http://www-lipn.univ-paris13.fr/~petrucci/PAPERS
In Journal Européen des Systèmes Automatisés, volume 42, numéro 4, pages 459-478, Hermès, 2008.

Application des méthodes formelles à la robotique modulaire

Méthodes formelles pour l'analyse des robots autonomes et modulaires

Lom Messan Hillah* — **Fabrice Kordon*** — **Laure Petrucci****

* *Université P. & M. Curie - Paris 6, CNRS UMR 7606 - LIP6/MoVe*
4, place Jussieu, F-75252 Paris CEDEX 05, France
lom-messan.hillah@lip6.fr, fabrice.kordon@lip6.fr

** *LIPN, CNRS UMR 7030, Université Paris XIII*
99, avenue Jean-Baptiste Clément, F-93430 Villetaneuse, France
Laure.Petrucci@lipn.univ-paris13.fr

RÉSUMÉ. *Cet article se propose de montrer l'intérêt de l'application des méthodes formelles à la spécification et l'analyse des systèmes robotiques modulaires. Après avoir introduit le problème de la robotique modulaire et le modèle des réseaux symétriques, nous détaillons une étude de cas.*

ABSTRACT. *This paper aims at highlighting the advantages of applying formal methods to the specification and analysis of modular robotics. After introducing the modular robotics problem and the symmetric nets model, we detail a case study.*

MOTS-CLÉS : *robotique modulaire, méthodes formelles, vérification.*

KEYWORDS: *self-reconfiguring modular robotics, formal methods, verification.*

1. Introduction

La robotique recouvre divers champs d'application, des humanoïdes aux sous-marins en passant par la médecine et la construction (Garcia *et al.*, 2007). Parmi les tendances actuelles, les systèmes inspirés par la biologie suscitent un grand intérêt. En la matière, la robotique modulaire est particulièrement remarquable (Rus *et al.*, 2002).

Les systèmes auto-reconfigurables sont perçus comme un moyen de réaliser un compromis bénéfique entre coût et efficacité d'action. Ces systèmes sont capables de s'adapter à leur environnement physique et réaliser diverses tâches spécifiques. Une autre caractéristique intéressante est qu'ils sont constitués d'entités réactives, aux comportements similaires.

La conception et l'implémentation de tels systèmes, par essence répartis, doivent être robustes, durables, tolérantes aux pannes et passer à l'échelle. Elles requièrent de plus des algorithmes et méthodes indépendants des plates-formes et architectures cibles (Kurokawa *et al.*, 2002). De ce point de vue, la validation des algorithmes de contrôle constitue un problème non encore résolu lié à la conception. Elle se heurte principalement:

– au fort couplage qui existe entre le matériel et le logiciel,

– au nombre important de modules qui peuvent être mis en jeu,

– au parallélisme de comportement,

– à la détermination de la configuration optimale pour chaque tâche à réaliser,

– à la détermination d'un modèle de communication efficace,

– aux contraintes d'espace et de temps.

Des abstractions de haut niveau sont donc nécessaires pour modéliser et traiter ces problèmes à travers des couches de préoccupations compatibles afin de mieux maîtriser leur complexité. De par leur nature abstraite, ces considérations de modélisation sont fréquemment sujettes à des erreurs d'évaluation des aspects pertinents à intégrer par couche dans les modèles. Ce problème peut être surmonté en s'appuyant sur l'utilisation des méthodes formelles pour la spécification et l'analyse.

Nous proposons dans cet article l'utilisation de méthodes formelles, mathématiquement fondées, offrant des capacités de spécification et de vérification formelles des systèmes parallèles, concurrents et répartis (Gogen *et al.*, 1997; Girault *et al.*, 2003; Kordon, 2007).

En particulier, des méthodes reposant sur des techniques performantes de « model checking » (on utilise parfois le terme vérification de modèle) peuvent être utilisées (Bérard *et al.*, 2001). Celles tirant parti des symétries de comportement (Thierry-Mieg *et al.*, 2004) permettent d'analyser efficacement les systèmes présentant une telle caractéristique. La réduction importante de l'explosion combinatoire de l'espace d'états résultant de ces techniques, associée à l'important outillage logiciel bénéficiant des récentes architectures massivement parallèles (Hamez *et al.*, 2007) rend accessible l'analyse de systèmes de grande taille. De ce point de vue, les méthodes formelles sont

donc maintenant mieux à même d'aider les concepteurs en robotique mobile à approfondir la compréhension du comportement de leurs algorithmes de contrôle.

Nous souhaitons montrer les bénéfices des dernières avancées dans ces approches formelles pour la spécification de systèmes modulaires et autonomes, où les modules exhibent un comportement similaire dans la majeure partie de leur spécification. L'idée est d'exploiter la régularité de leur architecture pour exhiber des symétries. Nous utilisons ces symétries pour optimiser la vérification par model checking de spécifications de très grande taille. Cette investigation sera illustrée par une étude de cas mettant en œuvre deux reconfigurations permettant de franchir un obstacle ou de parcourir un environnement donné. Ces reconfigurations impliquent un changement de mode de locomotion.

Dans la suite de l'article, la section 2 présente les enjeux liés à la spécification et au développement des systèmes robotiques modulaires. La section 3 détaille une méthode de spécification formelle basée sur les Réseaux Symétriques pour l'analyse formelle prenant en compte les considérations mentionnées plus haut. Puis, la section 4 présente une application par une étude de cas sur un algorithme de contrôle de configuration. La section 5 fait un parallèle avec l'analyse des systèmes de transport intelligents, avant une conclusion ébauchant quelques perspectives.

2. La robotique modulaire

L'intérêt pour la R&D en robotique modulaire est guidé par la nécessité de trouver un équilibre entre le coût et la capacité d'être multi-tâche, tout en améliorant l'efficacité d'action. Lorsque l'on envoie un robot en mission dans un environnement hostile (exploration spatiale, ruines d'une catastrophe, ...), il est préférable de considérer l'emploi d'un robot modulaire et auto-reconfigurable (ci-après abrégé par RMA).

Le développement des robots pose des problèmes de mécanique, d'énergie, d'électronique et d'informatique. Le fort couplage entre le matériel et le logiciel exige une démarche de prototypage et de vérification permettant l'ajustement au mieux des critères pour l'efficacité et la survie du robot. En particulier, un RMA en déplacement doit disposer de plusieurs modes de locomotion, choisir le mode approprié en fonction de l'environnement, changer effectivement de mode et recouvrer d'une défaillance imprévue mais non fatale (Shen *et al.*, 2006).

Le contrôle de l'auto-reconfiguration suit généralement, une approche soit centralisée, soit répartie. La première fait appel aux concepts de planification. Son calcul demeure très sensible à l'explosion combinatoire relative au couple *nombre de modules - nombre de séquences d'actions*. Elle souffre également d'inefficacité dans la stratégie de reconfiguration, du fait de l'imprécision souvent constatée dans la représentation du modèle du système. La seconde, en répartissant le contrôle de la reconfiguration entre les modules, permet d'éviter dans une grande mesure les écueils précédemment mentionnés. Cependant, le problème principal qu'elle soulève concerne la coordination.

Le RMA de (Butler *et al.*, 2002) s'inspire du concept d'automate cellulaire, plus tard dérivé en *Molécule* (Kotay *et al.*, 2005). Ce modèle possède une architecture physique évoluant en treillis. Les cellules, de forme cubique, possèdent une connectivité physique sur toutes leurs faces. Dans cette approche, l'objectif visé est l'utilisation un algorithme générique, c'est-à-dire indépendant de la plate-forme utilisée.

Cet algorithme calcule la prochaine configuration à partir de la configuration courante en évaluant simultanément l'ensemble des cellules. De ce fait, il calcule l'ensemble des séquences d'actions pour atteindre cette configuration. Il a également été appliqué avec beaucoup d'adaptations au MTRAN de (Kurokawa *et al.*, 2002). Les modules du MTRAN disposent de six faces par lesquelles ils se connectent, communiquent et partagent l'énergie. Ce modèle possède une architecture pouvant évoluer en treillis et en chaîne. L'adaptation fut nécessaire à cause de cette différence de structure, car le modèle géométrique de l'automate cellulaire suppose une forme cubique disposant d'une connectivité sur toutes les faces.

En matière de reconfiguration statique, l'ATRON (Christensen *et al.*, 2006) est un système intéressant qui utilise un méta-module pour diriger la reconfiguration, mais le choix et le contrôle distribués du méta-module restent à déterminer.

Pour ce qui concerne l'établissement des critères dirigeant la reconfiguration pour la locomotion, le SuperBot (Shen *et al.*, 2006) inspiré du MTRAN, semble plus évolué. Il présente six modes de locomotion choisis selon des critères relatifs à la pente et aux tailles des obstacles du terrain, la vitesse supportée, l'efficacité de la consommation d'énergie, la capacité de tourner et de recouvrer d'une défaillance. Il revendique des algorithmes de contrôle réparti qui favorisent la communication et la coordination entre les modules. En effet, dans le MTRAN, la communication locale et globale intermodule est un support fort pour la coordination entre les modules. Cependant ces algorithmes ne sont pas présentés.

Dans tous ces travaux, il existe un souci permanent d'utiliser des algorithmes de contrôle réparti, que ce soit pour les architectures évoluant en treillis ou en chaîne. Pour mieux maîtriser l'explosion combinatoire liée au calcul des chemins de configuration sur des entités présentant des similitudes de comportement, l'utilisation des *Réseaux Symétriques* s'avère intéressante. De plus, la vérification des propriétés permet de mieux cerner les interactions au cours de l'évolution des systèmes.

Dans la section suivante, nous présenterons le modèle des *Réseaux Symétriques* ainsi que ses capacités de modélisation et d'analyse.

3. Les Réseaux Symétriques (Symmetric Nets)

L'objectif de cette section est de présenter informellement les Réseaux Symétriques et les techniques d'analyse associées. Des définitions formelles sont fournies dans (Chiola *et al.*, 1991; Girault *et al.*, 2003).

3.1. *Présentation du formalisme*

Les Réseaux Symétriques sont des réseaux de Petri colorés ; les jetons qui circulent dans les places comportent une valeur. Ainsi, les places du modèle sont typées par les valeurs des jetons qu'elles peuvent contenir. Mais, contrairement aux réseaux colorés « classiques » (Jensen, 1992), seuls des types de données simples (énumérés finis, intervalles, n-uplets, et les combinaisons de ceux-ci), ainsi que quelques fonctions de manipulation prédéfinies (prédécesseur, successeur, sélecteur dans un n-uplet) sont autorisées. Enfin, une constante dite « de diffusion » permet de générer un jeton par valeur possible dans un type de données. Cette constante est nécessaire pour modéliser des protocoles pour lesquels une station peut avoir à envoyer des messages à toutes les autres stations d'un réseau.

Illustrons l'utilisation des Réseaux Symétriques au moyen d'un exemple simple. Le réseau de la figure 1 représente une classe de processus légers (représentée par le type P) accédant à une ressource critique **CR**. Chaque processus léger peut obtenir une valeur du type Val depuis la ressource **CR**. Les constantes **PR** et **V** permettent de paramétrer le nombre de processus légers et l'intervalle des valeurs typant la ressource critique.

Figure 1. *Exemple simple de Réseau Symétrique.*

Dans l'exemple, les places **out** (de type P) et **compute** (de type $D = P \times Val$) représentent le comportement de la classe de processus légers. Quand un processus léger est dans l'état **out**, il n'a rien lu dans la variable partagée **CR**. Par contre, lorsqu'il arrive dans l'état **compute**, il a récupéré sa valeur. La place **Mutex** exprime l'exclusion mutuelle sur la variable **CR**.

On dispose, dans l'état initial du système, de **PR** processus légers numérotés par la valeur contenue dans le jeton qui les représente (d'où le marquage initial $< P.all >$). De même, le fait que toutes les valeurs possibles soient représentées dans le marquage de la place **CR** permet de modéliser toutes les possibilités de valeurs lues dans la variable partagée qu'elle représente.

Les transitions décrivent les évolutions du système. Une transition est déclenchée lorsque les places en précondition (c'est-à-dire en entrée de la transition) possèdent un nombre de jetons suffisant. Par exemple, **inCS** est déclenchable dès qu'il y a un jeton dans les places **out**, **CR** et **Mutex** (ce qui est le cas dans l'état initial du système).

Lorsque **inCS** est déclenchée, elle consomme un jeton dans les trois places précondition. Comme le réseau est coloré, les valeurs de ces jetons sont associées aux variables en précondition de la transition. Ainsi, les variables p et v sont liées respectivement au contenu du jeton des places **out** et **CR**. Le jeton de **Mutex** ne contenant pas de valeur (c'est un jeton « non coloré »), la notion de liaison n'a pas de sens. Le déclenchement de **inCS** provoque également la production d'un jeton composé dans la place postcondition (en sortie) **compute**. Ce jeton composé (respectant la composition de type associée à la place) est construit à l'aide des valeurs associées aux variables p et v.

3.2. *Relation entre Réseaux Symétriques et réseaux Places/Transitions*

Un réseau de Petri coloré peut être *déplié* en un réseau de Place/Transition équivalent. Les réseaux Places/Transitions sont des réseaux de Petri dont les jetons n'ont pas de valeur (les places ne sont donc pas typées).

Dans le cas des *Réseaux Symétriques*, la transformation en un réseau de Petri Place/Transition suit un algorithme particulièrement simple. Chaque place du Réseau Symétrique est transformée en un ensemble de places dans le réseau Place/Transition, chacune étant associée à une valeur du type de la place dans le Réseau Symétrique. De même, les transitions du réseau Place/Transition représentent chacune une instanciation possible des variables associées aux places précondition et postcondition. Le réseau Place/Transition de la figure 2 correspond au dépliage du *Réseau Symétrique* de la figure 1 avec $P = [1..2]$ et $Val = [1..2]$.

Figure 2. *Réseau de Petri Place/Transition obtenu par dépliage du modèle de la figure 1.*

La taille du réseau de Petri Place/Transition associé à un Réseau Symétrique est une fonction polynomiale de la taille du réseau initial. Plus spécifiquement, cette taille dépend de celle des domaines de couleurs. Prenons un exemple sur le réseau de la figure 1. Supposons, $PR = V = 20$. Le nombre de transitions du réseau déplié (sans optimisation) est : $NbT_d = 2 \times PR \times V = 800$. Le nombre de places est :

$NbP_d = PR + PR \times V + V + 1 = 441$. Le nombre total de nœuds du réseau déplié est donc $Nb_d = 1241$. La taille du réseau initial étant $Nb_i = 6$, nous pouvons exprimer : $Nb_d = 5x^3 + 4x^2 + 2x + 5$, avec $x = Nb_i$.

Cependant, des techniques de représentation s'appuyant sur des diagrammes de décision permettent de manipuler ces réseaux dépliés, même lorsqu'ils sont de grande taille (Kordon *et al.*, 2006). Ces optimisations sont réalisées en fonction du marquage initial. Elles permettent d'élaguer des transitions « mortes » reliées à des places dont on calcule qu'elles ne contiendront jamais de jetons.

Si les Réseaux Symétriques présentent l'avantage d'être compacts, les réseaux de Petri Places/Transitions permettent quant à eux de calculer des propriétés structurelles, i.e., sans déployer l'espace d'état. Parmi ces propriétés, les invariants (tels que les familles génératrices d'invariants) sont des formules valides quel que soit l'état initial du système (Girault *et al.*, 2003).

Par exemple, sur le modèle de la figure 2, la formule suivante est vérifiée :

compute_1_1+**compute_1_2**+**compute_2_1**+**compute_2_2**+**Mutex**$= 1$

Cela signifie que l'on ne peut avoir plus d'un processus en section critique. Par conséquent, l'exclusion mutuelle est préservée pour ce modèle.

3.3. *Graphe des marquages symboliques*

Le model checking est une méthode d'analyse classique pour les systèmes parallèles. L'objectif est de vérifier automatiquement qu'un modèle représentant le comportement d'un système satisfait certaines propriétés. Ces propriétés sont exprimées par des formules (e.g., CTL, LTL) qui sont vérifiées en analysant l'espace d'états de la spécification qui a été au préalable construit de manière exhaustive. Dans notre approche de model checking, nous ne nous intéressons qu'aux systèmes de taille finie.

Le modèle des réseaux de Petri (et ses extensions) offre une notion similaire appelée *graphe des marquages accessibles*. Ce graphe peut-être généré automatiquement en « jouant » le *Réseau Symétrique* pour obtenir toutes les évolutions possibles du système.

La figure 3 représente le graphe des marquages accessibles du Réseau Symétrique de la figure 1 lorsque les constantes **PR** et **V** valent 2. Le graphe des marquages comporte 5 états (l'état initial est représenté par un double cercle). Le nombre d'états est donné par la formule $PR \times V$. Ainsi, même pour une spécification simple comme celle de notre exemple, il existe des configurations qui ne peuvent être représentées directement en mémoire.

Un grand avantage des *Réseaux Symétriques* est de permettre d'utiliser un *graphe des marquages symboliques*. Il s'agit d'une représentation condensée des états du système qui s'appuie sur les symétries que l'on peut y détecter. Ainsi, un *état symbolique* du *graphe des marquages symboliques* ne représente pas une configuration concrète

Figure 3. *Graphe des marquages accessibles du modèle de la figure 1.*

mais un ensemble de configurations ayant la même structure. En cas de besoin, on peut déduire un ensemble *d'états concrets* (chacun représentant une configuration) à partir d'un *état symbolique*. La figure 4 représente le *graphe des marquages symboliques* du modèle de la figure 1, soit une représentation compacte du graphe des marquages de la figure 3. Nous sommes dans un cas favorable car ce graphe est indépendant de l'étendue des types P et Val (et donc des constantes **PR** et **V**).

Figure 4. *Graphe des marquages symboliques du modèle de la figure 1.*

Les états du graphe de la figure 4 s'interprètent comme suit. Dans l'état initial (double cercle), toutes les valeurs possibles du domaine P sont stockées dans la place **out**. Il en est de même avec les valeurs possibles de Val dans la place **CR**. Le déclenchement de la transition **InCS** prend une valeur dans la place **out** et une valeur dans la place **CR**. L'autre état du graphe exprime donc toutes les possibilités de marquage de la place **compute** aux permutations près des valeurs prises dans **out** et **CR**. Ainsi, on exprime que l'on a dans **compute** un jeton composé d'une valeur de P et d'une valeur de Val sans se préoccuper de cette valeur. La place **out** contient toutes les valeurs de P moins le jeton qui a été consommé lorsque **InCS** a été déclenchée. On ne se préoccupe pas non plus de la valeur consommée. Il en est de même avec le marquage de la place **CR**. L'état initial de ce *graphe des marquages symboliques* représente l'unique état initial du système (la place en double-cercle de la figure 3). Par contre, l'autre état symbolique représente tous les autres états du système.

Cette technique de représentation symbolique s'appuie sur la notion de symétries dans le modèle (Thierry-Mieg *et al.*, 2003). Elle permet de représenter des espaces d'états de grande taille puisque l'on observe souvent un gain exponentiel entre le

nombre d'états symboliques dans le *graphe des marquages symboliques* et le nombre d'états concrets dans le graphe des marquages accessibles (Hugues *et al.*, 2004).

En conclusion, cette représentation ensembliste des états dans le graphe des marquages symboliques est extrêmement efficace, en particulier lorsque les systèmes modélisés possèdent des symétries. C'est le cas en robotique modulaire puisque les modules, qui embarquent les mêmes algorithmes, sont relativement interchangeables. Nous nous appuyons sur la notion de *classes d'équivalence* entre des sous-ensembles des acteurs du système qui exhibent des comportements similaires.

4. Application à une étude de cas

Nous nous proposons d'illustrer l'utilisation d'une approche formelle basée sur les Réseaux Symétriques au moyen d'un cas d'étude simple.

Considérons un système composé de robots identiques lui permettant d'adopter un mode de locomotion multipède ou ondulaire. Ce système évolue dans un environnement sur lequel il dispose d'une base de connaissances peu étendue. Lorsqu'un obstacle est rencontré, une alternative s'offre au système : i) il existe une solution connue pour franchir l'obstacle ou le contourner ou ii) aucune solution n'est applicable et le système s'arrête en attente d'une intervention humaine. Les mécanismes d'apprentissage, relevant des techniques d'intelligence artificielle, sont hors du champ de cette étude et ne seront donc pas considérés.

Les modules sont interconnectés par un canal de communication qui fait transiter les messages. Nous faisons abstraction de la mise en œuvre de ce dispositif et des protocoles complexes qui pourraient être utilisés. Ce qui est modélisé ici concerne l'émission du message, son transit via le canal (qui peut contenir un nombre maximum de messages), et sa réception par un module.

Nous considérons également que les modules robotiques sont équipés d'un dispositif de perception de l'environnement. Sa technologie de mise en œuvre n'est pas représentée dans ce modèle, mais est supposée suffisamment fiable pour assurer son utilisation. Ce dispositif est à l'origine de l'émission des messages faisant cas de la rencontre d'un obstacle.

Trois entités sont donc clairement identifiées dans ce modèle : les modules robotiques, les dispositifs de communication et de perception de l'environnement. Chacune est sujette à une abstraction nous permettant de l'intégrer dans une couche de raffinement compatible avec l'objectif d'analyse.

4.1. *Objet de l'analyse*

Nous nous intéressons à la validité d'un algorithme de contrôle réparti dans les modules leur permettant de déterminer la configuration appropriée pour le franchissement d'un obstacle.

La spécification de cet algorithme démarre au plus haut niveau d'abstraction possible permettant déjà une première analyse intéressante. Par exemple, quels traitements effectue le système quand il est en mouvement ou à l'arrêt ? L'identification ou non d'un obstacle permet-elle une convergence de décision de l'ensemble des modules ?

L'intérêt de procéder par raffinements successifs est tout d'abord analogue à celui que nous aurions à concevoir un programme dans lequel les traitements complexes sont délégués à des fonctions et procédures. La spécification de ces procédures est détaillée et ainsi de suite. Associée à la similitude (ci-après également appelée *symétrie*) de comportement des modules robotiques, cette démarche permet la maîtrise de l'explosion combinatoire. Elle permet également, dans une certaine mesure, la préservation de propriétés au cours de l'évolution du modèle.

L'algorithme 1 présente le mécanisme général de contrôle des modules, excepté le dispositif de communication. Dans cet algorithme, l'instruction *Arrêt du système* est une abstraction de l'attente d'une intervention humaine.

Algorithme 1 : Algorithme gros grain de contrôle de système robotique modulaire

```
1  Se mettre en mouvement
2  répéter
3      Déplacement sans rencontre d'obstacle
4      si message d'obstacle alors
5          Arrêt pour évaluation
6          si obstacle répertorié alors
7              Identification de l'obstacle
8              suivant Recherche configuration faire
9                  cas où Configuration inexistante
10                     Arrêt du système
11                 cas où Configuration prévue et nouvelle
12                     Formation de la nouvelle configuration
13                 cas où Configuration prévue et actuelle
14                     Garder la configuration actuelle
15
16             Se mettre en mouvement pour franchir l'obstacle
17             Franchir l'obstacle
18         sinon
19             Arrêt du système
20 jusqu'à Niveau d'énergie nul
```

L'essentiel du contrôle exercé concerne les obstacles répertoriés, les configurations prévues et les messages échangés. Notez qu'à ce niveau d'abstraction, aucun échange explicite de messages, autre que l'annonce d'un obstacle, n'est observé entre

les modules robotiques. Ceci pourrait intervenir à tous les niveaux de l'algorithme. Il serait par exemple intéressant de faire une spécification étendue de la formation d'une nouvelle configuration, prenant en compte l'échange explicite de messages entre les modules.

4.2. *Modèle des modules robotiques*

La figure 5 présente un *Réseau Symétrique* modélisant les modules robotiques. Les bases de connaissances sur l'environnement sont représentées par les places **Obstacles_Connus**, **Obstacles_Inconnus**, **Config_Prevues** et **Pas_de_Config**. La place **Obstacles_surgissants** est une abstraction du canal de communication délivrant les messages. À l'assemblage du modèle final (avec le canal de communication), cette place sera remplacée par le modèle du canal. La transition **t1** sera synchronisée avec le canal pour déclencher la réception des messages.

L'assemblage est automatiquement réalisé par l'outil PetriScript (Hamez *et al.*, 2005). Avec cet outil, nous pouvons manipuler via des scripts les *Réseaux Symétriques* au moyen d'opérations de création, modification, suppression et fusion de nœuds et de leurs attributs. PetriScript fait partie de la suite CPN-AMI (LIP6-MoVe, 2007) utilisée pour accomplir la modélisation et l'analyse de ce cas d'étude.

Ce modèle simple fait appel à des domaines de données discrets pour coder les bases de connaissance et l'identification des modules. Chaque module exécute l'algorithme 1 défini plus haut. Des instants de synchronisation de tous les modules sont requis, par exemple pour la remise en marche après formation d'une configuration.

Sur la figure 5, le cadre n° 1 correspond à l'identification de l'obstacle exprimée par la ligne 7 de l'algorithme 1. Le cadre n° 2 correspond à la recherche de la configuration appropriée exprimée par les lignes 8 à 14. Enfin, le cadre n° 3 correspond à la remise en mouvement du robot pour franchir l'obstacle, exprimée par les lignes 16 et 17.

Dans ce modèle, six modules, cinq types d'obstacle (y compris une abstraction des obstacles inconnus) et trois types de configuration sont représentés. L'absence de configuration est également explicitement représentée. Ceci correspond généralement au dimensionnement couramment évoqué dans la littérature. Ce modèle comporte 26 places, 18 transitions et 70 arcs.

Pour un obstacle reconnu et pour lequel une configuration existe, son graphe des marquages accessibles (GMA) comporte 536 nœuds et 1.308 arcs, ce qui est très modeste. Pour un obstacle reconnu et pour lequel il n'existe pas de configuration, le GMA comporte 306 nœuds et 715 arcs. Pour un obstacle non reconnu, 114 nœuds et 202 arcs.

Son étude nous montre qu'il y a toujours convergence et cohérence de décision de l'ensemble des modules à l'annonce d'un obstacle : les configurations prévues sont bien sélectionnées et les cas d'arrêt pour intervention humaine respectés. Par

Figure 5. *Modèle de base des robots*

exemple, soient n le nombre de modules et la configuration actuelle celle du *serpent*. La négation de la formule CTL suivante :
$implies(Mesg\_File\_Depart = \langle .bois. \rangle, AF(Config\_Actuelle = \langle .araignee. \rangle)$
$and\ AF(card(Deplacement\_Obstacle\_Free) = n))$
rend zéro chemin, qui est la réponse attendue. Cette formule spécifie que, étant donné un marquage initial annonçant l'obstacle *morceau de bois*, la nouvelle configuration atteinte est celle de l'*araignée*, le robot a franchi l'obstacle et est en marche normale. Le cardinal du marquage de la place **Deplacement_Obstacle_Free** est donc égal au nombre de modules du robot. Ce comportement correspond aux lignes 6 à 8, 11 à 12, 16 et 17 de l'algorithme 1. Et ceci, finalement sur tous les chemins à partir de ce marquage initial.

L'ensemble du système est généralement à l'arrêt dès la réception d'une annonce d'obstacle jusqu'à la formation de la configuration appropriée ou au passage en attente d'intervention humaine. Par exemple, la négation de la formule CTL suivante (n étant le nombre de modules) : $implies(Mesg\_File\_Depart = \langle .obs\_inconnu. \rangle, AF(card(Arret\_Blocage) = n))$ rend zéro chemin, qui est la réponse attendue. Cette formule spécifie que, étant donné un marquage initial annonçant un obstacle inconnu (i.e. non répertorié, lignes 18 et 19 de l'algorithme 1), le cardinal

du marquage de la place **Arret_Blocage** est égal au nombre de modules du robot. Et ceci, également finalement sur tous les chemins à partir de ce marquage initial.

4.3. *Raffinement*

Nous avons ensuite considéré le SuperBot de (Shen *et al.*, 2006) et intégré ses six modes de locomotion, comme illustré et indiqué par le cadre n° 4 sur la figure 6. Six critères définissent les conditions de fonctionnement de chaque mode :

Pente Le degré d'inclinaison du terrain sur lequel évolue le robot. Ce critère est défini par un intervalle.

Obstacle Absence ou présence d'obstacle. La taille de l'obstacle rencontré est en rapport avec celle du robot. En général elle ne doit pas dépasser celle du robot.

Vitesse La vitesse appropriée au parcours d'un terrain.

Virage La capacité de tourner ou non.

Energie La consommation d'énergie nécessaire pour naviguer dans un mode donné.

Recouvrement La capacité de recouvrer d'une défaillance, typiquement d'une chute.

Figure 6. *Extrait du modèle du Superbot*

Ce cas d'étude est très intéressant. Nous avons noté que la description des six modes de locomotion n'est pas accompagnée par la définition du sous-ensemble discriminant de critères qui permet de déclencher le changement de configuration. Par exemple, le mode *6M Loop* permet de rouler vite mais ne recouvre pas d'une chute.

Cependant le *6M4C T-Wheel* ajoute des modules supplémentaires pour disposer de la capacité de recouvrement. Le *2M4C Loop* fournit la même capacité par un agencement particulier des modules.

La pente et l'obstacle pour ces trois modes ont les mêmes valeurs. Ces informations, qui nous intéressent pour déterminer un changement de configuration, ne sont manifestement pas suffisants. Nous avons donc défini une variable supplémentaire pour indiquer si le robot a chuté. Cette donnée est le pendant de celle de *recouvrement*. Il permet, par exemple dans les mêmes conditions, de changer d'un mode ne disposant pas de capacité de recouvrement comme le *6M Loop* vers un mode qui en est doté, comme le *2M4C Loop*.

En matière d'énergie, il faut connaître la capacité actuelle du robot pour déterminer si celle de la configuration choisie est réalisable. Nous avons par ailleurs considéré que la vitesse est une recommandation émise pour une nouvelle configuration atteinte. Elle n'intervient donc pas directement dans les critères déterminant un changement de mode. Le virage n'intervient pas non plus comme critère déterminant pour le changement de mode. Cependant, il pourrait également avoir un critère pendant comme pour le recouvrement, pour indiquer la nécessité d'effectuer un virage. Ceci pourrait déclencher un changement de mode si l'actuel ne convenait pas. Sa prise en compte figure dans les pistes pour un raffinement ultérieur.

Un message indique l'obstacle, la pente actuelle et si le robot a chuté. Le changement de mode dans le modèle est géré par six transitions, une pour chaque mode. Une garde (expression booléenne) établie pour chaque transition scrute les critères déterminant le changement de mode supporté par la transition. Par exemple, sur la transition gérant le mode *2M4C Loop*, nous avons : $[(new\_config <> current\_config\ and\ slo >= 20\ and\ slo <= 70\ and\ current\_ener >= new\_ener\ and\ ch = 1\ and\ rec = oui]$

Cette garde stipule que :

1) la configuration actuelle doit être différente de la nouvelle prévue,

2) la pente annoncée doit être comprise entre 20 et 70,

3) la capacité actuelle d'énergie doit être au moins égale à celle exigée dans la nouvelle configuration,

4) le robot doit avoir subi une chute,

5) un mode avec capacité de recouvrement est exigé en cas de chute,

6) *règle implicite* : la taille de l'obstacle doit être celle de la configuration prévue,

avant de déclencher le changement vers le mode *2M4C Loop*. Si la configuration actuelle est gardée, cela signifie qu'après traitement de l'information transmise par le message, aucun changement ne s'avère nécessaire. Les six transitions seront toutes invalidées par leurs gardes et seule une autre transition dédiée à ce cas est alors franchissable.

4.4. *Statistiques et analyse*

Le réseau de Petri assemblé modélisant le SuperBot comporte 25 places, 23 transitions et 112 arcs. Le marquage initial de base comporte toujours 7 modules, les six configurations prévues, la configuration initiale et un message dans le canal. Ce message sert à initier la détection d'un obstacle dans le système.

Figure 7. *Évolution de l'espace d'états (GMA) en fonction du nombre de modules robotiques.*

La Figure 7 présente l'évolution des espaces d'états des modèles pour un nombre de modules compris entre 2 et 7 à partir du traitement du message initial $< pente = 45,\ obstacle = non,\ chute = oui >$ avec exécution complète de l'algorithme (i.e., le système détermine la nécessité d'un changement de mode). Avec 5 modules, la taille de l'espace d'états reste de taille raisonnable, même si son évolution suit une courbe exponentielle.

Nos outils de calcul du graphe des marquages symboliques nous ont déjà permis d'étudier des espaces symboliques correspondant à plus de 10^{35} états concrets pour des problèmes similaires, sans saturer la mémoire centrale. Les GMA de cette taille correspondent à des configurations de quelques dizaines de modules. Ce sont des configurations réalistes de SuperBot si l'on en croit la littérature. Par exemple, (Kurokawa *et al.*, 2002) évoque 10 modules (et 20 dans le futur).

L'analyse des espaces d'états révèle que les modes *6M Loop* (roulade standard), *2M4C Loop* (roulade avec agencement particulier) et *6M4C T-Wheel* (roulade avec béquilles) peuvent être utilisés. La configuration actuelle du message initial est *6M4C T-Wheel*. Cette situation révèle un indéterminisme dans le choix du mode, que nous qualifierons de *flexibilité*.

Cette flexibilité est un avantage dans la mesure où le robot pourra effectuer un autre choix en cas de problème mécanique pour la formation d'un mode. Elle présente cependant l'inconvénient de ne pas forcément faire le choix du mode optimal. Par exemple, l'utilisation du mode *6M4C T-Wheel* sur un terrain dont la pente ne présente pas de difficulté particulière, le parcours pouvant également très bien être effectué avec un mode peut-être plus économe en énergie. Et inversement, le choix d'un mode économe en énergie au détriment de son adaptation à la pente du terrain.

L'indéterminisme n'est pas systématique, car avec le message $pente = 45,\ obstacle = non,\ chute = oui$, seul le mode *2M4C Loop* est adapté.

Une préoccupation récurrente des concepteurs de robots modulaires et auto-reconfigurables est ici mise en exergue. L'optimisation de certains critères se fait en général au détriment d'autres. Le mode optimal parfait est alors difficilement atteignable. Nous pensons que le compromis obtenu en faveur de la *flexibilité* est acceptable dans la mesure où le robot évolue dans un environnement imprévisible.

Cependant, cette flexibilité introduit un autre problème dans ce modèle : les modules peuvent ne pas faire le même choix de mode. D'où interblocage, au moment de changer de configuration. Dans le cas spécifique de ce modèle, une définition plus fine des critères accompagnée d'une heuristique est nécessaire pour assurer le contrôle réparti du changement de mode. A cause de l'impossibilité de consensus, un algorithme déterministe et imposé à tous les modules devra nécessairement prendre le relai sauf si nous comptons arrêter le système.

La définition des intervalles de pente pour chaque mode pourrait par exemple être plus restreinte. Pour ces intervalles, une heuristique traitant du changement de mode aux limites en fonction de l'énergie et de la capacité de recouvrement pourrait être mise en œuvre. Surtout, une deuxième heuristique permettant d'assumer le choix de la majorité en cas de choix multiple devra être implémentée. Il sera peut-être nécessaire d'effectuer plusieurs tours pour le choix final. Un module déterminé pourrait en cas de non convergence décider du choix final après un nombre déterminé de tours.

4.5. *Pistes pour d'autres raffinements*

L'exemple présenté ci-dessus montre que d'autres informations utiles pourraient être encore tirées de l'analyse du modèle de robot basée sur les Réseaux Symétriques. La spécification et l'analyse gros grain pourraient donc être encore raffinées pour, par exemple, approfondir la spécification sur les points suivants :

Mise à l'arrêt Lors de la réception d'un message annonçant un obstacle, un protocole de mise à l'arrêt peut être mis en œuvre. Par exemple, un module constatant l'arrêt de ses voisins s'arrête immédiatement aussi, quelle que soit l'action en cours.

Formation d'une configuration Les modules se synchronisent avant de démarrer la formation d'une nouvelle configuration ou garder l'actuelle. Cette synchronisation exige un point de rendez-vous, en général modélisé par une transition de synchronisation. On peut décider qu'elle donne lieu à un échange de messages représenté dans le modèle. Ceci implique un approfondissement de la spécification du canal, de la réception et du traitement de messages, ainsi que de la gestion de l'interblocage lors de la prise de décision.

De plus, à chaque module peut être affecté un emplacement physique précis de la nouvelle configuration. Ceci pourrait être fait statiquement avant la mise

en marche initiale du robot ou dynamiquement à la formation de la nouvelle configuration.

En cas de problème avec un ou plusieurs modules, il faut déterminer si le nombre restant est suffisant pour la configuration désirée et décider des emplacements facultatifs. Ensuite, il faudra réaffecter les emplacements physiques vitaux aux modules restants, si nécessaire. Sinon, le système s'arrête.

Séparation en cas de problème Lorsqu'il apparaît qu'un ou plusieurs modules sont inertes, ne répondent à aucune sollicitation ou ne se synchronisent pas, une séparation est décidée par un leader choisi dans une liste préétablie de leaders. Pour traiter ce problème et toutes les difficultés qu'il soulève, une heuristique sophistiquée doit être mise en œuvre. Elle peut être basée sur la réception de message de la part du module défectueux ou d'un voisin l'ayant constaté, un comptage des présences et un temporisateur d'attente discret, etc.

Enrichir les critères Comme indiqué dans la section 4.3, les critères déterminant un changement de mode pourraient être étendus. Comme par exemple, la nécessité d'effectuer un virage pour contourner un obstacle. Dans le cas où le mode actuel n'offre pas cette capacité, un changement devra être effectué.

5. Similitudes avec les Systèmes de Transport Intelligents

Les robots autonomes sont en principe destinés à évoluer dans un environnement imprévisible. Cet environnement comporte des terrains inconnus et d'autres robots, des animaux et d'autres objets avec lesquels interagir. L'intervention humaine directe dans le contrôle du robot est rare et mineure.

Pour connaître son environnement, le robot doit récupérer et traiter l'information de ses capteurs visuel, sonore, télémétrique, infrarouge, thermique etc. Cette information continuellement mise à jour permet de construire une représentation abstraite du contexte. Les décisions prises par le robot sont ainsi basées sur cette représentation. De ce point de vue, la modélisation des robots autonomes doit prendre en compte des préoccupations contraintes par le temps et l'espace.

L'approche de modélisation et d'analyse développée pour les robots autonomes est en droite ligne avec une approche similaire que nous avons déjà également appliquée aux Systèmes de Transport Intelligents (ITS) (Bonnefoi *et al.*, 2006; Bonnefoi *et al.*, 2007).

Les ITS (Bishop, 2005; Blosseville, 2005) pour la route et l'autoroute, sont principalement constitués de deux types d'acteurs : les véhicules et l'infrastructure. Chaque type d'acteur est composé d'équipements de perception, de communication et de décision. L'équipement de décision côté infrastructure est baptisé *Road-Side Unit* (RSU). Celui côté véhicules est appelé *On-Board Unit* (OBU). Ce sont les équipements de contrôle dans lesquels l'intelligence est embarquée.

Les ITS partagent avec les robots autonomes les caractéristiques suivantes : l'autonomie de la prise de décision, la perception de l'environnement, la symétrie de comportement, l'interaction avec d'autres objets et les contraintes spatio-temporelles.

Cependant, dans le cas des ITS, le schéma d'évolution est beaucoup plus stable que pour les robots. En effet, l'environnement des ITS (routes, autoroutes) est beaucoup moins imprévisible. De plus, il offre un cadre pour le développement et la variation de scénarios sur un même cas d'étude (Bonnefoi *et al.*, 2006). Enfin, il n'y a qu'un seul niveau de coordination inter-véhicule ou véhicule-infrastructure, alors que chez les robots autonomes nous devons également gérer la coordination inter-module pour la réalisation de configurations physiques [1].

C'est cette dernière qui a fait l'objet de la présente étude, en appliquant différemment les mêmes techniques qui ont guidé l'étude que nous avons menée sur les ITS. Cette fois-ci à l'échelle des robots modulaires.

La symétrie de comportement est également une caractéristique observable chez les véhicules. Elle exhibe beaucoup de coopération entre les véhicules. De ce fait, elle donne naturellement lieu à des études sur la conduite coopérative décentralisée, certaines s'inspirant des concepts du multi-agent (S. Halle *et al.*, 2004). Une retombée attendue est la réduction des coûts de conception en embarquant des algorithmes de contrôle standards chez plusieurs constructeurs.

6. Conclusion

Dans cette étude, nous avons pris le parti d'examiner l'apport des méthodes de spécifications formelles pour la modélisation et l'analyse formelles de systèmes robotiques modulaires. Cette approche fait principalement cas des techniques sous-jacentes de *model checking* qui tirent avantage de la symétrie de comportement des entités modélisées dans leurs algorithmes de contrôle. Le formalisme utilisé ici, les Réseaux Symétriques, en est un support représentatif.

L'exemple proposé décrit un algorithme de contrôle simple, que nous avons ensuite raffiné en introduisant les principes de locomotion du robot *SuperBot*. Ce cas d'étude nous a permis d'observer une maîtrise effective de la réduction de l'espace d'états et d'observer des caractéristiques intéressantes de comportement. Nous avons également pu relever et intégrer l'information nécessaire à l'évolution entre les différents modes de locomotion du *SuperBot*, information manquante dans la présentation initiale des auteurs de ce robot.

La démarche présentée dans ce papier peut supporter beaucoup de cycles de raffinement, dans la mesure où le problème de l'explosion combinatoire de l'espace d'états reste sous contrôle. Les récentes techniques de *model checking* mettant en œuvre des

1. Même si les véhicules peuvent évoluer en formations (*platoons*), celles-ci sont plus faiblement couplées.

réductions efficaces et tirant parti des architectures massivement parallèles nous permettent de modéliser et d'analyser des systèmes de complexité croissante.

Une telle démarche est cependant limitée pour les heuristiques d'auto-configuration, que nous n'avons pas intégrées dans notre exemple. Nous pensons que les concepts structurants d'intelligence artificielle constitueraient un apport important pour leur réalisation. De telles heuristiques peuvent intégrer la reconnaissance et la localisation autonomes d'objets (Modayil *et al.*, 2006).

Enfin, nous avons considéré l'application cette démarche de spécification et d'analyse aux Systèmes de Transport Intelligents (ITS), mise en œuvre dans de précédentes études. L'analogie se rapporte à l'autonomie de ces systèmes qui évoluent dans un environnement plus ou moins imprévisible dans lequel perception, interaction, prise de décisions et coordination sont contraintes par l'espace et le temps.

7. Bibliographie

Bérard B., Bidoit M., Finkel A., Laroussinie F., Petit A., Petrucci L., Schnoebelen Ph., *Systems and Software Verification. Model-Checking Techniques and Tools*, Springer, 2001.

Bishop R., « Intelligent Vehicle R&D: a review and contrast of programs worldwide and emerging trends. », *Annals of Telecommunications - Intelligent Transportation Systems*, vol. 60, n° 3-4, p. 228-263, March-April, 2005.

Blosseville J.-M., « Driving assistance systems and road safety: State-of-the-art and outlook », *Annals of Telecommunications - Intelligent Transportation Systems*, vol. 60, n° 3-4, p. 281-298, March-April, 2005.

Bonnefoi F., Hillah L., Kordon F., Frémont G., « An approach to model variations of a scenario: Application to Intelligent Transport Systems », *Workshop on Modelling of Objects, Components, and Agents (MOCA'06)*, Turku, Finland, June, 2006.

Bonnefoi F., Hillah L., Kordon F., Renault X., « Design, modeling and analysis of ITS using UML and Petri Nets », *IEEE 10th International Conference on Intelligent Transportation Systems*, Seattle WA, USA, September, 2007.

Butler Z., Kotay K., Rus D., Tomita K., « Generic decentralized control for a class of self-reconfigurable robots », *IEEE International Conference on Robotics and Automation, ICRA 2002*, vol. 1, p. 809-816, May, 2002.

Chiola G., Dutheillet C., Franceschinis G., Haddad S., « On well-formed coloured nets and their symbolic reachability graph », *in* K. Jensen, G. Rozenberg (eds), *Procedings of the 11th International Conference on Application and Theory of Petri Nets (ICATPN'90). Reprinted in High-Level Petri Nets, Theory and Application.*, Springer, 1991.

Christensen D. J., Stoy K., « Selecting a meta-module to shape-change the ATRON self-reconfigurable robot », *IEEE International Conference on Robotics and Automation, ICRA 2006*, p. 2532-2538, 2006.

Garcia E., Jimenez M. A., De Santos P. G., Armada M., « The evolution of robotics research », *Robotics & Automation Magazine, IEEE*, vol. 14, n° 1, p. 90-103, 2007.

Girault C., Valk R., *Petri Nets for Systems Engineering*, Springer Verlag - ISBN: 3-540-41217-4, 2003.

Gogen J., Luqi, « Formal Methods: Promises and Problems », *IEEE Software*, vol. 14, n° 1, p. 75-85, 1997.

Hamez A., Kordon F., Thierry-Mieg Y., Legond-Aubry F., « dmcG: a distributed symbolic model checker based on GreatSPN », *Application and Theory of Petri Nets and Other Models of Concurrency*, vol. 4546 of *Lecture Notes in Computer Science*, Springer-Verlag, 2007.

Hamez A., Renault X., *PetriScript Reference Manual*, LIP6, `http://www-src.lip6.fr/logiciels/mars/CPNAMI/MANUAL_SERV`. 2005.

Hugues J., Thierry-Mieg Y., Kordon F., Pautet L., Baarir S., Vergnaud T., « On the Formal Verification of Middleware Behavioral Properties », *9th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'04)*, Elsevier, p. 139-157, September, 2004.

Jensen K., *Coloured Petri Nets: Basic concepts, analysis methods and practical use. Volume 1: basic concepts*, Monographs in Theoretical Computer Science, Springer, 1992.

Kordon F., « Mastering Complexity in Formal Analysis of Complex Systems: Some Issues and Strategies Applied to Intelligent Transport Systems », *International Symposium on Object-oriented Real-time distributed Computing (ISORC'07)*, IEEE Computer Society, Santorini, Greece, p. 420-427, May, 2007.

Kordon F., Linard A., Paviot-Adet E., « Optimized Colored Nets Unfolding », *International Conference on Formal Methods for Networked and Distributed Systems (FORTE'06)*, vol. 4229 of *Lecture Notes in Computer Science*, Springer, Paris, France, p. 339-355, September, 2006.

Kotay K., Rus D., « Efficient Locomotion for a Self-Reconfiguring Robot », *IEEE International Conference on Robotics and Automation, ICRA 2005*, p. 2963-2969, 2005.

Kurokawa H., Kamimura A., Yoshida E., Tomita K., Murata S., Kokaji S., « Self-Reconfigurable Modular Robot (M-TRAN) and its Motion Design », *7th International Conference on Control, Automation, Robotics and Vision (ICARCV'02)*, Singapore, December, 2002.

LIP6-MoVe, *The CPN-AMI Home page.*, `http://www.lip6.fr/cpn-ami`. 2007.

Modayil J., Kuipers B., « Autonomous shape model learning for object localization and recognition », *IEEE International Conference on Robotics and Automation, ICRA 2006*, p. 2991-2996, 2006.

Rus D., Butler Z., Kotay K., Vona M., « Self-reconfiguring robots », *Communications of the ACM*, vol. 45, n° 3, p. 39-45, 2002.

S. Halle J. L., Chaib-Draa B., « A decentralized approach to collaborative driving coordination », *IEEE 7th Conference on Intelligent Transportation System*, p. 453-458, 2004.

Shen W.-M., Krivokon M., Chiu H., Everist J., Rubenstein M., Venkatesh J., « Multimode locomotion via SuperBot robots », *IEEE International Conference on Robotics and Automation, ICRA 2006*, p. 2552-2557, 2006.

Thierry-Mieg Y., Baarir S., Duret-Lutz A., Kordon F., « Nouvelles techniques de model checking pour la vérification de systèmes complexes », *Génie Logiciel*, n° 69, p. 17-23, June, 2004.

Thierry-Mieg Y., Dutheillet C., Mounier I., « Automatic Symmetry Detection in Well-Formed Nets », *Proc. of ICATPN 2003*, vol. 2679 of *Lecture Notes in Computer Science*, Springer, p. 82-101, June, 2003.

ANNEXE POUR LE SERVICE FABRICATION
A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :

JESA. Volume 8 – n°2/2008

2. AUTEURS :

Lom Messan Hillah[*] *— Fabrice Kordon*[*] *— Laure Petrucci*[**]

3. TITRE DE L'ARTICLE :

Application des méthodes formelles à la robotique modulaire

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :

Méthodes formelles pour la robotique

5. DATE DE CETTE VERSION :

5 juin 2011

6. COORDONNÉES DES AUTEURS :

– adresse postale :
[*] Université P. & M. Curie - Paris 6, CNRS UMR 7606 - LIP6/MoVe
4, place Jussieu, F-75252 Paris CEDEX 05, France
lom-messan.hillah@lip6.fr, fabrice.kordon@lip6.fr

[**] LIPN, CNRS UMR 7030, Université Paris XIII
99, avenue Jean-Baptiste Clément, F-93430 Villetaneuse, France
Laure.Petrucci@lipn.univ-paris13.fr
– téléphone : 01 44 27 88 20
– télécopie : 01 44 27 74 95
– e-mail : fabrice.kordon@lip6.fr

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :

LaTeX, avec le fichier de style `article-hermes.cls`,
version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :

Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
`http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr