Tâches, types et tactiques pour les systèmes de calculs locaux

Pierre Castéran — Vincent Filou

Laboratoire Bordelais de Recherche en Informatique, LabRI[1]*, UMR 5800,*
Université Bordeaux 1,
351 Cours de la Libération, 33405 Talence Cedex, France
{pierre.casteran, vincent.filou}@labri.fr

RÉSUMÉ. *Nous présentons une formalisation en* Coq *du modèle des* calculs locaux*, un modèle de calcul distribué fondé sur les réétiquetages de graphes. Cette formalisation permet non seulement de prouver la correction d'algorithmes, mais aussi de comparer divers modes de synchronisation et de détection de la terminaison des calculs. Cette comparaison est illustrée par des exemples d'impossibilité de réaliser certaines spécifications, ou des preuves d'inclusion entre modes de détection de la terminaison via des transformations d'algorithmes.*

ABSTRACT. *We present a formalization in* Coq *of* local computations*, a model for distributed computing based on graph relabelings. This formalization allows us to prove distributed algorithms' correctness, as well as to compare various synchronization and termination detection modes. We illustrate our approach by two impossibility proofs and a certified algorithm transformation which helped us to compare two termination detection modes.*

MOTS-CLÉS : *algorithmes distribués, calculs locaux, réétiquetages de graphes, preuves formelles d'algorithmes, assistants à la preuve*

KEYWORDS: *distributed algorithms, local computation systems, graph relabeling, formal proofs of algorithms, proof assistants*

1. Ce travail a bénéficié du soutien de l'ANR : projets **A3PAT** ANR-05-BLAN-0146 et **RIMEL** ANR-06-SETIN-015.

1. Introduction

Les *calculs locaux* sur les graphes, et en particulier les systèmes de réétiquetage de graphes, ont été introduits par Litovsky, Métivier et Sopena [26] comme un outil pour le codage d'algorithmes distribués, leur preuve de correction et l'analyse de leur pouvoir d'expression.

Dans ce modèle, un réseau est représenté par un graphe fini, ses processeurs par les sommets de ce graphe, et les liens de communication par ses arêtes. L'état local d'un processeur (resp. lien) est représenté par une étiquette attachée au sommet (resp. à l'arête) correspondant.

Un système de réétiquetage de graphes est un ensemble de règles dont les membres gauche et droit sont des étiquetages définis sur un même graphe connexe. Chaque règle décrit des modifications locales de l'*état* d'un réseau et non de sa structure.

Nous considérons des systèmes de calculs *locaux*, où chaque règle décrit une transformation de l'étiquetage d'une *étoile*, sous-graphe formé d'un sommet *centre*, d'un ou de plusieurs voisins et des arêtes reliant le centre à ce ou ces voisins.

Dans le cadre du projet Visidia [1, 3, 23, 8], nous nous intéressons à la preuve formelle de propriétés du modèle des calculs locaux. Les propriétés que nous voulons prouver comportent, outre la correction d'algorithmes, l'étude des puissances respectives de primitives de synchronisation, ainsi que des propriétés génériques permettant de prouver, par exemple, l'impossibilité de réaliser certaines spécifications.

Dans cet article, nous présentons le modèle des calculs locaux, puis les choix de représentation des divers aspects de ce modèle en *Coq* [29, 4]. Les preuves complètes en *Coq* des résultats que nous montrons sont disponibles sur la page du projet : http://www.labri.fr/∼casteran/Loco.

Les notions de tâche et de mode de détection de la terminaison sont issues des travaux de Godard *et al.* [20, 11, 12, 28]. Formali-

ser ces notions en théorie des types a motivé des changements de définition respectant scrupuleusement l'esprit de ces travaux.

Relation avec d'autres travaux :

Le travail de Gascard et Pierre [19] s'intéresse aux réseaux symétriques d'interconnections : anneaux, tores, hypercubes, pour l'exécution de programmes parallèles. Ce travail, bien qu'éloigné du modèle de calculs locaux que nous utilisons, présente un grand intérêt méthodologique, principalement par son approche compositionnelle de composants certifiés, son utilisation de la géométrie du réseau et son utilisation de l'assistant à la preuve Nqthm [6].

Ching-Tsun Chou [14] utilise l'assistant *HOL* [22] pour prouver la correction d'algorithmes distribués. Le modèle utilisé est celui des systèmes de transition étiquetés. Les spécifications se font en logique temporelle et les preuves de correction s'appuient sur des preuves de simulation à l'aide d'invariants de liaison.

Cansell et Méry [7] utilisent des techniques de construction de programme par raffinements successifs pour obtenir, à partir d'une spécification, un système de transitions sous forme de machine abstraite en `B` événementiel. Chaque raffinement est validé avec l'aide de Rodin [16], ce qui assure la correction du système obtenu vis-à-vis de la spécification. La méthodologie utilisée par les auteurs assure que ce système est conforme au modèle des calculs locaux. Notre travail est complémentaire du leur, en ce sens qu'il s'intéresse aux preuves de propriétés de diverses classes de systèmes de calcul locaux, et à l'applicabilité de ces propriétés aux cas concrets de spécifications, plus qu'à la dérivation d'algorithmes à partir des spécifications. Néanmoins, les quelques preuves de correction d'algorithmes que nous avons menées utilisent les mêmes invariants que l'article cité.

2. Le modèle des calculs locaux

Nous présentons brièvement les principaux composants du modèle des calculs locaux : les structures de données employées et plusieurs variations sur les notions de spécification et de réalisation.

Cette présentation diffère parfois des travaux théoriques dont nous nous sommes inspirés, car nous avons dû l'adapter aux exigences de la preuve sous *Coq* : formalisation en théorie des types, ergonomie de la construction interactive de preuves.

2.1. *Les domaines de calcul*

Nous considérons uniquement dans cet article des graphes finis, non orientés, simples et connexes. L'ensemble des sommets d'un graphe G sera noté V_G et l'ensemble de ses arêtes E_G.

Les valeurs sur lesquelles opèrent les calculs sont réparties sur les sommets et les arêtes des graphes. Plus précisément, si L est un type, un *étiquetage* (ou *état*) de G est une fonction totale σ associant à tout sommet ou arête de G un élément de L [1]. Nous noterons $\Sigma_{G,L}$ le type des étiquetages sur G ainsi définis. Nous emploierons fréquemment la notation (G, σ) pour désigner un graphe étiqueté ; G sera appelé *le graphe sous-jacent de* σ.

Si G_1 est un sous-graphe de G et σ un étiquetage sur G, la restriction de σ à G_1 sera notée $\sigma_{|G_1}$. Nous dirons que (G_1, σ_1) est un *sous-graphe étiqueté* de (G, σ) (ou que (G, σ) *étend* (G_1, σ_1)) si G_1 est un sous-graphe de G et σ_1 est la restriction de σ_1 à G_1.

Soient deux types L et L' et $f : L \rightarrow L'$; nous noterons $\hat{f}$ la fonction transformant tout graphe G étiqueté sur L en un graphe étiqueté sur L' par application de f à toutes les étiquettes de G.

Soient deux graphes étiquetés (G, σ) et (G', σ'). Un *morphisme de graphes étiquetés* est une fonction $\phi : V_G \rightarrow V_{G'}$ telle que si $\{x, y\}$ est une arête de G, alors $\{\phi(x), \phi(y)\}$ est une arête de G', et ϕ commute avec σ et σ'.

Nous nous intéressons aux algorithmes effectuant des *réétiquetages*, c'est-à-dire à des transformations de graphes étiquetés *ne modifiant pas la structure du graphe sous-jacent.* L'extension de notre

1. Dans la formalisation en *Coq*, nous utilisons deux types, l'un pour étiqueter les sommets, l'autre pour les arêtes, et donc un couple de fonctions d'étiquetage, mais nous avons préféré simplifier les notations de cette présentation.

travail aux réseaux dynamiques reste cependant un des objectifs de notre projet.

2.2. *Notion de tâche*

Une *tâche* est une spécification d'une transformation de graphes étiquetés préservant le graphe sous-jacent. Si L_i (resp. L_o) est le type servant à étiqueter les états d'entrée (resp. de sortie), une tâche est un prédicat de type T de type $\forall G$, $\Sigma_{G,L_i} \rightarrow \Sigma_{G,L_o} \rightarrow \texttt{Prop}$. Le domaine de T est donc la famille de graphes étiquetés sur laquelle la transformation doit être définie. Définir une tâche revient alors à préciser sur quel ensemble de graphes étiquetés elle doit opérer et quelle relation lie états d'entrée et de sortie. Considérons deux exemples de tâches analysés dans cet article :

Tâche 1 (Calcul du degré de chaque sommet d'un graphe) *Il s'agit, à partir d'un graphe non étiqueté G, d'obtenir un état dans lequel chaque sommet est étiqueté par son degré dans G. Un graphe non étiqueté peut se représenter à l'aide d'un étiquetage uniforme où tous les sommets et arêtes sont étiquetés par l'unique valeur* `tt` *du type singleton* `unit`*. Nous pouvons alors définir la tâche qui associe à tout graphe G ainsi étiqueté un graphe où les sommets sont étiquetés par des entiers naturels et les arêtes à l'aide du type* `unit`*, et tel que chaque sommet v de G est étiqueté par son degré dans G.*

Tâche 2 (Élection dans un arbre) *Nous considérons une tâche d'élection dont le domaine est l'ensemble des arbres G (graphes connexes et acycliques) dont chaque sommet est initialement étiqueté par son degré, les arêtes étant non étiquetées. À tout élément de ce domaine, nous associons un étiquetage où chaque sommet est étiqueté par une des deux constantes* `élu` *ou* `battu`*, et tel qu'un unique sommet de G est étiqueté* `élu`*.* Traditionnellement, la spécification des algorithmes d'élection précise qu'un sommet battu (resp. élu) reste définitivement dans cet état. Cet aspect concerne le déroulement de l'algorithme et non le résultat final, et sera traité

en section 3.2. *Nous évoquerons également une variante de cette tâche où le graphe de départ est non étiqueté.*

2.3. *Relations et systèmes de réétiquetage, systèmes localement engendrés*

Les tâches définies précédemment se réalisent par des transformations successives de l'étiquetage d'un graphe G, en utilisant un type d'étiquetage L (pouvant être différent de L_i et L_o, du fait de la présence éventuelle de données auxiliaires utiles aux calculs).

Une *relation de réétiquetage* sur G est une relation binaire R définie sur les étiquetages de G par L. De façon classique, nous notons R^* la fermeture réflexive et transitive de R. Nous appellerons *état* toute étape d'un calcul $(G,\sigma) \xrightarrow{R^*} (G,\sigma')$. Un *système de réétiquetage* $\mathcal{S}$ est une fonction associant à tout graphe G une relation de réétiquetage $\mathcal{S}_G$ sur G.

Pour tout sommet c de G, nous notons $E(c)$ ("étoile de centre c") le sous-graphe de G comportant le sommet c, ses voisins immédiats et les arêtes reliant c à ces voisins. Nous considérons uniquement des systèmes de réétiquetage $\mathcal{S}_G$ possédant les caractéristiques suivantes :

> Pour tout graphe G, si $(G,\sigma) \overset{\mathcal{S}_G}{\to} (G,\sigma')$, il existe un sommet c de G tel que :
>
> – Seules les étiquettes de $E(c)$ sont modifiées,
>
> – Si $\sigma_{|E(c)} = \sigma_{1|E(c)}$, alors il existe σ'_1 tel que $\sigma'_{|E(c)} = \sigma'_{1|E(c)}$ et $(G,\sigma_1) \xrightarrow{\mathcal{S}_G} (G,\sigma'_1)$.
>
> Si ϕ est un isomorphisme de graphes étiquetés et si $(G,\sigma) \xrightarrow{\mathcal{S}_G} (G,\sigma')$, alors $\phi(G,\sigma) \xrightarrow{\mathcal{S}_{\phi(G)}} \phi(G,\sigma')$.

On dit alors que le système est *localement engendré*. De façon intuitive, l'application d'un pas de réétiquetage autour d'un sommet c est entièrement déterminée par l'étiquetage courant au voisinage immédiat de c.

2.4. *Types de synchronisation*

Les travaux précédemment cités [2, 25] considèrent trois types de règles permettant de construire des systèmes localement engendrés. Ils se distinguent par les conditions d'application d'une règle (sa *garde*) et les modifications d'étiquetage autorisées.

2.4.1. *Synchronisation **LC0** (Rendez-vous)*

Une règle **LC0** opère sur deux sommets adjacents du graphe considéré. Cela correspond à un *rendez-vous* [2, 25, 26] entre ces deux sommets. Ce type de calcul correspond aussi au modèle client-serveur de nombreuses applications réseau. Une règle **LC0** s'écrit sous la forme d'un prédicat associé à la transformation d'état de deux sommets voisins c et v et de l'arête les reliant. Ce prédicat relie les anciennes étiquettes de c, v et $\{c, v\}$ aux nouvelles étiquettes. Il suffit donc de définir un prédicat à six arguments sur le type L. Nous dirons qu'un système de réétiquetage est **LC0** s'il est décrit par un ensemble de règles **LC0**. Nous donnons deux exemples de telles règles :

Règle 1 (Élection dans un arbre) *Les sommets sont étiquetés sur le type* `option nat` [2] *Intuitivement, une étiquette* `Some` *i signifie que le sommet qui la porte peut encore espérer être élu, i étant le nombre de ses voisins pouvant aussi encore être élus. Un sommet porte l'étiquette* `None` *s'il est battu. Si deux sommets adjacents c et v sont respectivement étiquetés par* `Some (S k)` *et* `Some 1`*, l'étiquetage devient respectivement* `Some` *k et* `None` *(voir figure 1).*

Règle 2 (Calcul du degré des sommets d'un graphe) *Les sommets sont étiquetés à l'aide du type* `nat` *et les arêtes à l'aide du type bool. Si deux sommets adjacents c et v sont respectivement étiquetés par i et j et l'arête qui les relie par* ***false****, alors les étiquettes respectives de c et de v deviennent* `S` *i et* `S` *j ; celle de l'arête joignant c à v devient* ***true*** *(voir figure 2).*

---

2. Rappelons que `None` et `Some` sont les constructeurs du type `option nat`, et que `S` n désigne le successeur de n.

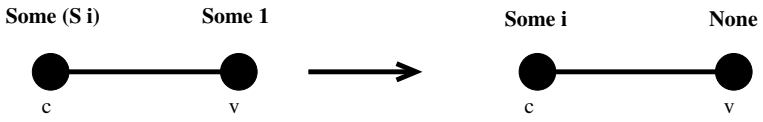

Figure 1 – Élection dans un arbre : règle **LC0**.

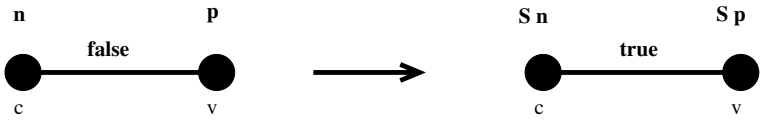

Figure 2 – Calcul du degré : règle **LC0**.

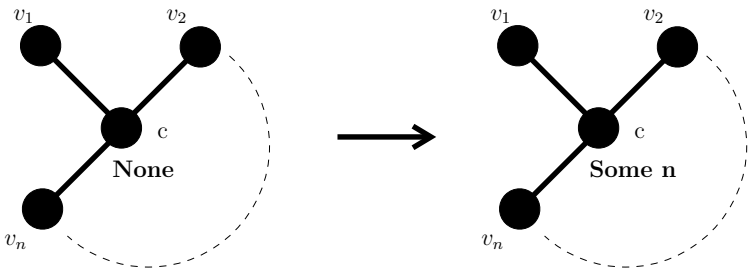

Figure 3 – Calcul du degré : règle **LC1**.

2.4.2. *Synchronisations **LC1** et **LC2***

Dans la synchronisation **LC1**, la garde d'une règle porte sur tout l'étiquetage d'une étoile de centre c. En revanche, seules les étiquettes du centre c et des arêtes reliant c à ses voisins peuvent être modifiées. Dans la synchronisation **LC2**, on autorise la modification des étiquettes de toute l'étoile. À titre d'exemple, la règle **LC1** ci-dessous permet de calculer le degré de chaque sommet d'un graphe :

Règle 3 (Calcul du degré des sommets d'un graphe, LC1) *Les sommets sont étiquetés par le type* `option nat`*. Si le centre c d'une étoile est étiqueté par* `None`*, on remplace cette étiquette par* `Some` *n, où n est le degré de c dans G (voir figure 3).*

Bauderon *et al.* [2] donnent une suite d'exemples d'algorithmes distribués utilisant des règles **LC1** ou **LC2** provoquant des modifications des étiquettes d'arêtes, ou des systèmes **LC2**. L'algorithme d'énumération de Mazurkiewicz [27] est un système **LC2** complexe où les sommets sont étiquetés par des ensembles finis d'ensembles finis d'entiers naturels.

2.5. *Réalisation d'une tâche*

Soit une tâche T, définie à l'aide des types d'étiquetage L_i et L_o. Nous nous proposons de définir à quelle condition un système de réétiquetage S *réalise* T. À titre d'exemple, la figure 4 montre les étapes de l'élection d'un sommet dans un graphe par le système décrit par la règle 1 [3].

Une première phase d'initialisation permet de transformer un étiquetage sur L_i en un état initial pour le système de réétiquetage S. Nous faisons l'hypothèse que cette initialisation se fait uniformément en appliquant une fonction de L_i dans L. De façon symétrique, tout état final pour le système $\mathcal{S}$ est projeté de façon uniforme vers un étiquetage sur le type de sortie L_o.

Plus formellement, une réalisation de T est une structure comportant les champs suivants :

– Un type d'étiquetage L,

– Un système de réétiquetage $\mathcal{S}$ sur L,

– Une injection $\iota : L_i \rightarrow L$. Cette fonction, appliquée aux sommets et arêtes de G, permet d'associer à tout graphe étiqueté $(G, \sigma_i) \in D$ un *état initial* noté $\hat{\iota}(\sigma_i)$.

– Une projection $\pi : L \rightarrow L_o$. À l'inverse de ι, cette fonction permet d'extraire d'un étiquetage σ sur L un étiquetage $\hat{\pi}(\sigma)$ sur L_o. Intuitivement, lorsque σ est l'état final d'un calcul, $\pi(\sigma(v))$ repré-

3. Notons que nous n'avons représenté qu'un calcul parmi d'autres possibilités et qu'une autre suite de réétiquetages aurait mené à l'élection d'un autre sommet. En effet tout choix d'une réécriture parmi plusieurs réécritures possibles donne lieu à un calcul admis.

sente la part locale de l'information calculée portée par le sommet v.

– Une preuve de terminaison de tout calcul suivant $\mathcal{S}_G$ issu d'un état initial,

– Une preuve que pour tout graphe étiqueté (G, σ_i) du domaine de T et pour tout calcul $(G, \hat{\iota}(\sigma_i)) \xrightarrow{S_G^*} (G, \sigma_f)$ où σ_f est irréductible (c-à-d. en forme normale pour la relation $\mathcal{S}_G$), (G, σ_i) et $\hat{\pi}(\sigma_o)$ sont en relation par la tâche T.

Nous dirons simplement qu'un état σ est *accessible* [4] s'il est accessible à partir d'un état initial.

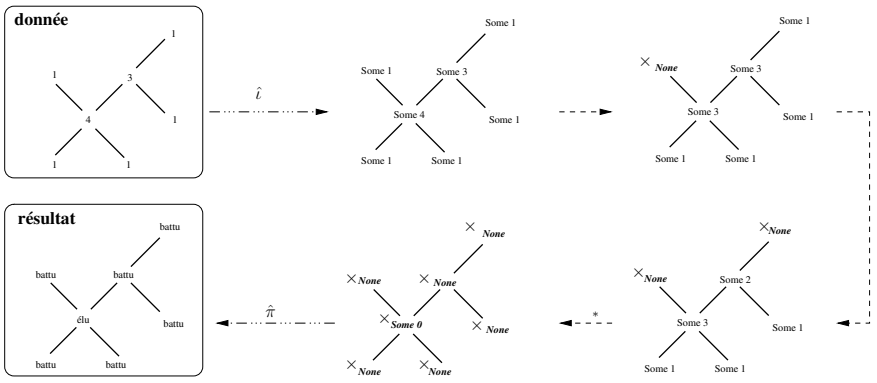

Figure 4 – Étapes d'une élection : les sommets marqués d'un × sont dans un état (local) terminal.

2.5.0.1. Remarque :

Dans l'article de Godard, Métivier et Tel [20], les fonctionnalités d'initialisation et d'extraction du résultat du calcul sont assurées *via* des tuples de valeurs redondants. Notre formalisation à l'aide des fonctions ι et π permet de supprimer cette redondance et d'éviter d'avoir à prouver qu'elle ne risque pas d'introduire d'incohérence dans la représentation des états. La même remarque s'applique aux oracles décrits en section 3.

4. À ne pas confondre avec le prédicat `Acc` de la bibliothèque *Coq* sur les relations bien fondées.

Reprenons l'exemple de la figure 4. L'injection ι convertit toute étiquette d'entrée i en `Some` i, afin de construire un état initial pour le système de réétiquetage. La projection π est définie ci-dessous :

$$\begin{aligned} \pi(\texttt{Some 0}) &= \texttt{élu} \\ \pi(\texttt{None}) &= \texttt{battu} \\ \pi(\texttt{Some (S } n\texttt{)}) & \quad \text{non spécifiée} \end{aligned}$$

La valeur non spécifiée peut être arbitrairement choisie entre `élu` ou `battu`. Nous verrons en effet, que le troisième cas de la définition de π est associé aux sommets « encore en ballottage » et par conséquent n'apparaît jamais dans les états irréductibles. Les techniques utilisées pour montrer que nous avons bien construit une réalisation de la tâche 2 sont montrées en section 4.1.

3. Modes de détection de la terminaison

Notre définition de réalisation d'une tâche donnée en section 2.5 s'intéresse uniquement aux propriétés des états irréductibles d'une relation de réétiquetage. Par exemple, il est spécifié que dans un tel état, un unique sommet est élu, les autres battus. Mais le fait qu'un état σ soit irréductible est une propriété *globale* de (G, σ). La spécification de l'élection donnée par la tâche 2 ne répond pas aux questions suivantes :

– Un sommet peut-il « savoir » s'il est élu ou battu définitivement ?

– Un sommet peut-il « savoir » si le résultat de l'élection est connu ?

D'une façon similaire considérons le système de réétiquetage associé à la règle 2. La figure 5 montre une suite de réétiquetages permettant le calcul du degré des sommets d'un petit graphe. Au long de ce calcul, certains sommets portent une information numérique incorrecte, car strictement inférieure à la valeur à calculer. C'est encore une connaissance globale (terminaison des réécritures) qui permet de garantir ou non la correction de la valeur portée par

chaque sommet. Pour répondre à ces questions, Godard, Métivier et Tel [20, 11, 12, 28] proposent de considérer quatre modes de *détection de la terminaison.* Rappelons que nous travaillons toujours sous l'hypothèse que tout calcul issu d'un état initial se termine, et que nous nous concentrons sur la détection par les sommets de cette terminaison. Ces modes peuvent bien sûr être définis par des formules de logique temporelle [24] ; le fait de nous restreindre à une famille limitée de modes favorise l'étude de leurs propriétés abstraites.

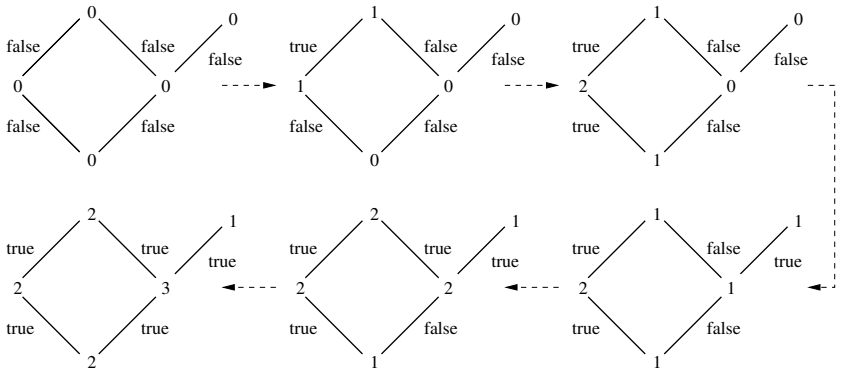

Figure 5 – Étapes d'un calcul du degré de chaque sommet

3.1. *Détection implicite de la terminaison (ITD)*

Un système de réétiquetage $\mathcal{S}$ est une réalisation d'une tâche T avec *détection implicite de la terminaison* (en abrégé **ITD**) dès que $\mathcal{S}$ satisfait aux conditions précisées en section 2.5.

Les trois autres modes considèrent chacun un « oracle » que chaque sommet peut consulter sur l'état du calcul du point de vue de la terminaison. Cet oracle prend la forme d'une fonction τ de L dans `bool`. Ces modes diffèrent par l'interprétation qui est faite de $\tau(\sigma(v))$ pour tout état σ et tout sommet v.

3.2. *Détection locale de la terminaison locale (LTD)*

Rappelons que dans une réalisation d'une tâche, la projection π sert à extraire de l'état local $\sigma(v)$ d'un sommet la valeur de sortie $\pi(\sigma(v))$. Une réalisation d'une tâche T possède la *détection locale de la terminaison* s'il existe un oracle τ tel que pour tout état accessible σ :

– Si σ est irréductible alors $\tau(\sigma(v)) = \textbf{true}$ pour tout sommet v de G et

– Pour tout sommet v de G tel que $\tau(\sigma(v)) = \textbf{true}$ et pour tout état σ' accessible à partir de σ, $\tau(\sigma'(v)) = \texttt{true}$ et $\pi(\sigma'(v)) = \pi(\sigma(v))$. Autrement dit, la participation de v au résultat final est définitivement acquise.

Notons que $\tau(\sigma(v)) = \texttt{true}$ ne signifie pas que le calcul soit terminé, ni même que l'état local en v soit stabilisé ; en effet, seule la composante $\pi(\sigma(v))$ devient constante et le sommet v peut continuer à participer à une transmission d'information (et donc voir varier son état local).

Considérons à nouveau notre exemple d'élection dans un arbre, vu cette fois selon le mode **LTD**. Définissons τ de la façon suivante :

$$\begin{aligned} \tau(\texttt{Some 0}) &= \textbf{true} \\ \tau(\texttt{None}) &= \textbf{true} \\ \tau(\texttt{Some (S } n\texttt{)}) &= \textbf{false} \end{aligned}$$

Cette définition spécifie que, si $\tau(\sigma(v)) = \textbf{true}$, alors le résultat de l'élection *pour* v , c'est-à-dire $\pi(\sigma(v))$, est définitif. Bien sûr, d'autres sommets w peuvent se trouver encore dans l'incertitude sur leur statut, exprimée par l'égalité $\tau(\sigma(w)) = \textbf{false}$, la valeur $\pi(\sigma(w))$ étant alors non significative.

Nous retrouvons bien la présentation classique des algorithmes d'élection : l'état local d'un sommet $\sigma(v)$ est dit *terminal* si $\tau(\sigma(v)) = \textbf{true}$. D'un point de vue anthropomorphiste, v « connaît » déjà son statut, mais pas celui des autres candidats.

3.3. *Détection observée de la terminaison globale (OTD)*

Dans le mode précédent l'information apportée par un diagnostic $\tau(\sigma(v)) = \mathbf{true}$ ne concerne que la valeur portée par le sommet v lui-même. Dans le mode de *détection observée de la terminaison*, si $\tau(\sigma(v)) = \mathbf{true}$, alors pour *tout* sommet w du graphe la valeur $\pi(\sigma(w))$ est constante à partir de l'état σ. En termes intuitifs, v « sait » que la partie utile du calcul $\hat{\pi}(\sigma)$ ne sera plus modifiée. Comme pour la classe **LTD**, ceci ne signifie pas que σ soit irréductible.

Formellement, un système est **OTD** s'il existe un oracle τ tel que la propriété suivante est vraie pour tout état accessible σ et tout état σ' accessible à partir de σ :

$$\begin{cases} \tau(\sigma(v)) = \mathbf{true} & \Rightarrow & \begin{cases} \tau(\sigma'(v)) = \mathbf{true} \\ \forall w \in V(G), \pi(\sigma'(w)) = \pi(\sigma(w)) \end{cases} \\ \sigma \text{ irréductible} & \Rightarrow & \tau(\sigma(v)) = \mathbf{true} \end{cases}$$

3.4. *Détection locale de la terminaison globale (GTD)*

Ce mode de terminaison est le plus restrictif : un état accessible est irréductible si et seulement s'il existe un sommet v de G tel que $\tau(\sigma(v)) = \mathbf{true}$.

Remarquons que si les inclusions de classes **OTD** $\subseteq$ **LTD** $\subseteq$ **ITD** se prouvent facilement en Coq, il n'en est pas de même pour l'inclusion **GTD** $\subseteq$ **OTD**. En effet, si $\mathcal{S}$ est une réalisation **GTD** de la tâche T, sa transformation en réalisation **OTD** de T passe par l'adjonction d'une étape de diffusion par laquelle le sommet qui détecte la terminaison des calculs par $\mathcal{S}$ avertit les autres sommets de cette terminaison. Cette diffusion sera réalisée en ajoutant par exemple un champ booléen à l'étiquetage des sommets. Le calcul **OTD** associé au système $\mathcal{S}$ comportera donc :

1) Une première phase reflétant les calculs de $\mathcal{S}$, dans laquelle tous les marquages booléens sont à **false**,

2) À la détection de la fin des calculs de $\mathcal{S}$ par un sommet v, le passage du marqueur de v à **true**,

3) La diffusion de ce booléen aux autres sommets.

4. Premiers résultats

Le modèle décrit dans les pages précédentes a été formalisé en *Coq* [23]. Outre les preuves de réalisation de tâches par des systèmes de réétiquetage, nous avons prouvé des lemmes génériques portant sur des classes de systèmes de réétiquetage associées aux divers types de synchronisation et aux divers modes de détection de la terminaison. Ces lemmes nous ont permis de montrer les deux résultats d'impossibilité décrits en sections 4.2.2 et 4.2.3.

Nous illustrons notre démarche à l'aide de quelques exemples.

4.1. *Preuves d'algorithmes*

Prouver qu'un système de réétiquetage $\mathcal{S}$ réalise bien une tâche T se fait classiquement à l'aide d'invariants. Prenons pour exemple l'élection dans un arbre décrite par la tâche 2, en utilisant le système associé à la règle 1.

Pour tout état (G, σ), nous considérons le sous-graphe *actif* G_σ composé des sommets étiquetés par le constructeur `Some` (autrement dit les sommets non encore battus) et les arêtes les reliant.

L'invariant de notre algorithme est le suivant :

Invariant 1 (Élection dans un arbre)

– *Pour tout sommet v de G, si $\sigma(v) =$* `Some` *k, alors k est le degré de v dans G_σ,*

– *G_σ contient au moins un sommet,*

– *G_σ est un sous-arbre de G.*

Cet invariant implique que si σ est un état accessible irréductible, il existe un unique sommet v étiqueté par `Some 0`, tous les autres

étant étiquetés par `None`. Ceci justifie la définition de π donnée en section 2.5.0.1.

La terminaison du système de réétiquetage est assurée par l'ordre `Some (S` k`)` $>$ `Some` $k >$ `None`, étendu aux multi-ensembles d'étiquettes. Il suffit de prouver que le multi-ensemble associé à la partie gauche de la règle 1, soit $\{\{$`Some (S` i`),Some 1`$\}\}$ est strictement supérieur à $\{\{$`Some` i`,None`$\}\}$.

Nous prouvons de plus que ce système est **LTD** avec la définition de τ donnée en section 3.2. En effet, si $\tau(\sigma(v)) = \mathbf{true}$, alors $\sigma(v)$ est soit `None` soit `Some 0` et v ne peut plus participer à un pas de réétiquetage.

La tâche d'élection présentée en section 2 est donc réalisable par un système **LC0** en mode **LTD**. Selon la même méthode, nous avons validé les algorithmes suivants :

– Calcul du degré, règle 2, mode **ITD**,

– Calcul du degré, règle 3, mode **LTD**,

– Calcul d'un arbre recouvrant, règles **LC0**, mode **LTD**.

4.2. *Résultats génériques*

À la lecture des définitions et exemples ci-dessus, il est tentant d'explorer les questions suivantes et de justifier les réponses éventuelles par des preuves formelles :

– Existe-t-il un système **LC0** de calcul du degré en mode **LTD** ?

– Peut-on se passer de l'information initiale sur les degrés lors de l'élection dans un arbre (en utilisant des règles **LC0**) ?

– Quelle est la hiérarchie entre modes de synchronisation ?

4.2.1. *Propriétés de la classe **LC0***

La réponse aux deux premières questions est négative. Nous montrons quels outils nous ont permis de prouver ces résultats en *Coq*[5].

5. L'article [8] contient la description d'une preuve antérieure de résultats très proches, mais qui n'utilisait pas les modes de détection de la terminaison.

La démarche suivie est très inspirée par l'utilisation de lemmes d'itération utilisés pour montrer que certains langages ne sont pas rationnels ou algébriques.

Rappelons que l'applicabilité d'une règle **LC0** n'est déterminée que par les étiquettes de deux sommets adjacents et de l'arête les reliant. Si une règle peut se déclencher dans un graphe étiqueté (G, σ) sur une arête (c, v), elle le pourra aussi dans le cadre d'un graphe (G', σ') étendant (G, σ). Les deux lemmes suivants utilisent cette remarque :

Lemme d'extension. Soit L un type, $\mathcal{S}$ un système de réétiquetage **LC0** sur L, et deux états (G, σ) et (G_1, σ_1) tels que (G, σ) est un sous-graphe étiqueté de (G_1, σ_1). Alors pour tout calcul $(G, \sigma) \xrightarrow{\mathcal{S}^*_G} (G, \sigma')$ il existe un calcul $(G_1, \sigma_1) \xrightarrow{\mathcal{S}^*_{G_1}} (G_1, \sigma'_1)$ tel que (G, σ') soit un sous-graphe étiqueté de (G_1, σ'_1).

Ce lemme est clairement faux dans le cas de synchronisation **LC1** et a fortiori **LC2**. Il suffit de considérer la règle 3. Si G est un sous-graphe strict de G_1 (G et G_1 étant connexes), il existe au moins un sommet v de G dont le degré est inférieur au degré de v dans G_1.

Lemme de composition. Soit L un type, $\mathcal{S}$ un système de réétiquetage **LC0** sur L, (G, σ) un graphe étiqueté, et deux sous graphes étiquetés de G : (G_1, σ_1) et (G_2, σ_2). On suppose que G_1 et G_2 sont disjoints ; soit deux calculs $(G_1, \sigma_1) \xrightarrow{\mathcal{S}^*_{G_1}} (G_1, \sigma'_1)$ et $(G_2, \sigma_2) \xrightarrow{\mathcal{S}^*_{G_2}} (G_2, \sigma'_2)$. Alors on peut construire un calcul $(G, \sigma) \xrightarrow{\mathcal{S}^*_G} (G, \sigma')$ où (G_1, σ'_1) et (G_2, σ'_2) sont des sous-graphes étiquetés de (G, σ').

Ce lemme est également faux dans le cadre **LC1** ou **LC2** : considérons à nouveau la règle 3, deux graphes disjoints G_1 et G_2, et une arête $\{v_1, v_2\}$ reliant G_1 à G_2. La présence de v_2 dans le voisinage immédiat de v_1 va évidemment perturber le calcul du degré de v_1.

Notons le haut degré de généricité de ces deux lemmes : leur énoncé est universellement quantifié sur le type d'étiquette L, sur le

système de réétiquetage considéré, puis sur un ou plusieurs graphes étiquetés.

4.2.2. *Application au calcul du degré des sommets*

Nous voulons prouver qu'il n'existe pas de système **LC0** permettant de calculer le degré des sommets d'un graphe G en mode de détection locale de la terminaison (**LTD**). Ce résultat est à comparer avec ceux de la section 4.1 et montre l'importance du mode de détection de la terminaison sur la réalisabilité d'une tâche.

Soit un type L et $\mathcal{S}$ un système **LC0**-**LTD** réalisant la tâche 1. Considérons par exemple un graphe G réduit à une arête $\{a, b\}$ et G' obtenu en ajoutant à G une arête $\{a, c\}$. Le degré de a est égal à 1 dans G et à 2 dans G'.

Nous travaillons par hypothèse sur des graphes initialement non étiquetés, aussi l'état initial de tout calcul selon S sera un graphe étiqueté uniformément par $\iota(\mathtt{tt})$. Par conséquent, l'état initial de tout calcul pour G sera un sous-graphe étiqueté de l'état initial d'un calcul associé à G'. L'état final d'un calcul pour G sera un étiquetage σ vérifiant $\pi(\sigma(a)) = 1$ et $\tau(\sigma(a)) = \mathbf{true}$. Par le lemme d'extension, nous pouvons étendre σ en un état accessible (G', σ'), mais comme $\tau(\sigma'(a)) = \tau(\sigma(a))$, $\pi(\sigma'(a))$ sera le degré du sommet a dans G', ce qui est en contradiction avec le fait que σ' est une extension de σ, par conséquent $\pi(\sigma'(a)) = \pi(\sigma(a)) = 1$.

Il est important de voir que cette argumentation utilise bien l'absence d'étiquetage en entrée, ainsi que la définition du type de synchronisation **LC0** du mode **LTD** de détection de la terminaison. En effet, nous avons prouvé que le calcul du degré des sommets d'un graphe peut se faire par un système **LC0** en mode **ITD** et aussi par un système **LC1** en mode **LTD** (voir section 4.1).

4.2.3. *Application au problème de l'élection*

En section 1, nous avons présenté une règle **LC0** permettant de construire une réalisation **LTD** de la tâche d'élection. Or cette réalisation prend comme entrée la famille des arbres où tout sommet est étiqueté par son degré dans l'arbre. Nous montrons l'importance

de cette information initiale en prouvant qu'il n'existe aucun système **LC0-LTD** $\mathcal{S}$ réalisant l'élection pour la classe des arbres non étiquetés.

Dans cette preuve, nous utilisons le lemme de composition, avec une argumentation similaire à l'exemple précédent. Supposons l'existence de $\mathcal{S}$. On construit un graphe G comme la réunion de deux graphes disjoints G_1 et G_2 reliés par une arête. Appliqué à G_1, $\mathcal{S}$ va élire un unique sommet v_1, c'est-à-dire atteindre un état σ_1 tel que $\pi(\sigma_1(v_1)) = \texttt{élu}$ et $\tau(\sigma_1(v_1)) = \textbf{true}$. De même pour G_2, σ_2 et v_2. Du fait de l'absence d'étiquetage en entrée, les états initiaux pour G_1, G_2 et G seront uniformément étiquetés par $\iota(\texttt{tt})$. Le lemme de composition est alors applicable. Il existe donc un état (G, σ) prolongeant à la fois (G_1, σ_1) et (G_2, σ_2). On aura alors $\pi(\sigma_1(v_1)) = \pi(\sigma_2(v_2)) = \texttt{élu}$ et $\tau(\sigma_1(v_1)) = \tau(\sigma_2(v_2)) = \textbf{true}$; $\mathcal{S}$ étant par hypothèse **LTD**, les images par π ne seront pas modifiées dans la suite du calcul associé à G et ce calcul se terminera avec deux sommets élus, en contradiction avec la spécification de $\mathcal{S}$.

Cette argumentation n'est bien sûr pas applicable au cas où tous les sommets sont initialement étiquetés par le degré. Si G est obtenu en reliant G_1 et G_2 par une arête, on change le degré des points d'ancrage et l'étiquetage initial dans G ne peut plus prolonger celui de G_1 et G_2.

Remarquons que l'hypothèse de non-étiquetage initial des deux exemples peut être généralisée à un étiquetage aléatoire, c'est-à-dire où tout sommet et toute arête prennent comme valeur initiale n'importe quel élément du type L_i.

4.3. *Transformation de règles*

Revenons sur l'inclusion **GTD** $\subseteq$ **OTD**. Nous avons prouvé cette inclusion dans le cadre des systèmes **LC0**. La transformation décrite en section 3.4 s'apparente à une transformation de règles certifiée dans la mesure où la terminaison et la correction du système **OTD** par rapport à une tâche donnée sont dérivées de celles du système **GTD** de départ.

5. Modélisation en *Coq*

Nous présentons quelques aspects de la bibliothèque *Coq* `Loco` consacrée à la théorie des calculs locaux [9]. Il s'agit d'un travail en cours et quelques choix de représentation sont susceptibles d'être modifiés, sans remettre en cause les possibilités évoquées plus haut.

La partie de la bibliothèque consacrée aux exemples contient essentiellement ceux présentés dans cet article. Ces exemples sont de deux types : preuve qu'un système de réétiquetage est bien la réalisation d'une tâche donnée (section 4.1) et les deux preuves d'impossibilité (sections 4.2.2 et 4.2.3).

5.1. *Graphes étiquetés*

Notre développement représente les graphes comme des ensembles finis de sommets et d'arêtes, et les étiquetages par des fonctions finies, en utilisant les bibliothèques `FSets` et `FMaps` écrites par Letouzey. Nous avons choisi une représentation très proche de la définition mathématique usuelle, afin que les définitions de notions comme la connexité, l'acyclicité, etc., soient facilement reconnaissables. La non-orientation des arêtes est obtenue en normalisant leur représentation à l'aide d'un ordre total décidable sur le type servant à représenter les sommets.

La mécanisation de la théorie des graphes n'étant pas un objectif prioritaire, cette partie de notre développement se construit de façon *ad-hoc*, les priorités étant déterminées par l'étude des systèmes de réétiquetage. Certains lemmes du folklore [6] sur les graphes sont provisoirement admis.

Plusieurs contributions écrites en *Coq* [18, 17, 21, 5] utilisent ou prennent comme sujet d'étude la théorie des graphes. Il serait intéressant d'étudier comment intégrer ces formalisations à notre travail.

6. Exemple : tout graphe acyclique non vide possède au moins un sommet de degré ≤ 1.

5.2. *Systèmes de réétiquetage de graphes*

Cette partie est consacrée à la théorie des calculs locaux. Les systèmes de réétiquetage sont définis à l'aide d'une sémantique relationelle : à tout système $\mathcal{S}$ sur le type L on associe une fonction qui à tout graphe G associe une relation binaire sur $\Sigma_{G,L}$.

Tous les résultats présentés dans cet article sont formellement prouvés. Dans l'état actuel de notre travail, ce sont les systèmes **LC0** qui ont bénéficié de la plupart des développements. Ceux-ci comprennent entre autres une preuve que tout système **LC0** définit bien un système de réétiquetage localement engendré au sens de la définition en section 2.3, la preuve des lemmes d'extension et de composition, ainsi qu'une tactique pour réduire une preuve de terminaison à la comparaison locale entre anciennes et nouvelles étiquettes (section 4.1). De même, l'inclusion **GTD** $\subseteq$ **OTD** est prouvée dans le cas particulier **LC0**.

6. Développements futurs

L'état actuel de notre développement montre qu'il est possible de mécaniser la théorie des calculs locaux et de faire communiquer dans un même environnement preuves d'algorithmes et études de propriétés abstraites de diverses classes d'algorithmes, ces dernières pouvant d'ailleurs servir à montrer que telle spécification est irréalisable sous certaines conditions.

L'évolution de nos travaux doit se faire dans plusieurs directions : prise en compte de tous les types de synchronisation, écriture de tactiques spécialisées dans le domaine des calculs locaux, et intégration sous forme d'outils effectifs des développements théoriques.

6.1. *Règles LC1 et LC2*

La plupart de nos exemples et outils ont été construits sur des règles **LC0**. L'extension aux règles **LC1** et **LC2** posera quelques problèmes dûs à la grande expressivité de ces règles.

Un exemple typique de règle (**LC2**) est le suivant, emprunté à un système **GTD** de calcul distribué d'arbre recouvrant [23] ; les sommets sont étiquetés dans un type à quatre constructeurs `{A,A',N,F}` et les arêtes par des booléens.

Règle 4 (Construction d'un arbre recouvrant, GTD)

– Si le sommet c est étiqueté par `A` *ou* `A'`*, et si c a un voisin v étiqueté* `N`*, alors l'étiquette de v devient* `A'` *et l'arête* $\{c, v\}$ *prend l'étiquette* ***true****,*

– Si c est étiqueté `A'`*, s'il n'a aucun voisin étiqueté* `N` *et si tous les voisins auxquels c est relié par une arête étiquetée* ***true*** *sont étiquetés* `F`*, alors l'étiquette de c devient* `F`*.*

Un tel énoncé, parfaitement compréhensible, comporte une quantification sur v à portée dynamique, qui peut rendre complexe la traduction vers un système de réétiquetage. Cette traduction doit de plus s'accompagner d'une preuve que le système engendré respecte toujours les conditions de la section 2.3.

L'équivalent pour **LC1** et **LC2** des lemmes d'extension et de composition (4.2.1) reste à construire.

6.2. *Tactiques et automatismes*

Dans l'état actuel de notre développement, beaucoup de preuves sont longues et manuelles, du fait du peu de tactiques spécialisées déjà développées.

Des améliorations pourraient être apportées dans les domaines suivants :

Preuves de terminaison. Une tactique permettant de prouver la terminaison d'un système de réétiquetage consiste à définir un ordre bien fondé sur le type servant à étiqueter les sommets et arêtes, puis à montrer que le multi-ensemble des étiquettes modifiées par un pas de réécriture décroît strictement durant ce pas. Les techniques de preuve de terminaison sont alors

très voisines de celles utilisées pour les réécritures de termes et pourraient bénéficier d'une plus grande automatisation [15, 30].

Génération d'obligations de preuve. Nous avons vu dans les exemples précédents que l'écriture des invariants et celle des règles d'étiquetage utilisent des systèmes de variables différents. Les règles ont pour portée une arête ou plus généralement une étoile de centre arbitraire, alors que les invariants portent sur des propriétés globales de l'état courant. Pour faciliter les preuves de réalisation, il est souhaitable de concevoir des outils pour générer les obligations de preuve liées au maintien des invariants. Cansell et Méry [7] utilisent des invariants de liaison pour relier propriétés locales et globales, et ces techniques devront être adaptées à notre travail.

6.3. *Théorie des calculs locaux*

Les travaux théoriques sur les calculs locaux [13, 20, 10] comportent de nombreux résultats que nous souhaitons intégrer dans notre développement. Par exemple, la notion d'épimorphisme de graphe étiqueté est un outil mathématique permettant de prouver que certaines spécifications ne sont pas réalisables par des systèmes **LC1** ou **LC2**. Il serait très intéressant de disposer de tactiques pour, à partir d'une tâche donnée, explorer divers choix de types de synchronisation et de modes de détection de la terminaison et, en cas d'impossibilité de réaliser la tâche sous certaines conditions, aider à construire un contre-exemple.

7. Conclusion

Le modèle des calculs locaux permet de traiter dans un formalisme simple plusieurs notions importantes en algorithmique distribuée : répartition des données initiales, types de synchronisation, détection de la terminaison (locale ou globale) des calculs. Loin d'être indépendantes, ces notions interagissent fortement et conditionnent la réalisabilité de tâches.

Le projet que nous avons entamé montre qu'il est possible, dans un même environnement, d'écrire des spécifications, d'en certifier formellement des réalisations, mais aussi de prouver et d'appliquer des résultats abstraits, pour déterminer avec quels types d'algorithmes elles sont ou non réalisables.

L'enseignement de l'algorithmique distribuée dans quatre universités ou écoles d'ingénieurs s'appuie déjà sur l'outil de simulation et visualisation Visidia [23], fondé sur les calculs locaux. Cet enseignement pourra se renforcer d'une partie consacrée à la preuve de tels algorithmes.

En général l'enseignement de la théorie de la programmation et notamment des modèles de calcul s'appuie sur des concepts assez abstraits. En particulier les preuves de non-calculabilité classiques, pour élégantes qu'elles soient, « passent » difficilement. Sans prétendre les remplacer, nous pensons qu'une réduction de ces résultats à un formalisme comportant une hiérarchie de modes de calculs exprimés simplement permet d'aborder concrètement la notion de puissance d'expression. Le fait que ces preuves portent sur des exemples dont l'exécution peut être *montrée* (avec des outils comme Visidia) et que ces mêmes preuves soient rejouables sur machine accentue leur caractère concret.

L'exemple du calcul du degré est particulièrement intéressant car sa spécification est très simple et ses diverses variantes (**LC0**, **LC1**, **ITD**, **LTD**) montrent à quel point les hypothèses de localité relativisent la notion de calculabilité. Par exemple, si l'applicabilité d'une règle de réétiquetage est décidable — ce qui est le cas de tous les exemples que nous connaissons — l'irréductibilité d'un état sur un graphe fini l'est aussi. La prise en compte des divers modes de détection de la terminaison montre que même si une information est calculable, l'*accès à cette information* par les sommets doit être pris en compte.

Pour finir, les articles théoriques dont nous nous sommes inspirés proposent des définitions semi-formelles : usage de langue naturelle, hypothèses implicites, langage impératif pour décrire des changements d'état. La « mécanisation » de cette théorie en *Coq* a rendu

nécessaires quelques modifications de définitions (sections 2.5 et 3) et nous a permis de construire de nouvelles preuves, présentées en section 4.2. Les quelques modifications mineures de définitions ont été rendues nécessaires lors de preuves effectuées sous *Coq* par l'apparition de sous-buts insolubles. Citons par exemple la définition des systèmes localement engendrés de la section 2.3 : l'impossibilité de prouver formellement que notre représentation des règles **LC0** définit bien des systèmes localement engendrés nous a conduits à proposer quelques retouches aux auteurs de la version papier. Le dialogue qui s'en est suivi permet de déduire deux faits encourageants : la théorie des calculs locaux s'adapte bien à la preuve formelle ; d'autre part, elle fournit aux chercheurs intéressés à la preuve d'algorithmes un champ d'application très vaste.

Remerciements

Nous remercions vivement Emmanuel Godard, Yves Métivier, Mohamed Mosbah, Xavier Urbain et les deux rapporteurs anonymes pour leurs éclairantes remarques sur le fond et la forme de cet article.

Références

[1] M. Bauderon, S. Gruner, and M. Mosbah. A new tool for the simulation and visualization of distributed algorithms. Technical Report 1245-00, LaBRI, 2000. MFI'01, Toulouse, 21-23 May 2001.

[2] M. Bauderon, Y. Métivier, M. Mosbah, and A. Sellami. From local computations to asynchronous message passing systems. Technical Report RR-1271-02, LaBRI, 2002. http ://www.labri.fr/visidia/.

[3] M. Bauderon and M. Mosbah. A unified framework for designing, implementing and visualizing distributed algorithms. In *International Workshop on Graph Transformation and Visual Modeling Techniques*, number 72 in LNCS, 2002.

[4] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art : The Calculus of Inductive*

Constructions. Texts in Theoretical Computer Science. Springer Verlag, 2004.

[5] S. Blazy, B. Robillard, and E. Soutif. Vérification formelle d'un algorithme d'allocation de registres par coloration de graphe. In *JFLA (Journées Francophones des Langages Applicatifs)*, pages 31–46, Etretat France, 2008. INRIA.

[6] R.S. Boyer and J.S. Moore. *A Computational Logic Handbook*. Academic Press, 1988.

[7] D. Cansell and D. Méry. *Logics of Specification Languages*, chapter The Event-B Modelling Method : Concepts and Case Studies, pages 47–152. Springer Verlag, 2007.

[8] P. Castéran, V. Filou, and M. Mosbah. Certifying distributed algorithms by embedding local computation systems in the coq proof assistant. In *Symbolic Computation in Software Science (SCSS'09)*, 2009.

[9] P. Castéran and V. Filou. Loco : A Coq Library on Local Computation Systems. http ://www.labri.fr/∼casteran/Loco.

[10] J. Chalopin. *Algorithmique Distribuée, Calculs Locaux et Homomorphismes de Graphes*. PhD thesis, Université Bordeaux 1, 2006.

[11] J. Chalopin, E. Godard, and Y. Métivier. Local terminations and distributed computability in anonymous networks. In *DISC*, volume 5218 of *Lecture Notes in Computer Science*, pages 47–62. Springer, 2008.

[12] J. Chalopin, E. Godard, Y. Métivier, and G. Tel. About the termination detection in the asynchronous message passing model. In *SOFSEM (1)*, volume 4362 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2007.

[13] J. Chalopin and Y. Métivier. An efficient message passing election algorithm based on Mazurkiewicz's algorithm. *Fundam. Inf.*, 80(1-3) :221–246, 2008.

[14] Chin-Tsun Chou. Mechanical verification of distributed algorithms in higher-order logic. *The Computer Journal*, 38(2) :152–161, 1995.

[15] É. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. In B. Konev and F. Wolter, editors, *6th International Symposium on Frontiers of Combining Systems (FroCos 07)*, Lecture Notes in Artificial Intelligence, pages 148–162. Springer Verlag, Sep. 2007.

[16] Deploy European Community Project, www.event-b.org. *Event-B and the Rodin Platform.*

[17] J-F. Dufourd. Discrete Jordan Curve Theorem : a proof formalized in Coq with hypermaps. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, pages 253–264, 2008.

[18] J. Duprat. A Coq toolkit for graph theory. Technical report, École normale supérieure de Lyon, 2001.

[19] E. Gascard and L. Pierre. Formal proof of applications distributed in symmetric interconnexion networks. *Parallel Processing Letters*, 13(1) :3–18, 2003.

[20] E. Godard, Y. Métivier, and G. Tel. Termination detection of distributed tasks. Technical Report 1418-06, LaBRI, UMR 5800, 2006.

[21] G. Gonthier. A computer-checked proof of the Four Colour Theorem. Technical report, Microsoft Research Cambridge, 2005.

[22] M. Gordon and T. Melham. *Introduction to HOL.* Cambridge University Press, 1993.

[23] LaBRI Distributed Algorithms Group. The Visidia project. http ://www.labri.fr/visidia.

[24] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems, 16(3)*, pages 872–923, May 1989.

[25] I. Litovsky, Y. Métivier, and E. Sopena. Different local controls for graph relabelling systems. *Mathematical Systems Theory*, 28 :41–65, 1995.

[26] I. Litovsky, Y. Métivier, and E. Sopena. Graph relabelling systems and distributed algorithms. In H. Ehrig, H.J. Kreowski,

U. Montanari, and G. Rozenberg, editors, *Handbook of graph grammars and computing by graph transformation*, volume 3, pages 1–56. World Scientific, 1999.

[27] A. Mazurkiewicz. Distributed enumeration. *Information Processing Letters*, 61 :233–239, 1997.

[28] Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *SIROCCO*, pages 237–251. Carleton Scientific, 2000.

[29] "Coq Development Team". *The Coq Proof Assistant Reference Manual*. coq.inria.fr.

[30] X. Urbain. Modular and incremental automated termination proofs. *Journal of Automated reasoning*, 32 :315–355, 2004.