En effet, si l'on peut souhaiter que les ingénieurs aient une connaissance générales de ces domaines fondamentaux, il semble parfaitement utopique de croire qu'ils pourraient en être des experts [22].

La réalisation de systèmes réactifs fiables reste cependant une affaire délicate. Suivre une méthode formelle revient à se frayer un chemin parfois difficile entre une rigueur mathématique très terre à terre et des intuitions conceptuelles parfois trop métaphoriques (ou trop ad hoc) donc difficiles à analyser et à formaliser.

Le développement de modèles et de méthodes offrant encore une meilleure intégration des intentions créatrices des ingénieurs tout en préservant la rigueur mathématique nécessaire à leurs validations reste d'actualité; les bénéfices attendus croissent d'autant que la demande en fiabilité augmente [22].

Mais il ne suffit pas de formaliser une pratique d'ingénierie courante pour réussir une telle entreprise. En effet, rien ne permet de supposer, a priori, que le formalisme résultant conduira à une théorie mathématique solide *et* effective. L'indécidabilité de la plupart des problèmes d'analyse et de synthèse de systèmes dès lors que ces systèmes se composent d'au moins deux processus séquentiels communicants [20] est, à ce titre, significative.

Dans cette article, nous proposons au contraire d'étudier l'*applicabilité* d'une théorie mathématique *existante* : la théorie des semi-groupes inversifs [18].

Plus précisément, nous proposons d'*expérimenter* le modèle des *bi-arbres étiquetés* à la modélisation des systèmes réactifs. Découvert dans les années 70 par Scheiblich et Munn [21, 19], le modèle des bi-arbres offre une représentation effective des éléments du monoïde inversif libre $FIM(A)$ engendré par un alphabet A. Il hérite donc des nombreuses propriétés algébriques de ces monoïdes. En étiquetant les sommets des bi-arbres, on obtient un cas particulier des monoïdes de tuilages utilisés en physique mathématique pour l'étude des quasi-cristaux [16, 17].

Notons que les monoïdes inversifs apparaissent déjà au cours des décennies précédentes dans le contexte de la logique linéaire [9] ou de l'analyse de la réversibilité des calculs en programmation fonctionnelle [1]. Néanmoins, l'étude faite ici s'apparente bien plus à l'approche de modélisation hiérarchique des systèmes informatisés par graphes d'états et de transitions, telle qu'elle apparaît dans les *Statecharts* [10] ou, plus récemment, dans la modélisation par raffinement successifs en *Event-B* [5, 2].

2 Principes généraux de modélisation

Nous présentons ici le modèle des bi-arbres, les concepts sous-jacents, et quelques *principes de modélisation* qui en découle. Le problème du dîner des philosophes, modélisé dans la section suivante, constitue alors une mise en œuvre de ces principes.

2.1 Observations spatio-temporelles partielles

En toute généralité, le comportement d'un système informatisé peut être vu comme une certaine trace d'exécution dans un espace spatio-temporel complexe constitué, en particulier, des données distribuées qu'il manipule, et de leur évolution et/ou leur migration au cours du temps.

Dans les modèles classiques couramment utilisés aujourd'hui, notamment lors d'une modélisation par graphe d'*états et transitions* [3], les dimensions spatiales de ces traces sont structurellement distinguées de leurs dimensions temporelles. Les dimensions spatiales apparaissent dans la structure des états

du système. Les dimensions temporelles sont modélisées par des arcs étiquetés: les transitions.

Cependant, d'un point de vue strictement mathématique, rien n'oblige à faire cette distinction. De plus, dans la démarche de modélisation, une telle distinction peut paraître arbitraire.

Par exemple, une structure `x` en langage C de type {`int a`; `int b`} peut apparaître comme la description d'une paire d'attributs de type entier. Dans ce cas, les noms des attributs `a` et `b` participent à la description *spatiale/statique* de la structure `x`. Cependant, dès lors qu'on manipule l'entier `x.a`, on décrit implicitement l'action, positionnée dans le temps, qui nous permet d'accéder à la valeur de l'attribut `a` dans la structure `x`. Dans ce cas, les mêmes noms d'attributs `a` et `b` participent tout aussi bien à la description *temporelle/dynamique* de la structure `x`. Dans les langages objets, on confondra même le nom de l'attribut avec la fonction d'accès associée.

Dans cet article, nous proposons de représenter de façon uniforme ces deux aspects spatial et temporel du comportement des systèmes, en modélisant tout ou partie du comportement d'un système comme une *observation partielle* de sa trace spatio-temporelle.

Pour cela, nous explicitons d'une part la partie observée de cette trace et, d'autre part, le point d'entrée et le point de sortie de cette observation. Le modèle résultant est le modèle des bi-arbres étiquetés. La composition de deux observations successives induit une structure algébrique particulièrement riche : le monoïde inversif des bi-arbres étiquetés.

La notion d'état d'un système, c'est-à-dire la description (partielle) des propriétés d'un point dans l'espace-temps de son exécution, se traduit par les observations pour lesquelles point d'entrée et point de sortie coïncident. Ce sont les bi-arbres idempotents.

Dans le cas de la structure `x` évoquée ci-dessus, on peut distinguer sur un même graphe arborescent, l'*état* de la structure `x`, représenté à gauche dans la Figure 1, de l'*accès* à l'un de ces champs, représenté à droite dans la même figure. Les arcs pendants, marqués *in* et *out*, qui indiquent les points d'entrée et de

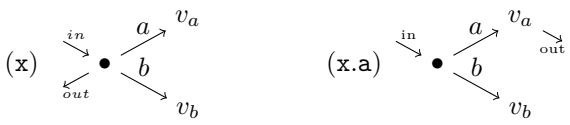

Figure 1: Représentation de la structure `x` et de l'accès à son attribut `x.a`

sortie des observations modélisées, nous permettent de faire cette distinction. Les étiquettes de sommets v_a et v_b nous permettent de modéliser les valeurs des attributs `a` et `b` de la structure `x`.

2.2 Les bi-arbres étiquetés

Soit A un alphabet (fini) d'étiquettes d'arcs et F un alphabet d'étiquettes de sommets.

Définition 1 (Bi-arbre) *Un bi-arbre B est un graphe orienté de support arborescent dont les arcs sont étiquetés sur l'alphabet A, les sommets sur l'alphabet F, bi-déterministe, avec deux sommets distingués: la racine d'entrée et la racine de sortie.*

L'ensemble des sommets d'un bi-arbre B est noté $dom(B)$ et la racine d'entrée (resp. la racine de sortie) est notée $in(B) \in dom(B)$ (resp. $out(B) \in dom(B)$). Pour tout $a \in A$ et tout $x, y \in dom(B)$, on note $x \xrightarrow{a} y$ l'existence d'un arc étiqueté par a allant du sommet x au sommet y.

La condition de bi-déterminisme est formellement définie de la façon suivante. Pour tout $a \in A$, pour tout $x, y, z \in dom(B)$:

- ▷ *déterminisme avant:*ăsi $x \xrightarrow{a} y$ et $x \xrightarrow{a} z$ alors $y = z$,
- ▷ *déterminisme arrière:* si $x \xrightarrow{a} y$ et $z \xrightarrow{a} y$ alors $x = z$.

Deux exemples de bi-arbres sont décrits Figure 2 ci-dessous, les sommets d'entrée (resp. de sortie) étant marqués par des flèches entrantes (resp. sortantes).

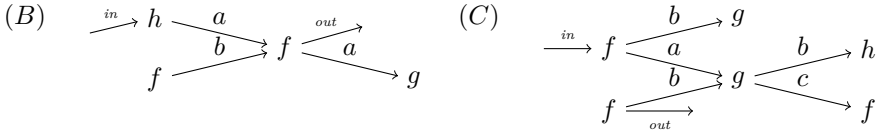

Figure 2: Deux bi-arbres sur les alphabets $F = \{f, g, h\}$ et $A = \{a, b, c\}$

Pour la modélisation de systèmes, les bi-arbres seront donc utilisés de la façon suivante:

Chaque bi-arbre est une observation de tout ou partie de l'espace *spatio-temporel* dans lequel le système modélisé évolue. La racine d'entrée marque le début de cette observation, la racine de sortie en marque la fin.

Ainsi, un arc dans un bi-arbre pourra décrire l'effet d'une action élémentaire exécutée dans le temps. Un arc pourra aussi bien modéliser la structuration spatiale des données de cet espace.

2.3 Composition d'observations et produit de bi-arbres

L'ensemble des bi-arbres, étendu par un élément supplémentaire noté 0, le *bi-arbre indéfini*, est muni d'une opération de produit.

Définition 2 (Produit de bi-arbres) *Le produit $B \cdot C$ de deux bi-arbres B et C, lorsqu'il est défini, s'obtient par* synchronisation *de la racine de sortie de B avec la racine d'entrée de C, puis* fusion *par bi-déterminisation des graphes ainsi positionnés. Les étiquettes des sommets fusionnés doivent être les mêmes. Dans le cas contraire, on pose $B \cdot C = 0$, le produit étant étendu par $B \cdot 0 = 0 \cdot B = 0$ pour tout bi-arbre B.*

Le produit $B \cdot C$ des bi-arbres B et C est représenté Figure 3. L'ovale

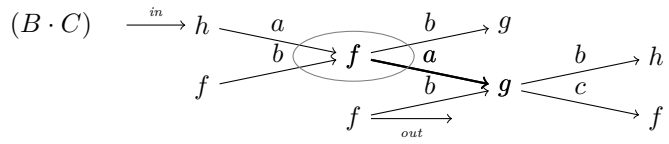

Figure 3: Le produit $B \cdot C$ des bi-arbres B et C.

marque le sommet de synchronisation résultant de la fusion de la sortie de B avec l'entrée de C. L'arc étiqueté $f \xrightarrow{a} g$ marqué en gras est l'arc commun aux bi-arbres B et C ainsi synchronisés.

On note $\mathcal{B}(F, A)$ l'ensemble des bi-arbres étiquetés sur les alphabets F et A.

Lemme 3 (Semi-Groupe et monoïde des bi-arbres) *La structure algébrique obtenue en munissant $\mathcal{B}(F, A)$ du produit de bi-arbres est un semi-groupe, i.e. le produit de bi-arbres étiquetés est associatif.*
Si F est un singleton, le semi-groupe $\mathcal{B}(F, A)$ admet un élément neutre : l'arbre réduit à un seul sommet. Sinon, on étend le semi-groupe par un élément neutre 1*.*

On note toujours $\mathcal{B}(P, A)$ le monoïde ainsi obtenu. Un bi-arbre défini et, dans le cas où F n'est pas un singleton, distinct de 1, est dit *non trivial*.

La composition d'observations partielles cohérentes apparaît comme une superposition suivie d'une fusion. Cette composition induit un produit associatif, qui vaut zéro lorsque les observations sont incompatibles.

2.4 États et idempotents

Comme souvent en théorie des semi-groupes, les idempotents jouent un rôle prépondérant. Rappelons qu'un élément $B \in \mathcal{B}(F, A)$ est idempotent lorsque $B \cdot B = B$. Le bi-arbre indéfini 0 est idempotent.

Lemme 4 (Idempotents) *Un bi-arbre non trivial $B \in \mathcal{B}(F, A)$ est* idempotent *si et seulement si ses sommets d'entrée et de sortie coïncident. De plus, l'ensemble des idempotents forme un sous-monoïde commutatif de $\mathcal{B}(F, A)$.*

Les états des systèmes correspondant à des observations où points d'entrée et de sortie coïncident, il vient:

Les états d'un système sont modélisés par les bi-arbres non triviaux idempotents.

Lemme 5 (Ordre naturel sur les idempotents) *La relation $\leq$ définie sur les bi-arbres idempotents par $B \leq B'$ lorsque $B \cdot B' = B$ est une relation d'ordre. L'ensemble des idempotents ainsi ordonné est un $\wedge$-semi-treillis, le produit d'idempotents coïncide avec l'opérateur $\wedge$.*

L'ordre naturel sur les modèles d'états correspond à une relation de raffinement par accumulation. Le produit de deux états correspond à leur superposition "parallèle". Cette superposition peut être incohérente, le résultat du produit est alors le bi-arbre indéfini.

2.5 Transitions, états initiaux et finaux, projections

Lemme 6 (Monoïde inversif) *Pour tout bi-arbre $B \in \mathcal{B}(F, A)$ il existe un unique bi-arbre $B^{-1} \in \mathcal{B}(F, A)$, appelé* inverse *de B, tel que $B \cdot B^{-1} \cdot B = B$ et $B^{-1} \cdot B \cdot B^{-1} = B^{-1}$. Le monoïde $\mathcal{B}(F, A)$ est donc un monoïde inversif.*

Preuve. On vérifie facilement que $0^{-1} = 0$ et $1^{-1} = 1$. L'inverse d'un bi-arbre non trivial B est alors simplement obtenu à partir de B en intervertissant ses sommets d'entrée et de sortie comme illustré Figure 4; □

À tout bi-arbre $B \in \mathcal{B}(F, A)$ on associe sa *projection gauche* défini par $B^L = B^{-1} \cdot B$ et sa *projection droite* défini par $B^R = B \cdot B^{-1}$.

On vérifie que, dans tous les cas, B^L et B^R sont es éléments idempotents. De plus, lorsque B est non trivial, la projection gauche B^L (resp. projection droite B^R) du bi-arbre B s'obtient à partir de B en déplaçant sa racine d'entrée (resp. de sortie) sur sa racine de sortie (resp. d'entrée), comme illustré Figure 4.

On vérifie aussi que lorsque B est idempotent, on a $B^L = B = B^R$ car B est alors son propre inverse. Autrement dit, les fonctions $B \mapsto B^L$ et $B \mapsto B^R$ sont bien des projections surjectives de l'ensemble des bi-arbres quelconques sur l'ensemble des bi-arbres idempotents au sens ou, pour tout bi-arbre B, on a bien $(B^L)^L = B^L$ et $(B^R)^R = B^R$.

Lemme 7 (Ordre naturel) *La relation $\leq$ sur $\mathcal{B}(F, A)$ définie, pur tout B et $C \in \mathcal{B}(F, A)$, par $B \leq C$ lorsque $B = B^R \cdot C$ ou, de façon équivalente, $B = C \cdot B^L$, est une relation d'ordre stable par produit.*

Cet relation d'ordre est illustrée Figure 5. On voit en particulier que, pour des bi-arbres B et B' non triviaux, on a $B \leq B'$ lorsqu'il existe un *plongement* du graphe de B' dans le graphe de B c'est à dire une fonction $f : dom(B') \rightarrow$

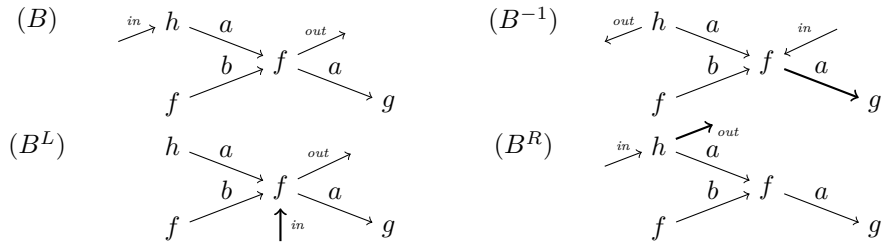

Figure 4: Inverse et projections d'un bi-arbre non trivial

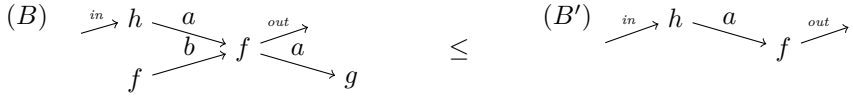

Figure 5: Ordre naturel des bi-arbres étiquetés

$dom(B)$ qui préserve les arcs, les étiquettes des sommets et des arcs et les racines d'entrée et de sortie.

On vérifie aisément que 0 est le plus petit des éléments du monoïde et que l'unité 1 est le plus grand de ses idempotents.

Lemme 8 (Décomposition) *Pour tout bi-arbre non trivial B, l'ensemble $\{B' \in \mathcal{B}(F,A) : B \leq B'\}$ admet un bi-arbre maximum $\hat{B}$. Le bi-arbre $\hat{B}$ est aussi le bi-arbre maximum tel que $B = B^R \cdot \hat{B} \cdot B^L$.*

On vérifie facilement que $\hat{B}$ est un bi-arbre réduit à un chemin de sa racine d'entrée à sa racine de sortie. C'est le *chemin racine* associé à B. Cette notion est illustrée Figure 5 (où $B' = \hat{B}$).

On voit alors apparaître le principe de décomposition suivant:

Tout bi-arbre non trivial B se décompose de façon unique en un état initial minimal B^R (sorte de pré-condition), un état final minimal B^L (sorte de post-condition), et un chemin d'observation maximal $\hat{B}$, avec $B = B^R \cdot \hat{B} \cdot B^L$.

2.6 Comportements partiels et langages de bi-arbres

Une facilité de modélisation supplémentaire est offerte par la possibilité de manipuler non pas des bi-arbres seuls, représentant une approximation d'un comportement du système modélisé, mais plutôt des ensembles de bi-arbres, représentant les approximations possibles d'un ensemble de comportements de ce système.

Une étude récente de ces langages de bi-arbres montre qu'ils héritent largement des bonnes propriétés des langages d'arbres réguliers. Plus précisément,

en s'appuyant sur la notion de langages de bi-arbres définissables en logique monadique du second ordre (MSO), une version logique de la notion de langage régulier, nous avons démontré le résultat suivant:

Théorème 9 *[13] La classe des langages de bi-arbres définissables en MSO est close par union, intersection, produit, produit itéré, inverse et projection gauche ou droite. Cette classe est par ailleurs engendrée par les langages finis et les opérateurs d'union, complément*[1]*, produit et produit itéré.*

Remarque 10 *L'étude mentionnée ci-dessus reste de nature théorique. Elle prouve que la classe des langages définissables en MSO constitue une classe robuste qui jouit de bonnes propriétés. L'étude ultérieure des automates de bi-arbres [12] montre cependant que d'autres classes de langages, plus restreintes mais avec plus de propriétés, pourraient constituer de meilleurs candidats pour la spécification de comportements de systèmes informatisés.*

Remarquons qu'étant donné deux langages de bi-arbres non triviaux L et M, le produit $L \cdot M$ restreint aux bi-arbres définis ne contient que les produits des bi-arbres de L et M qui sont compatibles. Il vient:

Le produit de langages de bi-arbres restreint aux bi-arbres définis agit naturellement comme un filtre pour supprimer les compositions incompatibles. Les critères d'incompatibilité sont modélisés dans les bi-arbres eux-mêmes.

Ce principe est utilisé en particulier Section 3.5 pour modéliser la composition et l'itération des comportements locaux philosophes qui se conforment à l'algorithme distribué de Chandy et Misra [7].

Cette approche de la modélisation de calculs distribués, tout en offrant une approche différente, plus algébrique, n'est pas sans rappeler la modélisation par système de réécritures locales [6].

3 Mise en œuvre sur le dîner des philosophes

Nous proposons ici une modélisation du problème du dîner des philosophes de Dijkstra – et de la solution de Chandy et Misra – à l'aide de langages de bi-arbres étiquetés.

Dans cette approche, tout se passe comme si ce modèle était obtenu par ajout et superposition de "calques" successifs d'ensemble de modèles d'états et de comportements locaux dont le positionnement relatif et la composition résultante sont définis et filtrés par le produit de bi-arbre.

3.1 Le problème

Nous rappelons ici en quelques mots le problème du dîner des philosophes: n philosophes $\varphi_0, \varphi_1, \cdots, \varphi_{n-1}$ sont assis autour d'une table ronde sur laquelle

[1] À la différence de la classe des langages de mots réguliers, le complément est ici nécessaire.

sont disposés n couverts $c_0, c_1, \cdots, c_{n-1}$ (chaque couvert étant placé entre deux philosophes). Pour se restaurer, chaque philosophe a besoin des deux couverts situés à ses côtés. Si par exemple chaque philosophe se saisit de son couvert droit et refuse ensuite de le libérer sans s'être restauré, tous les philosophes mourront de faim. Le problème à résoudre est de proposer un algorithme distribué, c'est-à-dire un programme par philosophe, qui garantit que chaque philosophe affamé finira par manger – en supposant qu'un philosophe qui mange est rassasié au bout d'un temps fini. Ce problème typique de contrôle distribué a été posé par

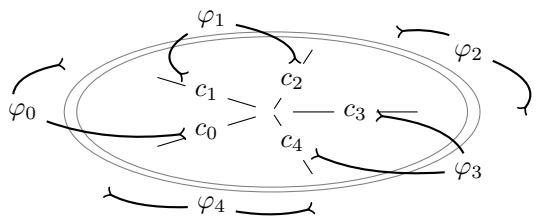

Figure 6: Une configuration globale du dîner des cinq philosophes

Dijkstra [8]. Une solution générique, indépendante du nombre de philosophes, a été proposée par Chandy et Misra [7].

Une configuration globale du dîner avec cinq philosophes est décrite Figure 6. Dans cette figure, les philosophes $\varphi_0, \varphi_1, \cdots, \varphi_4$ sont placés autour de la table dans le sens horaire – ce sera toujours le cas par la suite. Les philosophes φ_1 et φ_3 se restaurent. Les philosophes φ_0, φ_2 et φ_4 parlent et commencent à avoir faim. Le philosophe φ_0 tient un seul couvert, celui à sa droite, mais il lui manque l'autre à sa gauche pour se restaurer. A partir de là, le philosophe φ_1 peut, par exemple, libérer son couvert droit, permettant ainsi au philosophe φ_0 de s'en saisir pour se restaurer.

3.2 Structure locale et globale des états des couverts

Nous commençons notre modélisation par la représentation des états des couverts. On utilise n étiquettes d'arcs $\{c_0, c_1, \cdots, c_{n-1}\}$ pour placer autour d'un sommet racine, sorte d'entrée/sortie unique de chaque observation locale, les sommets étiquetés, pour chaque k, par la situation v_k du k-ième couvert c_k, suivant qu'il est posé sur la table ou pris par l'un des philosophes. On utilise $\bullet \in F$ comme unique étiquète possible de ce sommet racine. La structure du modèle est représentée Figure 7. On constate que $S \leq S_k$ pour tout $0 \leq k \leq n-1$, et, de plus $S = \prod_{0 \leq k \leq n-1} S_k$. Ce constat nous conduit à l'énoncé du principe suivant:

Les sous-états locaux S_k d'un état global S satisfont la relation d'ordre $S \leq S_k$. De plus, tout état global S peut être retrouvé par produit d'un ensemble $\{S_k\}_{k \in I}$ de ses composants locaux: $S = \prod_{k \in I} S_k$.

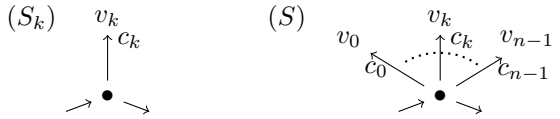

Figure 7: Modélisation d'un état local (S_k) et d'un état global (S) des couverts

3.3 Structure locale des états et transitions des philosophes

Un état local d'un philosophe φ_k $(0 \leq k < n)$ est modélisé par un bi-arbre idempotent P_k tel que décrit Figure 8.

En effet, à chaque instant, le philosophe φ_k n'a connaissance que des couverts c_k et c_{k+1} qui sont à sa portée. Cela nous conduit au principe suivant:

La restriction des modèles à des sous-ensembles d'étiquettes d'arcs permet de rendre compte de la localité de l'information disponible.

La modélisation de l'avancement du système se fait à l'aide d'une étiquette d'arc a représentant l'avancement élémentaire du temps.

Dans la suite, on note aussi a le langage de tous les bi-arbres à un seul arc a dont l'entrée est la source de cet arc, et la sortie la cible de cette arc. On convient par ailleurs, comme pour les langages de mots, d'identifier tout bi-arbre au langage singleton réduit à ce bi-arbre.

Les transitions élémentaires du philosophe φ_k sont modélisées par des bi-arbres T_k de la forme $T_k = P_k \cdot a \cdot P'_k$ comme illustré Figure 8:

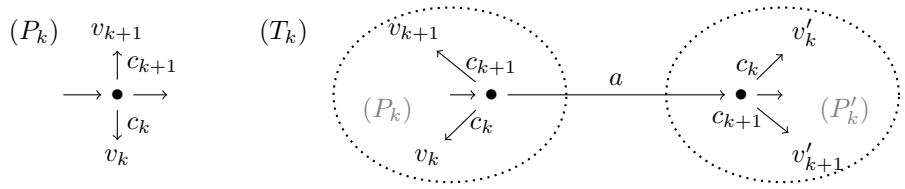

Figure 8: Etat local et transition locale T_k du philosophe φ_k

Nous pouvons remarquer que $P_k \leq T_k^R$, $P'_k \leq T_k^L$ et, de plus, on a $T_k = T_k^R \cdot a = a \cdot T_k^L = T_k^R \cdot a \cdot T_k^L$ comme nous l'indique le Lemme 8. Plus encore, les projections droite et gauche d'un modèle T_k de comportement du philosophe φ_k sont des idempotents; elles peuvent donc être interprétés comme des états. Plus généralement:

La projection droite (resp. gauche) d'un bi-arbre non idempotent apparaît comme l'état initial (resp. final) le plus contraint (minimal dans l'ordre naturel) qu'on puisse associer au comportement modélisé par ce bi-arbre.

3.4 Structure globale des états et transitions des philosophes

On suppose maintenant définies n transitions $\{T_k\}_{k\in[0,n-1]}$ décrivant le comportement local de chaque philosophe au format ci-dessus.

L'état initial global S et l'état final global S' associés à ces comportements sont naturellement décrit par les produits $S = \left(\prod_{0\leq k<n} P_k\right)$ et $S' = \left(\prod_{0\leq k<n} P'_k\right)$ dont la structure est la même que celle de l'état global décrit Figure 7, les états locaux des philosophes étant naturellement agglomérés puisque chaque couvert est partagé par deux philosophes.

Remarque 11 *On voit ici apparaitre le fait que l'état global S ainsi obtenu, tout comme l'état S', peut être null. Cela signifiera que les transitions modélisées sont incompatibles, cette incompatibilité provenant soit de leurs états initiaux $\{P_k\}_k$ soit de leur états finaux $\{P'_k\}_k$. Dans ce cas, le comportement global résultat est lui aussi nul (ou vide si l'on adopte la vision langage).*

Un modèle T de comportement global du système est alors défini par $T = S \cdot a \cdot S'$ dont la structure est décrite Figure 9.

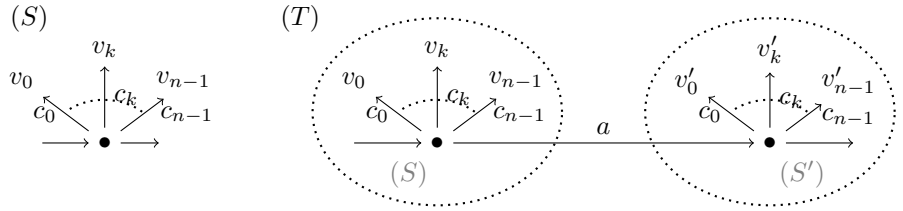

Figure 9: Etat et transition globale des philosophes

On constate qu'on a aussi

$$T = \left(\prod_{0\leq k<n} T_k^R\right) \cdot a = a \cdot \left(\prod_{0\leq k<n} T_k^L\right) = \left(\prod_{0\leq k<n} T_k^R\right) \cdot a \cdot \left(\prod_{0\leq k<n} T_k^L\right)$$

Autrement dit:

Un comportement global T peut être défini uniquement à partir du produit de comportements locaux T_k qui le composent et de leurs projections droites et gauches

3.5 Modélisation de l'algorithme de Chandy et Misra

La description de la structure de notre modélisation du diner des philosophes étant complète, nous nous proposons maintenant de modéliser l'algorithme de Chandy et Misra [7].

Les valeurs possibles des états des couverts sont à choisir dans

$$V = \{0, +, -\} \times \{c, d\} \times \mathcal{P}(\{+, -\})$$

avec $V \subseteq F$, de façon que si $v_k = (x_k, y_k, z_k)$ est la valeur du couvert c_k

- ▷ x_k représente la position du couvert (0: sur la table, +: dans la main du philosophe φ_k, −: dans la main du philosophe φ_{k-1})
- ▷ y_k la qualité du couvert (c: propre, d: sale)
- ▷ z_k les requêtes d'usage: z_k contient + (respectivement: −) si et seulement si le philosophe φ_k (resp. φ_{k-1}) a demandé à l'utiliser.

Ces valeurs caractérisent également les états des philosophes: dire que le philosophe φ_k a faim, c'est dire qu'il a demandé à utiliser les couverts c_k et c_{k+1}; il est en train de manger s'il a faim et a les deux couverts en main.

L'algorithme de Chandy et Misra définit les transitions admissibles pour le philosophe φ_k: c'est le langage L_k de toutes les $T_k = P_k \cdot a \cdot P'_k$ vérifiant les conditions

- ▷ $+ \notin z_k$ et $+ \in z'_k$ si et seulement si $- \notin z_{k+1}$ et $- \in z'_{k+1}$ (si le philosophe a faim, il émet une requête pour les deux couverts);
- ▷ $+ \in z_k$ et $+ \notin z'_k$ si et seulement si $- \in z_{k+1}$ et $- \notin z'_{k+1}$ (le philosophe n'a plus faim: il annule ses requêtes), et dans ce cas $x_k = x'_k = +$, $x_{k+1} = x'_{k+1} = -$, et $y'_k = y'_{k+1} = d$ (le philosophe a fini de manger: les couverts sont sales);
- ▷ si $x_k = +$ et $x'_k \neq +$ (le philosophe libère le couvert c_k), alors $y_k = d$ (le couvert était sale), $- \in z_k$ (le voisin l'avait demandé), $x'_k = -$ et $y'_k = c$ (le philosophe nettoie le couvert et le donne au voisin qui l'a demandé); de même, si $x_{k+1} = -$ et $x'_{k+1} \neq -$ (le philosophe libère le couvert c_{k+1}), alors $y_{k+1} = d$, $+ \in z_{k+1}$, $x'_{k+1} = +$ et $y'_{k+1} = c$ et $+ \notin z'_{k+1}$;
- ▷ si $x_k = x'_k = +$ et $y_k = d$, alors $y'_k = d$ (le philosophe ne nettoie le couvert c_k que s'il le donne à son voisin); de même, si $x_{k+1} = x'_{k+1} = -$ et $y_{k+1} = d$, alors $y'_{k+1} = d$;
- ▷ pour $k > 0$, si $x_k \neq +$ et $x'_k = +$, alors $- \notin z_k$; et pour $k = n - 1$, si $x_0 \neq -$ et $x'_0 = -$, alors $+ \notin z_k$ (un philosophe ne peut prendre un couvert si un voisin de numéro inférieur a demandé ce couvert).

Les transitions globales résultantes constituent le langage L défini par

$$L = \prod_k (L_k)^R \cdot a = a \cdot \prod_k (L_k)^L = \prod_k (L_k)^R \cdot a \cdot \prod_k (L_k)^L$$

En partant de l'état initial S_0 dans lequel tous les couverts sont sur la table, et sales. Les comportements infinis globaux à partir de l'état S sont définis par

$C = S_0 \cdot L^\omega$. Prouver la correction de l'algorithme revient à montrer, le langage C des comportements ainsi obtenu n'est pas vide et que, de plus, pour tout comportement possible, que chaque philosophe mange infiniment souvent.

Formellement, cela reviens à démontrer que, pour tout $0 \leq k < n$, la propriété "le philosophe φ_k se restaure" modélisée par $(x_k = + \wedge x_{k+1} = -)$ est vraie infiniment souvent. Cette dernière étape de vérification, pour laquelle des techniques d'automates de langages de mots infinis sont applicables, sort cependant du cadre de cet article.

4 Conclusion

Nous avons présenté le monoïde des *bi-arbres* à des fins de modélisation de comportements de systèmes informatisés complexes par *accumulation de modèles partiels* dont les positionnements relatifs sont définis par la structure algébrique sous-jacente: le *monoïde inversif* des bi-arbres.

Plusieurs caractéristiques de ces monoïdes inversifs trouvent une interprétation simple en modélisation: la notion d'état est par exemple traduite par la notion de bi-arbre idempotent. Les transitions sont tout à la fois des actions qui permettent de passer d'un état à un autre, mais aussi les bi-arbres obtenus en appliquant ces actions aux états correspondants. Les transitions induisent en retour une notion enrichie d'état (leurs projections droite et gauche).

Notre approche reste cependant exploratoire, limitée à une expérimentation des *outils* et *concepts* offerts par le monoïde des bi-arbres et, au delà, la théorie des monoïdes inversifs. Nous sommes encore bien loin de proposer une *méthode* de modélisation, telle que *Event-B* [2], qui accompagne et documente les outils et concept qu'elle propose.

En particulier, dans la perspective d'une modélisation par raffinements successifs par substitution, il reste à explorer une caractéristique majeure du monoïde des bi-arbres: la possibilité définir un raffinement d'actions avec superpositions partielles. Plus précisément, comme illustré Figure 10, la nature tout à la fois

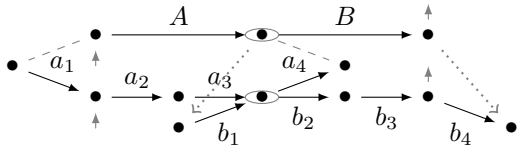

Figure 10: Un exemple de raffinement d'actions avec superpositions partielles

séquentielle et parallèle du produit de bi-arbres permet de remplacer une composition purement séquentielle de la forme $A \cdot B$ à un certain niveau d'abstraction par une composition de la forme $\theta(A) \cdot \theta(A)$ pour θ une fonction de substitution de tout A et B par des bi-arbres. On remarque qu'en faisant cela, des superpositions de comportements concrets peuvent apparaitre.

C'est cette caractéristique qui nous a conduits à définir et à utiliser le modèle des bi-arbres dans le domaine de la modélisation [11, 4, 15] ou la programmation [14]ămusicale. Son utilisation dans un contexte plus général reste à explorer.

References

[1] S. Abramsky. A structural approach to reversible computation. *Theoretical Comp. Science*, 347(3):441–464, 2005.

[2] J.-R. Abrial. *Modeling in Event-B - System and Software Engineering*. Cambridge University Press, 2010.

[3] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants*. Etudes et Recherches en Informatiques. Masson, 1992.

[4] F. Berthaut, D. Janin, and B. Martin. Advanced synchronization of audio or symbolic musical patterns: an algebraic approach. *International Journal of Semantic Computing*, 6(4):409–427, 2012.

[5] D. Cansell and D. Méry. Foundations of the B method. *Computers and Informatics*, 22, 2003.

[6] J. Chalopin and Y. Métivier. A bridge between the asynchronous message passing model and local computations in graphs. In *Mathematical Foundations of Computer Science (MFCS 2005)*, volume 3618 of *Lecture Notes in Computer Science*, pages 212–223. Springer-Verlag, aug 2005.

[7] K.M Chandy and J. Misra. The drinking philosophers problem. *ACM Transactions on Programming Languages and Systems*, 6(4):632–646, 1984.

[8] E. W. Dijkstra. Hierarchical ordering of sequential processes. *Acta Informatica*, 1:115–138, 1971.

[9] J. Goubault-Larrecq. Musings around the geometry of interaction, and coherence. *Theoretical Comp. Science*, 412(20):1998–2014, 2011.

[10] D. Harel. Statecharts in the making: a personal account. In *Third ACM SIGPLAN History of Programming Languages (HOPL), San Diego, California, USA, 9-10 June 2007*, pages 1–43. ACM Press, 2007.

[11] D. Janin. Vers une modélisation combinatoire des structures rythmiques simples de la musique. *Revue Francophone d'Informatique Musicale (RFIM)*, 2, 2012.

[12] D. Janin. Algebras, automata and logic for languages of labeled birooted trees. In *Int. Col. on Aut., Lang. and Programming (ICALP)*, volume 7966 of *LNCS*, pages 318–329. Springer, 2013.

[13] D. Janin. Walking automata in the free inverse monoid. Technical Report RR-1464-12 (revised june 2013), LaBRI, Université de Bordeaux, 2013.

[14] D. Janin, F. Berthaut, M. DeSainte-Catherine, Y. Orlarey, and S. Salvati. The T-calculus : towards a structured programming of (musical) time and space. In *Workshop on Functional Art, Music, Modeling and Design (FARM)*. ACM Press, 2013.

[15] D. Janin, F. Berthaut, and M. DeSainteCatherine. Multi-scale design of interactive music systems : the libTuiles experiment. In *Sound and Music Computing (SMC)*, 2013.

[16] J. Kellendonk. The local structure of tilings and their integer group of coinvariants. *Comm. Math. Phys.*, 187:115–157, 1997.

[17] J. Kellendonk and M. V. Lawson. Tiling semigroups. *Journal of Algebra*, 224(1):140 – 150, 2000.

[18] M. V. Lawson. *Inverse Semigroups : The theory of partial symmetries*. World Scientific, 1998.

[19] W. D. Munn. Free inverse semigroups. *Proceeedings of the London Mathematical Society*, 29(3):385–404, 1974.

[20] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 746–757. IEEE Press, 1990.

[21] H. E. Scheiblich. Free inverse semigroups. *Semigroup Forum*, 4:351–359, 1972.

[22] W. Thomas. Logic for computer science: The engineering challenge. In Reinhard Wilhelm, editor, *Informatics - 10 Years Back, 10 Years Ahead.*, volume 2000 of *LNCS*, pages 257–267. Springer, 2001.