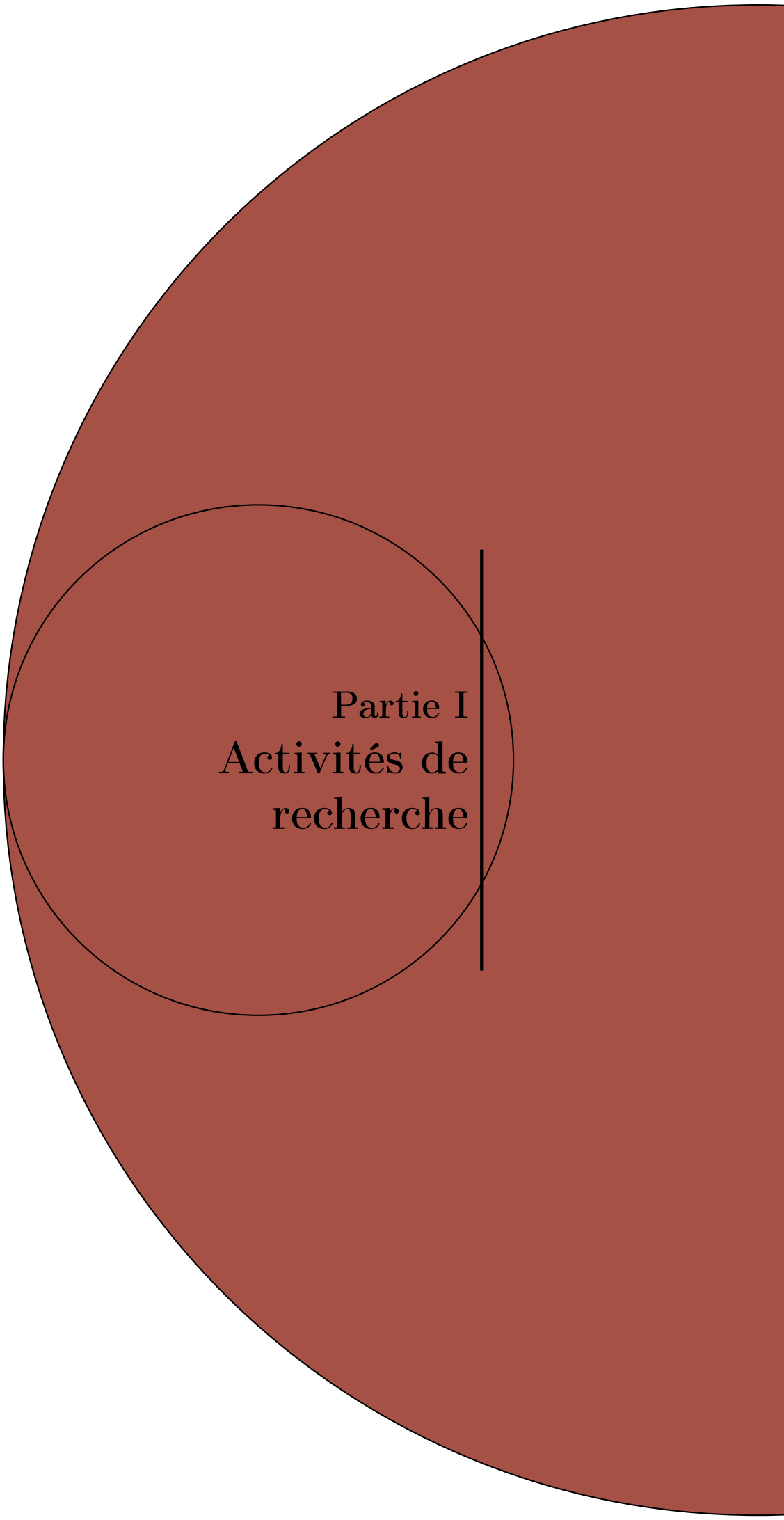
# Rapport d'activité 2003–2006

LIPN, CNRS UMR 7030

22 septembre 2007



# Partie I
# Activités de recherche

# Chapitre 1
# LCR (Logique, Calcul et Raisonnement)

## 1.1 Composante Spécification et Vérification

*Participants : C. Choppy, K. Klai, M. Mayero, L. Petrucci*

### 1.1.1 Thèmes de recherche

Les systèmes développés de nos jours sont de plus en plus complexes et leur fonctionnement peut avoir des conséquences importantes sur leur environnement, voire irréversibles : systèmes avioniques, médicaux ...

Il est par conséquent indispensable d'obtenir des systèmes sûrs, dont le fonctionnement a pu être vérifié avant leur mise en œuvre.

La spécification d'un système présente tout d'abord des avantages quant à la description et la compréhension du système considéré :

– L'écriture de la spécification permet au concepteur de mieux comprendre le fonctionnement attendu de son système ainsi que les interactions entre les différents

composants. L'écriture du modèle conduit à remettre en cause des choix conceptuels d'une part et à une meilleure compréhension du système développé d'autre part.
- Le modèle constitue une documentation précise lors de la maintenance.

De plus, les modèles formels permettent de prouver formellement (i.e. mathématiquement) le bon fonctionnement du système développé. On est alors complètement sûr que quelle que soit l'évolution du système, son comportement sera celui attendu.

Outre les avantages précédents, propres à la spécification, les méthodes formelles offrent des outils d'analyse du système modélisé :
- La simulation permet de se convaincre du bon fonctionnement du système, ou éventuellement de découvrir quelques erreurs. Leur correction à ce stade est peu coûteuse, comparée à celle d'un système déjà mis en œuvre (que ce soit avec des moyens logiciels ou matériels).
- La vérification exhaustive du comportement du système. On applique dans un premier temps des techniques de simulation pour effectuer un débogage grossier, puis le modèle est affiné lors de la vérification des propriétés souhaitées.

Les activités de la thématique *spécification et vérification* s'articulent autour de plusieurs axes de recherche :
- La méthodologie de développement d'une spécification : quel modèle/langage choisir, à quel niveau d'abstraction se placer, comment considérer le problème, comment commencer à écrire et développer la spécification.
- Les systèmes à modéliser étant en général de taille assez conséquente, ils sont spécifiés de manière structurée, par exemple avec une architecture modulaire. L'analyse modulaire vise à tirer parti de cette structuration pour repousser les limites des techniques d'analyse de propriétés.
- Lorsqu'une spécification a été validée à un niveau abstrait, des détails peuvent être introduits par un processus de raffinement. La vérification du modèle raffiné doit alors s'appuyer sur celle du modèle abstrait.
- Enfin, l'utilisation de prouveurs de théorèmes permet de valider les étapes de raffinement de modèles. L'utilisation de ces prouveurs pour des problèmes numériques devra à terme permettre d'aborder des propriétés de systèmes en temps continu.

### Méthodologie de spécification

Si l'utilité et la nécessité des spécifications formelles est bien démontrée dès qu'il s'agit de développer ou de manipuler des logiciels complexes, écrire les spécifications d'un système demeure une tâche ardue et délicate. Nous explorons ici comment rendre ce travail plus aisé en fournissant des lignes guide, soit pour écrire directement la spécification, soit en utilisant un langage ou un formalisme intermédiaire plus accessible, ou encore en s'appuyant sur des schémas identifiés (pour cette dernière approche, le lien avec les étapes ultérieures de développement est également étudié).

Ce travail de méthodologie a tout d'abord eu pour but de fournir des spécifications écrites dans le langage de spécification algébrique CASL, ou dans une extension CASL-LTL [Ra-1] pour les systèmes dynamiques. Mais les idées alors développées sont assez générales pour trouver leur application dans l'écriture de spécifications dans d'autres langages (Z, LOTOS, réseaux de Petri, RAISE).

Dans [RI-5] des lignes guide sont proposées pour l'écriture de spécifications de systèmes dynamiques (simples ou structurés) ainsi que de types de données. Différentes caractéristiques constitutives d'un système sont mises en évidence et les propriétés à rechercher sont détaillées.

Le lien de cette méthode avec les cas d'utilisation (qui sont munis d'une description structurée) est décrit dans [CI-5], ce qui permet de fournir une graphique familière aux utilisateurs de la notation UML.

Il existe des « patterns » pour décrire les problèmes, ce sont les « Problem Frames » de Michael Jackson. Afin de bénéficier des concepts structurants des problem frames (et aussi de leur graphique) nous avons travaillé pour les intégrer dans une démarche de spécification, ce qui nous a conduit à les équiper d'une description plus détaillée proposée en UML à laquelle nous savons associer une spécification formelle [CO-2, CI-4, RI-2]. Ces travaux nous ont aussi conduit à proposer des schémas de problème plus complexes adaptées toutefois à des larges classes d'application [RI-2, CI-10].

Les « patterns » ont d'abord été proposés pour des étapes ultérieures du développement, aussi avons nous exploré comment faire le lien entre une démarche de spécification utilisant les problem frames et un développement à l'aide de styles d'architecture [CI-1], et nous avons proposé dans [CO-4] une approche pour les systèmes d'information, puis dans [RI-3, CI-11] une approche plus générale, dans les deux cas produisant une architecture par composants.

Nous souhaitions mettre en œuvre pour d'autres formalismes les idées proposées dans [RI-5] pour l'écriture de spécification, et nous avons donc exploré dans [CO-3] l'écriture de réseaux de Petri. Ces travaux ont été poursuivis dans le cadre d'un stage de Master Recherche en 2006.

Dans [CL-2, CL-1, TU-1], nous présentons des idées générales sur les critères qui interviennent dans l'écriture des spécifications, et sur certains des travaux présentés ci-dessus. Il est important de noter que chaque méthode de spécification est plus ou moins adaptée à un problème spécifique. C'est pourquoi, il faut tirer parti au mieux des différentes techniques existantes [RE-3].

L'utilisation conjointe de différents outils est nécessaire. Pour atteindre un tel objectif, nous sommes impliqués dans le processus de normalisation des réseaux de Petri [Ra-2, RE-2, CI-8]. Les réseaux de Petri sont normalisés ISO/IEC 15909-1 (première partie d'une norme en 3 volets). Les aspects pratiques concernant un format d'échange (ISO/IEC 15909-2) sont en cours. Ils impliquent la définition d'un métamodèle. Ceci a permis d'automatiser la lecture/écriture d'un fichier écrit en PNML (Petri Net Markup Language) en utilisant une approche MDA (*Model-Driven Architecture*). Le langage PNML a également évolué au cours de ces travaux. De plus, nous avons soumis un addendum à la partie 1 de la norme pour définir les réseaux symétriques, qui sont des réseaux de haut niveau simples, et servent de base aux travaux sur la partie 2. Enfin, les travaux de la partie 3, portant sur les extensions de réseaux de Petri, viennent de démarrer. Nous sommes éditeurs de ce volet de la norme.

### Vérification Modulaire

Nous nous intéressons ici tout particulièrement à des systèmes complexes, parallèles, asynchrones, modélisés à l'aide de réseaux de Petri.

La principale méthode de vérification de propriétés à partir d'un modèle spécifié par un réseau de Petri consiste en l'exploration du *graphe d'états accessibles*. Toutefois, pour des systèmes de taille importante, on se heurte très rapidement au « problème de l'explosion combinatoire de l'espace d'états ». Le graphe d'états est alors trop gros pour être stocké en mémoire.

De nombreuses techniques ont été proposées pour pallier ce problème (ordres partiels, bit-state, ...). Nos travaux se concentrent sur l'analyse modulaire qui tire parti de la structure modulaire du modèle à analyser. L'activité des modules est explorée de manière indépendante plutôt qu'entrelacée. La technique de graphes d'états modulaires que nous développons consiste alors à construire un graphe d'états « local » par module plus

une structure appelée « graphe de synchronisation » décrivant les interactions entre les différents modules [IN-1].

Nous avons proposé, dans [CI-2], des algorithmes effectuant la construction du graphe d'états modulaire d'un réseau de Petri, ainsi que la vérification de propriétés (bornes, blocage, vivacité, ...). L'implémentation d'un prototype a permis de réaliser des expérimentations sur un problème classique d'automates industriels (les véhicules autoguidés). Les résultats ainsi obtenus ont montré des gains considérables — passage de 31.000.000 à 900 états. Toutefois, des études de cas ont été menées dans [RE-1], qui montrent que la technique est particulièrement adaptée à des systèmes composés de sous-systèmes autonomes faiblement couplés tels que ceux considérés en automatique. Dans le pire des cas, lorsque les sous-systèmes sont totalement synchronisés, on aboutit à un léger surcoût.

Nous avons adapté cette approche à la synthèse de réseaux de Petri, en collaboration avec Philiipe Darondeau de l'IRISA [Ra-6].

Suite aux résultats encourageants de la technique de graphes d'états modulaires, nous l'avons étendue à des modèles de réseaux de Petri comportant une modélisation du temps. Nous nous sommes intéressés à la fois aux réseaux temporisés, dans lesquels on considère une horloge globale et les jetons sont estampillés par la date à laquelle ils sont prêts, et aux réseaux temporels, où le temps est compté localement, un intervalle est associé à chaque transition et indique la période dans laquelle la transition doit être franchie, à partir du moment où toutes ses préconditions sont satisfaites.

Nous avons donné, dans [CO-7, RI-6], des algorithmes de construction de graphe d'états modulaires pour les réseaux temporisés. L'inconvénient majeur réside dans l'utilisation de l'horloge globale. En effet les états du graphe sont des couples (marquage, date). Lorsque le système peut évoluer indéfiniment, le graphe obtenu est alors infini.

Dans [CO-9], une technique similaire est présentée pour les réseaux temporels. Les états du graphe sont composés d'un marquage et d'une vecteur de valuations temporelles associées aux transitions. Le graphe obtenu est alors fini (si le réseau de Petri est borné).

Ces différents algorithmes ont été implémentés, et des expérimentations [RI-6] ont montré les apports de l'approche.

une autre approche est l'analyse compositionnelle, qui consiste à vérifer des propriétés d'un sous-système pour valider celles du système complet. Nous avons développé un tel algorithme pour le vérification de formules LTL\X [CI-12].

**Raffinement de spécifications de haut niveau**

De nos jours, les systèmes à modéliser et à analyser sont de taille de plus en plus grande. Pour s'affranchir des problèmes inhérents à l'explosion du nombre d'états, une spécification est souvent développée pas à pas (ce qui correspond à une démarche classique de développement de génie logiciel).

Tout d'abord, un modèle abstrait est défini. Une fois que ses propriétés sont vérifiées, une étape de raffinement, qui introduit un niveau de détail supplémentaire, peut être effectuée. Un tel ajout peut préciser la description de l'actuel fonctionnement d'une partie du système, ou introduire une partie supplémentaire. Le modèle ainsi raffiné est ensuite vérifié. Une nouvelle étape de raffinement peut alors être appliquée, et ce jusqu'à atteindre un niveau suffisant de description.

Trois types de raffinement ont été introduits par Lakos et Lewis, pour les réseaux de Petri de haut niveau : le raffinement de sous-réseau, le raffinement de nœud et le raffinement de type. Ces raffinements ont été utilisés dans le cadre d'une approche incrémentale à la spécification de systèmes complexes. Outre les avantages du raffinement

pour la spécification de systèmes, ils permettent de faciliter la vérification de propriétés. En effet, d'une part certaines propriétés sont préservées par le raffinement, et d'autre part, l'analyse du modèle abstrait permet de guider celle du modèle concret, économisant ainsi en temps de calcul et espace mémoire lors de la construction de graphes d'états.

Jusqu'à présent, le processus de raffinement de réseaux est complètement manuel. La vérification des contraintes peut être assez complexe. Par conséquent, nous souhaitons prouver automatiquement qu'un réseau est effectivement un raffinement d'un autre réseau (abstrait). Pour cela, nous utilisons des techniques de preuve de théorèmes, en particulier, l'assistant d'aide à la preuve Coq, développé à l'Inria. Nous proposons donc une formalisation en Coq des définitions de raffinements ainsi que des réseaux auxquels ils s'appliquent. Ceci a tout d'abord été développé pour les raffinements de sous-réseaux et de nœuds des réseaux places/transitions, et appliqué à quelques exemples. Ce travail est étendu, dans le cadre d'un stage de master, aux réseaux colorés.

Le raffinement de types pour les réseaux de Petri de haut niveau a été peu étudié. Pour développer cet aspect, nous nous sommes intéressés dans un premier temps à la vérification de systèmes paramétrés. Le bon fonctionnement de la famille de protocoles « stop-and-wait » a ainsi pu être vérifié [CO-8, RI-4], et ce, de manière paramétrée.

### Spécifications et preuves formelles

Dans le cadre du nouveau projet CerPAN [1] (Certification de Programmes d'Analyse Numérique, ANR décembre 2005), nous nous intéressons au développement et à la mise en application des méthodes permettant de démontrer formellement la correction de programmes issus du domaine de l'analyse numérique. Le projet concerne plus particulièrement des programmes apparaissant de manière récurrente dans la résolution de problèmes critiques. Beaucoup de programmes critiques sont issus de ce domaine, mais les travaux traitant directement des applications des méthodes formelles aux programmes d'analyse numérique sont rares. La principale raison est l'utilisation intensive des nombres réels (à virgule flottante) dans les programmes numériques, alors que les méthodes formelles manipulent plutôt des nombres entiers, ou plus généralement des structures discrètes. Deux méthodes sont explorées dans ce projet, toutes deux étant vouées à être mises en pratique sur une étude de cas (gradient analytique).

1. L'une consiste à utiliser des outils de vérification de programmes tels que Why et Caduceus (dédié aux programmes C) développés au LRI. Il s'agit de travailler sur un programme déjà existant, qui va être annoté suivant les propriétés à préserver. Ces annotations engendrent alors des obligations de preuve pour divers systèmes de preuve formelle (Coq, PVS, Isabelle/HOL, HOL 4, HOL Light, Mizar,...). Ces outils de vérification de programmes ne traitaient pas les nombres flottants. Nous avons donc participé aux phases de décision concernant l'extension de ces outils aux nombres flottants sous forme d'un langage de spécification pour les propriétés relatives aux nombres à virgule flottante. Ces ajouts sont distribués avec le système.
2. L'autre méthode consiste à effectuer la formalisation et la preuve du problème correspondant au programme ou à l'algorithme étudié, puis à utiliser le mécanisme d'extraction du système Coq afin d'obtenir automatiquement un programme prouvé correct. Ce principe d'extraction ne traite pas non plus la famille des nombres réels. Nous avons commencé par décider d'extraire vers des réels exacts et non vers des nombres flottants et nous sommes actuellement en train de rechercher un modèle adéquat de réels exacts (réaliste d'un point de vue implantatoire, efficace, ...) pour étendre l'extraction de Coq.

[1] http://www-lipn.univ-paris13.fr/CerPAN/

Nous essayons de tenir compte des aspects d'efficacité qui peuvent jouer un rôle déterminant dans le domaine de l'analyse numérique en particulier. Les méthodes développées dans ce projet auront un large champ d'application dans les environnements destinés à produire du code numérique garanti correct par rapport à des spécifications formelles et devra à terme permettre d'aborder des propriétés de systèmes en temps continu lors d'étapes de raffinement.

Nous avons développé une procédure de décision dans Coq pour les corps algébriquement clos [CI-9]. Cette procédure de décision est basée sur la méthode d'élimination des quantificateurs et utilise l'interface [RI-1] que nous avons développée entre le prouveur Coq et l'outil de calcul formel Maple.

Nous avons effectué en Coq la preuve du $20^{\text{ème}}$ problème de Diophante (l'aire d'un triangle rectangle dont les côtés sont mesurés par des entiers peut-elle être un carré parfait ?). Ce problème fut résolu négativement par Fermat au moyen de la méthode de descente infinie. La méthode de descente infinie permit à Fermat de prouver le théorème pour $n = 4$. Nous avons formalisé cette preuve en Coq.

Nous proposons une méthodologie pour détecter efficacement les erreurs sur des composants matériel génériques (SoC), utilisant à la fois des techniques de model-checking, de test et des prouveurs de théorèmes.

### 1.1.2 Perspectives de recherche

#### Méthodologie de spécification

Les idées développées dans [RI-5] peuvent être reprises et adaptées en fonction des langages de spécification formelle cibles. Des travaux ont été entamés notamment en direction :

- du langage Raise/RSL, en collaboration avec Anne Haxthausen qui est experte de ce langage et qui, à l'occasion du développement d'études de cas, a mené une réflexion sur les aspects méthodologie (sans l'avoir formalisée en tant que telle).
- des réseaux de Petri, pour lesquels, à la suite de [CO-3], des travaux ont été poursuivis en collaboration avec Gianna Reggio, pour les réseaux places/transitions [Ra-5], pour les réseaux colorés [Ra-7], et seront étendus aux réseaux hiérarchiques, puis temporisés.

Ces travaux vont nous permettre d'identifier une partie "générique" indépendante du formalisme cible, et une partie spécifique induite par le choix du type de modèle.

#### Vérification modulaire et distribuée

- Vérification modulaire temporisée : les travaux menés jusqu'à présent permettent la construction d'un graphe d'états modulaire pour des modèles de réseaux de Petri avec contraintes de temps. Toutefois, l'intérêt de l'approche est de pouvoir vérifier les propriétés du réseau de Petri sans avoir à déplier le graphe modulaire.
- Etudes de cas : les techniques développées seront appliquées à des exemples complexes. Nous avons entamé cette démarche en utilisant le « fieldbus protocol ».
- Vérification distribuée : dans [CI-3], nous avons présenté une implémentation distribuée du calcul de graphe d'états. Nous avons également montré comment les propriétés simples d'un réseau de Petri peuvent être vérifiées. L'analyse de propriétés plus élaborées telles que la vivacité est plus complexe. De plus, nous souhaitons utiliser un cluster de machines mis en place récemment au LIPN pour pouvoir implémenter une plateforme distribuée et effectuer des tests. Cela nous permettra en

particulier d'évaluer la répartition des nœuds du graphe sur les différents processeurs. À terme, l'architecture logicielle devra également permettre la construction distribuée du graphe modulaire.
- Heuristiques pour la construction de graphes d'états : nous avons soumis un projet de « discovery grant » à l'ARC (Australian Research Council), HAVOCS (Heuristic Approaches to the Verification Of Complex concurrent Systems »). L'objectif est d'utiliser des approches heuristiques (telles que les algorithmes fourmis) pour guider la génération du graphe d'états. Cette politique de construction devra rechercher en priorité des chemins pouvant conduire à des états invalidant une propriété souhaitée. Ce travail sera effectué en collaboration avec Hanene Azzag de l'équipe A³.

**Raffinement**

- Les premières expériences effectuées avec le prouveur de théorèmes Coq pour montrer qu'un réseau de Petri places/transitions est un raffinement d'un autre ayant été probantes, nous souhaitons étendre ces travaux aux différents types de raffinements de réseaux de Petri colorés (stage de master recherche en cours). De plus, pour rendre ces travaux utilisables, il est nécessaire d'implémenter un outil faisant la traduction automatique d'un réseau de Petri en une spécification Coq.
- Parmi les raffinements proposés par Lakos et Lewis, on en distingue deux (raffinements de nœud et de sous-réseau) qui portent sur la structure du réseau de Petri, et un (raffinement de type) qui s'intéresse aux données manipulées dans un réseau de haut niveau. La définition du raffinement de type est pour l'instant très limitée. Nous envisageons donc de l'étendre en nous inspirant des techniques d'abstraction de données et des travaux existant sur l'indépendance de données.
- Ces travaux seront appliqués à des études de cas, mettant ainsi en évidence les problèmes spécifiques que l'on peut rencontrer lors de tests à grande échelle.

**Autres activités**

- Collaboration avec l'ONERA (coencadrement de thèse depuis février 2006) : modélisation de la reconnaissance de chroniques dans le cadre de simulations de systèmes distribués [CO-13, CO-12]
- Collaboration avec l'équipe A³ : modélisation et vérification du protocole de communication du systèmes multi-agents développé par Hager Karoui, à l'aide de réseaux de Petri de haut niveau.
- Standardisation : les travaux sur la norme ISO/IEC 15909 partie 2 concernant le format d'échange de réseaux de Petri seront poursuivis. Un des points cruciaux est la prise en compte d'extensions qui seront normalisées par la suite dans la partie 3 du standard, dont nous sommes porteurs.

### 1.1.3 Diffusion de la recherche

**Evaluation de la recherche**

**présidence de comités de programme**

- ACSD, *International Conference on Application of Concurrency to System Design*, juin 2006, Turku, Finlande ; L. Petrucci
- PNS, *International Workshop on Petri Nets Standards*, juin 2007, Siedlce, Pologne ; L. Petrucci

– PNTApp, *International Workshop on Petri Nets Tools and APPlications*, mars 2008, Marseille, France ; L. Petrucci

**comités de programme**

– aasac, *The 23 rd Annual ACM Symposium on Applied Computing (Requirements Engineering)*, mars 2008, Fortaleza, Brazil ; C. Choppy
– acsd, *International Conference on Application of Concurrency to System Design*, juin 2004, Hamilton, Canada ; juin 2005, St Malo, France ; juin 2006, Turku, Finlande ; juillet 2007, Bratislava, Slovaquie ; L. Petrucci
– aiccsa, *ACS/IEEE International Conference on Computer Systems and Applications*, mars 2006, Dubai, UAE ; C. Choppy ; avril 2008, Doha, Qatar ; C. Choppy, L. Petrucci
– amast, *International Conference on Algebraic Methodology and Software Technology*, juillet 2004, Stirling, UK ; juillet 2006, Kuressaare, Estonie ; C. Choppy
– atpn, *International Conference on Application and Theory of Petri Nets and Other Models of Concurrency*, juin 2006, Turku, Finlande ; juin 2007, Siedlce, Pologne ; L. Petrucci
– coopn, *Workshop on Coordination and Petri Nets (ATPN)*, juin 2005, Miami, USA ; L. Petrucci
– cpn, *Workshop on Practical use of Coloured Petri Nets*, octobre 2004, 2005, 2006 et 2007, Århus, Danemark ; L. Petrucci
– dcacsd, *Doctoral Consortium (ACSD)*, juin 2006, Turku, Finlande ; L. Petrucci
– iee, *Informatics Education Europe*, novembre 2007, Thessalonike, Grèce ; novembre 2006, Montpellier, France ; special issue in : italics, Innovation in Teaching And Learning in Information and Computer Sciences (2007) ; C. Choppy
– isps, *International Symposium on Programming and Systems*, mai 2005 et 2007, Alger, Algérie ; L. Petrucci
– iwaapf, *International Workshop on Applications and Advances in Problem Frames*, mai 2006, Shangai, Chine ; C. Choppy
– mw, *14th Monterey Workshop, Workshop on Innovations for Requirements Analysis : from stakeholders needs to formal designs* septembre 2007, Monterey, CA, USA ; C. Choppy
– rst, *International Conference on Reliable Software Technologies (ADA-Europe)*, juin 2007, Genève, Suisse ; juin 2008, Venise, Italie ; L. Petrucci
– pnml, *Petri Net Markup Language and standardisation*, juin 2004, Bologne, Italie ; mai 2005, Helsinki, Finlande ; juin 2006, Turku, Finlande ; L. Petrucci
– pns, *Workshop on Petri Nets Standards*, juin 2007, Siedlce, Pologne ; L. Petrucci
– PNTApp, *Workshop on Petri Nets Tools and APPlications*, mars 2008, Marseille, France ; L. Petrucci
– TeaConc, *Workshop on Teaching Concurrency (ATPN)*, juin 2006, Turku, Finlande ; juin 2007, Siedlce, Pologne ; L. Petrucci

**comités éditoriaux**

– dso, ieee Distributed Systems Online, section « Software Engineering », 2006, 2007 ; L. Petrucci
– ToPNoC, Transactions on Petri Nets and other models of Concurrency, journal publié dans la série lncs, 2007 ; L. Petrucci
– édition d'un numéro spécial de la revue Fundamenta Informaticae, 2007 ; L. Petrucci
– édition d'un numéro spécial de la revue TSI, 2008 ; L. Petrucci

**organismes de normalisation**

- ISO, groupe de travail SC7/WG19 ; L. Petrucci

**comités d'évaluation**

- DFG, *Deutsche Forschungsgemeinschaft*, 2006 ; C. Choppy
- ANR 2006 ; expertises : M. Mayero, L. Petrucci
- ANR 2007 ; expertises : M. Mayero
- CRSNGC, *Conseil de recherches en sciences naturelles et en génie du Canada*, 2004, 2005 et 2007 ; C. Choppy — 2006 ; L. Petrucci
- FQRNT, *Fonds Québécois de la Recherche sur la nature et les technologies*, 2004 ; L. Petrucci
- NOSR, *Netherlands Organisation for Scientific Research*, 2005 ; L. Petrucci

**autres**

- Jurys de thèse : L. Petrucci (13)
- Rapporteurs de thèse étrangère : C. Choppy (1 Québec), L. Petrucci (1 co-tutelle Allemagne, 3 Australie, 1 Cameroun)

**Organisation de la recherche**

**organisation d'événements scientifiques et conférences**

- EUROTICS, *European Computer Science Summit*, octobre 2006, Zurich, Suisse ; C. Choppy
- FORTE, *International Conference on Formal Methods for Networked and Distributed Systems*, septembre 2006, Paris, France ; C. Choppy
- MOMPES, *MOdel-based Methodologies for Pervasive and Embedded Software (ACSD)*, juin 2005, Rennes, France ; L. Petrucci

**Contrats**

| Type | Coordinateur | Resp. scient. | Nom | Financeur | Période | Durée (mois) | Budget global (Keuros) | Part LIPN (Keuros) |
|---|---|---|---|---|---|---|---|---|
| ANR | • | M. Mayero | CerPAN | MENESR | 12/2005-12/2008 | 36 | 150 | 40 |
| Linkage Int. | | L. Petrucci | | ARC | 01/2004-12/2004 | 12 | 6 | 3 |
| PAI | • | L. Petrucci | FAST | MAE | 07/2004-06/2006 | 24 | 23 | 12 |
| Small grant | | L. Petrucci | | UniSA | 09/2005-12/2005 | 4 | 6 | 0 |

**CerPAN**

*Autres partenaires : Inria-Rocquencourt, CEDRIC (CNAM), LRI (Univ. Paris 11)*

CerPAN : Certification de programmes d'analyse numérique (ANR programme blanc 2005)

Dans ce projet nous nous intéressons au développement et à la mise en application des méthodes permettant de démontrer formellement la correction de programmes issus

du domaine de l'analyse numérique. Beaucoup de programmes critiques sont issus de ce domaine, mais les travaux traitant directement des applications des méthodes formelles aux programmes d'analyse numérique sont rares. La principale raison est l'utilisation intensive des nombres réels (à virgule flottante) dans les programmes numériques, alors que les méthodes formelles manipulent plutôt des nombres entiers, ou plus généralement des structures discrètes. Deux méthodes sont explorées dans ce projet, toutes deux étant vouées à être mises en pratique sur une étude de cas (gradient analytique). La première consiste à utiliser des outils de vérification de programmes (déjà existants) tels que Caduceus. La seconde consiste à effectuer la formalisation et la preuve du problème correspondant au programme ou à l'algorithme étudié, puis à en extraire (principe d'extraction de Coq) un programme qui sera, de fait, prouvé correct.

**PAI FAST & ARC Linkage international & UniSA small divisional grant**

*Autres partenaires : UniSA (University of South Australia), University of Adelaide*

Ce projet, démarré avec un programme Linkage International de l'ARC en 2004, poursuivi jusqu'en juin 2006 par un PAI FAST (French-Australian Science and Technology programme) et complété par une bourse de l'University of South Australia finançant quelques mois de travail d'un étudiant en thèse, s'est intéressé aux techniques modulaires d'analyse de réseaux de Petri. Il comportait quatre volets : l'analyse de réseaux communiquant par partage de données (places), l'étude de systèmes temporisés, la prise en compte des données et la mise en œuvre de ces techniques dans des outils ainsi que des études de cas. Il a conduit à plusieurs publications : [RI-6, RI-4, CO-7, CO-8, CI-2].

### Dissémination

#### invitations, tutoriels, école d'été, tables rondes

EUROTICS, *European Computer Science Summit*, octobre 2005, Zurich, Suisse ; C. Choppy (table ronde)

IASSE, *ISCA International Conference on Intelligent and Adaptative Systems and Software Engineering*, juillet 2004, Nice, France ; C. Choppy (exposé invité)

ICTAC, *International Colloquium on Theoretical Aspects of Computing*, novembre 2006, Tunis, Tunisie ; C. Choppy et L. Petrucci (tutoriel)

ISPS, *International Symposium on Programming and Systems*, mai 2005, Alger, Algérie ; L. Petrucci (exposé invité)

PNS, *Workshop on Petri Nets Standards (ATPN)*, juin 2007, Siedlce, Pologne ; L. Petrucci (exposé invité, synthèse et groupe de travail)

TEACONC, *Workshop on Teaching Concurrency (ATPN)*, juin 2006, Turku, Finlande ; L. Petrucci (table ronde)

GDR-IM, *GDR Informatique Mathématique*, janvier 2007, Montpellier, Rencontres arithmétique de l'informatique mathématique ; M. Mayero (exposé-cours invité)

VECOS, *Workshop on Verification and Evaluation of Computer and Communication Systems (ISPS)*, mai 2007, Alger, Algérie ; L. Petrucci (exposé invité)

### Collaborations et Invitations

#### Collaborations universitaires

- ONERA, co-encadrement par C. Choppy et P. Carle de la thèse de O. Bertrand (depuis février 2006)
- University of South Australia et University of Adelaide (Australie), PAI FAST 2004–2006

- University of South Australia et University of Adelaide (Australie), ARC Linkage International grant, 2004
- University of South Australia small grant (Australie), 2005
- Århus Universitet (Danemark), BRI Paris 13, 2004
- USTHB (Algérie), encadrement de 2 thésards (C. Boukala et S. Mazouz) par L. Petrucci
- LSV, ENS Cachan. Co-encadrement par L. Petrucci et A. Finkel de la thèse de S. Bardin, soutenue le 20 octobre 2005
- Paris 7, encadrement par C. Choppy de la thèse de E. Sendroiu, soutenue le 16 décembre 2004
- groupe de travail MeFoSyLoMa (Méthodes Formelles pour les Systèmes Logiciels et Matériels) : LIP6, Lamsade, LIPN, CEDRIC et LTCI

**Séjours scientifiques**

**Christine Choppy** :

- Université de Gênes, Italie, 1 semaine en août 2003.
- Université de Duisburg-Essen, Allemagne, 1 semaine en Décembre 2003 et 1 semaine en Août 2005.
- University of South Australia et University of Adelaide, Australie, 1 mois en Avril/Mai 2005 et 1 mois en Avril 2006.

**Micaela Mayero** :

- Universidad de la Coruña, Espagne, 1 semaine en septembre 2006 et février 2007.

**Laure Petrucci** :

- Université d'Århus, Danemark, 2 semaines en octobre 2004, et 1 semaine en octobre 2006.
- University of South Australia et University of Adelaide, Australie, 1 mois en Août 2004 et 1 mois en novembre/décembre 2005.
- Université de Yaoundé, Cameroun, 1 semaine en juillet 2004.
- USTHB, Alger, Algérie, 1 semaine en mai 2005, 10 jours en mai 2007.

### 1.1.4 Activités doctorales

**cours de DEA et stages / Master 2 recherche**

**Cours de DEA / Master 2 recherche**

- 2005-2007 (C. Choppy et L. Petrucci) : cours de systèmes parallèles asynchrones (option), master 2 recherche informatique, Paris 13.
- 2005-2007 (C. Choppy et L. Petrucci) : cours de spécification et vérification de systèmes (tronc commun), master 2 recherche informatique, Paris 13.
- 2007 (M. Mayero) : cours de Coq, master 2 recherche mathématiques, Université de la Corogne (Espagne).

**Stages de DEA / Master 2 recherche**

- 2004 (C. Choppy, L. Petrucci) : M. Bounif (DEA Paris 13), stage abandonné.
- 2005 (F. Kordon (LIP6), L. Petrucci, N. Trèves (CEDRIC)) : L. Hillah (M2 SAR Paris 6), *Construction d'un standard d'échange de réseaux de Petri dans le cadre de la norme ISO/IEC-15909.*

- 2006 (C. Choppy, L. Petrucci) : W. Luo (M2R MICR Paris 13), *Une méthode d'analyse et spécification pour les système complexes avec des réseaux de Petri de haut niveau et des spécifications algébriques.*
- 2007 (L. Petrucci) : Steven Gay (MPRI, ENS Cachan), *Raffinement de réseaux de Petri.*
- 2007 (C. Choppy, M. Mayero, L. Petrucci) : N. Ghozlane (M2R MICR Paris 13), *Raffinement de réseaux de Petri de haut niveau.*
- 2007 (K. Klai, L. Petrucci) : Rabih Chahine (M2R SLCHP, INP Toulouse), *Analyse compositionnelle de réseaux de Petri.*

# Bibliographie

### Livres

[LI-1] S. Haddad, F. Kordon et L. Petrucci, éditeurs. *Méthodes formelles pour les systèmes répartis et coopératifs*. Hermès, octobre 2006, ISBN 2-7462-1447-4.

### Chapitres de livres

[CL-2] C. Choppy et L. Petrucci. *Démarches de spécification*, Chapitre 2. Haddad et al. [LI-1], octobre 2006, ISBN 2-7462-1447-4.

[CL-1] L. Petrucci. *Panorama des modèles et langages de spécification*, Chapitre 1. Haddad et al. [LI-1], octobre 2006, ISBN 2-7462-1447-4.

### Édition d'ouvrages collectifs

[ED-2*] R. Janicki et L. Petrucci (eds.). Fifth Special Issue on Application of Concurrency to System Design. *Fundamenta Informaticae*, 2007. À paraître.

[ED-1] K. Goossens et L. Petrucci, éditeurs. *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD'06), Turku, Finland*. IEEE Comp. Soc. Press, juin 2006, ISBN 0-7695-2556-3.

### Articles dans des revues internationales avec comité de lecture sélectif

[RI-6] C. Lakos et L. Petrucci. Modular state space exploration for timed Petri nets. *Journal of Software Tools for Technology Transfer*, 9(3–4) :393–411, juin 2007.

[RI-5] C. Choppy et G. Reggio. A formally grounded software specification method. *Journal of Logic and Algebraic Programming*, 67(1-2) :52–86, 2006.

[RI-4] J. Billington, G. E. Gallasch et L. Petrucci. FAST verification of the class of stop-and-wait protocols modelled by coloured Petri nets. *Nordic Journal of Computing*, 12(3) :275–307, automne 2005.

[RI-3] C. Choppy, D. Hatebur et M. Heisel. Architectural patterns for problem frames. *IEE Proceedings - Software, Special Issue on Relating Software Requirements and Architectures*, 152(4) :198–208, 2005.

[RI-2] C. Choppy et G. Reggio. A UML-Based Approach for Problem Frame Oriented Software Development. *Journal of Information and Software Technology*, 47 :929–954, 2005.

[RI-1] David Delahaye et Micaela Mayero. Dealing with Algebraic Expressions over a Field in Coq using Maple. 39(5) :569–592, 2005.

### Articles dans des revues nationales ou internationales

[RE-4] H. Karoui, R. Kanawati et L. Petrucci. An intelligent peer-to-peer multi-agent system for collaborative management of bibliographic databases. *SGAI's expert update magazine*, 8(3) :22–31, printemps 2006.

[RE-3] F. Kordon et L. Petrucci. Toward formal methods oecumenism ? *IEEE Distributed Systems Online*, 7(7), juillet 2006.

[RE-2] L. Hillah, F. Kordon, L. Petrucci et N. Trèves. Building an API for ISO/IEC 15909, based on model engineering techniques. *Petri Net Newsletter*, 69, octobre 2005.

[RE-1] L. Petrucci. Cover picture story : Experiments with modular state spaces. *Petri Net Newsletter*, 68 :Cover page and 5–10, avril 2005.

### Communications invitées dans une conférence internationale

[IN-2] L. Petrucci. Modular construction of the symbolic observation graph. *Proc. 1st International Workshop on Verification and Evaluation of Computer and Communication Systems (VECOS'07), Algiers, Algeria*, eWiC, page 6. British Computer Society, mai 2007.

[IN-1] L. Petrucci. Modularity and Petri nets. *Proc. 7th International Symposium on Programming and Systems (ISPS'2005), Algiers, Algeria*, pages 7–8, mai 2005.

### Autres communications sur invitation

[AI-2] F. Kordon et L. Petrucci. A formal approach to designing autonomous systems : from Intelligent Transport Systems to robots. *Proc. 2nd National Workshop on Control Architectures of Robots : from models to execution on distributed control architectures, Paris, France*, mai 2007.

[AI-1] L. Petrucci. Iso/iec 15909-2, concepts of high-level nets and cpn. *Presentation at the Workshop on Petri Nets Standards (PNS'07, associated with PETRI NETS'07), Siedlce, Poland*, juin 2007.

### Tutoriaux dans des conférences internationales

[TU-1] C. Choppy, S. Haddad, H. Klaudel, F. Kordon, L. Petrucci et Y. Thierry-Mieg. Tutorial on formal methods for distributed and cooperative systems. *Proc. 3rd Int. Coll. on Theoretical Aspects of Computing (1-day tutorial at ICTAC'06), Tunis, Tunisia*, volume 4281 de *Lecture Notes in Computer Science*. Springer, novembre 2006.

### Communications dans des conférences internationales avec comité de lecture

[CI-12] K. Klai, L. Petrucci et M. Reniers. An incremental and modular technique for checking LTL\X properties of Petri nets. *Proc. 27th International Conference on Formal Methods for Networked and Distributed Systems (FORTE'07), Tallinn, Estonia*, volume 4574 de *Lecture Notes in Computer Science*. Springer, juin 2007.

[CI-11] C. Choppy, D. Hatebur et M. Heisel. Component composition through architectural patterns for problem frames. Pankaj Jalote, éditeur, *Proc. of the Asia Pacific Software Engineering Conference (APSEC-2006)*, pages 27–34. IEEE, 2006.

[CI-10] C. Choppy et G. Reggio. Requirements Capture and Specification for Enterprise Applications : a UML Based Attempt. Jun Han et Mark Staples, éditeurs, *Proc of the Australian Software Engineering Conference (ASWEC-2006)*, pages 19–28. IEEE, 2006.

[CI-9] David Delahaye et Micaela Mayero. Quantifier Elimination over Algebraically Closed Fields in a Proof Assistant using a Computer Algebra System. 151(1) :57–73, 2006.

[CI-8] L. Hillah, F. Kordon, L. Petrucci et N. Trèves. PN standardisation : a survey. *Proc. 26th International Conference on Formal Methods for Networked and Distributed Systems (FORTE'06), Paris, France*, volume 4229 de *Lecture Notes in Computer Science*, pages 307–322. Springer, septembre 2006.

[CI-7] H. Karoui, R. Kanawati et L. Petrucci. COBRAS : Cooperative CBR system for bibliographical reference recommendation. *Proc. 8th European Conference on Case-Based Reasonning (ECCBR'06), Ölüdeniz/Fethiye, Turkey*, volume 4106 de *Lecture Notes in Artificial Intelligence*, pages 76–90. Springer, septembre 2006.

[CI-6*] H. Karoui, R. Kanawati et L. Petrucci. Cooperative CBR system for peer agent committee formation. *Proc. 5th Workshop on Agents and Peer-to-Peer Computing (AP2PC'06), Hakodate, Japan*, Lecture Notes in Computer Science. Springer, mai 2006. À paraître.

[CI-5] C. Choppy et G. Reggio. Improving Use Case Based Requirements Using Formally Grounded Specifications. *Fundamental Approaches to Software Engineering (FASE)*, volume 2984 de *Lecture Notes in Computer Science*, pages 244–260. Springer, 2004.

[CI-4] C. Choppy et G. Reggio. Using UML for Problem Frame Oriented Software Development. Walter Dosch et Narayan Debnath, éditeurs, *Proc of the ISCA 13th Int. Conf. on Intelligent and Adaptative Systems and Software Engineering (IASSE-2004)*, pages 239–244. The International Society for Computers and Their Applications (ISCA), 2004.

[CI-3] L. Kristensen et L. Petrucci. An approach to distributed state space exploration for coloured Petri nets. *Proc. 25th Int. Conf. Application and Theory of Petri Nets (ICATPN'2004), Bologna, Italy, June 2004*, volume 3099 de *Lecture Notes in Computer Science*, pages 474–483. Springer, juin 2004.

[CI-2] C. Lakos et L. Petrucci. Modular analysis of systems composed of semiautonomous subsystems. *Proc. 4th Int. Conf. on Application of Concurrency to System Design (ACSD'04), Hamilton, Canada, June 2004*, pages 185–194. IEEE Comp. Soc. Press, juin 2004.

[CI-1] C. Choppy, D. Hatebur et M. Heisel. Use of Patterns in Formal Development : Systematic Transition From Problems to Architectural Designs. *Recent Trends in Algebraic Development Techniques*, volume 2755 de *Lecture Notes in Computer Science*, pages 205–220. Springer, 2003.

**Workshops et autres conférences avec comité de lecture**

[CO-13] O. Bertrand, P. Carle et C. Choppy. Chronicle modelling using automata and colored Petri nets. *The 18th International Workshop on Principles of Diagnosis (DX-07)*, pages 229–234, 2007.

[CO-12] O. Bertrand, P. Carle et C. Choppy. Vers une exploitation des simulations distribuées par les chroniques. *8e Rencontres nationales des Jeunes Chercheurs en Intelligence Artificielle (RJCIA 07)*, pages 15–30, 2007.

[CO-11] C. Boukala et L. Petrucci. Towards distributed verification of Petri nets properties. *Proc. 1st International Workshop on Verification and Evaluation of Computer and Communication Systems (VECOS'07), Algiers, Algeria*, eWiC, pages 15–26. British Computer Society, mai 2007.

[CO-10] C. Lakos et L. Petrucci. Modular state spaces and place fusion. *Workshop on Petri Nets and Software Engineering (PNSE'07, associated with PETRI NETS'07), Siedlce, Poland*, pages 175–190. University of Podlasie, juin 2007.

[CO-9] S. Mazouz et L. Petrucci. Modular discrete pseudo-state graphs for time Petri nets. *Workshop on Modelling of Objects, Components and Agents (MOCA'06, associated with PETRI NETS'06), Turku, Finland*, numéro 272 Universität Hamburg Bericht, pages 223–244, juin 2006.

[CO-8] J. Billington, G. E. Gallasch et L. Petrucci. Transforming couloured Petri nets to counter systems for parametric verification : A stop-and-wait protocol case study. *Proc. 2nd workshop on MOdel-based Methodologies for Pervasive and Embedded Software (MOMPES'05, satellite of ACSD'05), Rennes, France*, volume 39, pages 37–55. TUCS general publication, juin 2005. ISBN 952-12-1556-9.

[CO-7] C. Lakos et L. Petrucci. Distributed and modular state space exploration for timed Petri nets. *Proc. Workshop on Practical Use of Coloured Petri Nets, Aarhus, Denmark*, pages 191–210, octobre 2005. Proceedings published as Report DAIMI-PB 576, Aarhus, DK.

[CO-6] S. Bardin et L. Petrucci. COAST : des réseaux de Petri à la planification assistée. *Proc. 6ème Conférence sur les Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL'2004), Besançon, France*, pages 285–298, juin 2004.

[CO-5] S. Bardin et L. Petrucci. From PNML to counter systems for accelerating Petri nets with FAST. *Proc. of the Workshop on Interchange Formats for Petri Nets (at ICATPN 2004)*, pages 26–40, juin 2004.

[CO-4] C. Choppy et M. Heisel. Une approche à base de "patrons" pour la spécification et le développement de systèmes d'information. *Proceedings Approches Formelles dans l'Assistance au Développement de Logiciels - AFADL'2004*, pages 61–76, 2004.

[CO-3] C. Choppy et L. Petrucci. Towards a methodology for modelling with Petri nets. *Proc. Workshop on Practical Use of Coloured Petri Nets, Aarhus, Denmark*, pages 39–56, octobre 2004. Proceedings published as Report DAIMI-PB 570, Aarhus, DK.

[CO-2] C. Choppy et G. Reggio. A UML-Based Method for the Commanded Behaviour Frame. K. Cox, J.G. Hall et L. Rapanotti, éditeurs, *Proc. of the 1st International Workshop on Advances and Applications of Problem Frames (IWAAPF 2004)*, pages 27–34. An ICSE 2004 workshop, IEEE, 2004.

[CO-1] H. Karoui, R. Kanawati et L. Petrucci. An intelligent peer-to-peer multi-agent system for collaborative management of bibliographic databases. *Proc. Workshop on Case Based Reasoning, Cambridge, England*, décembre 2004.

**Rapports**

[Ra-7] C. Choppy, L. Petrucci et G. Reggio. A method for modelling with coloured nets. Rapport Technique, Universite Paris 13, 2007.

[Ra-6] Philippe Darondeau et Laure Petrucci. Modular automata 2 distributed Petri nets 4 synthesis. Rapport Technique 6192, INRIA, mai 2007.

[Ra-5] C. Choppy, L. Petrucci et G. Reggio. A method for modelling with place/transitions nets. Rapport Technique, Universite Paris 13, 2006.

[Ra-4] David Delahaye et Micaela Mayero. Diophantus' 20th Problem and Fermat's Last Theorem for $n = 4$ : Formalization of Fermat's Proofs in the Coq Proof Assistant. Rapport Technique, 2005.

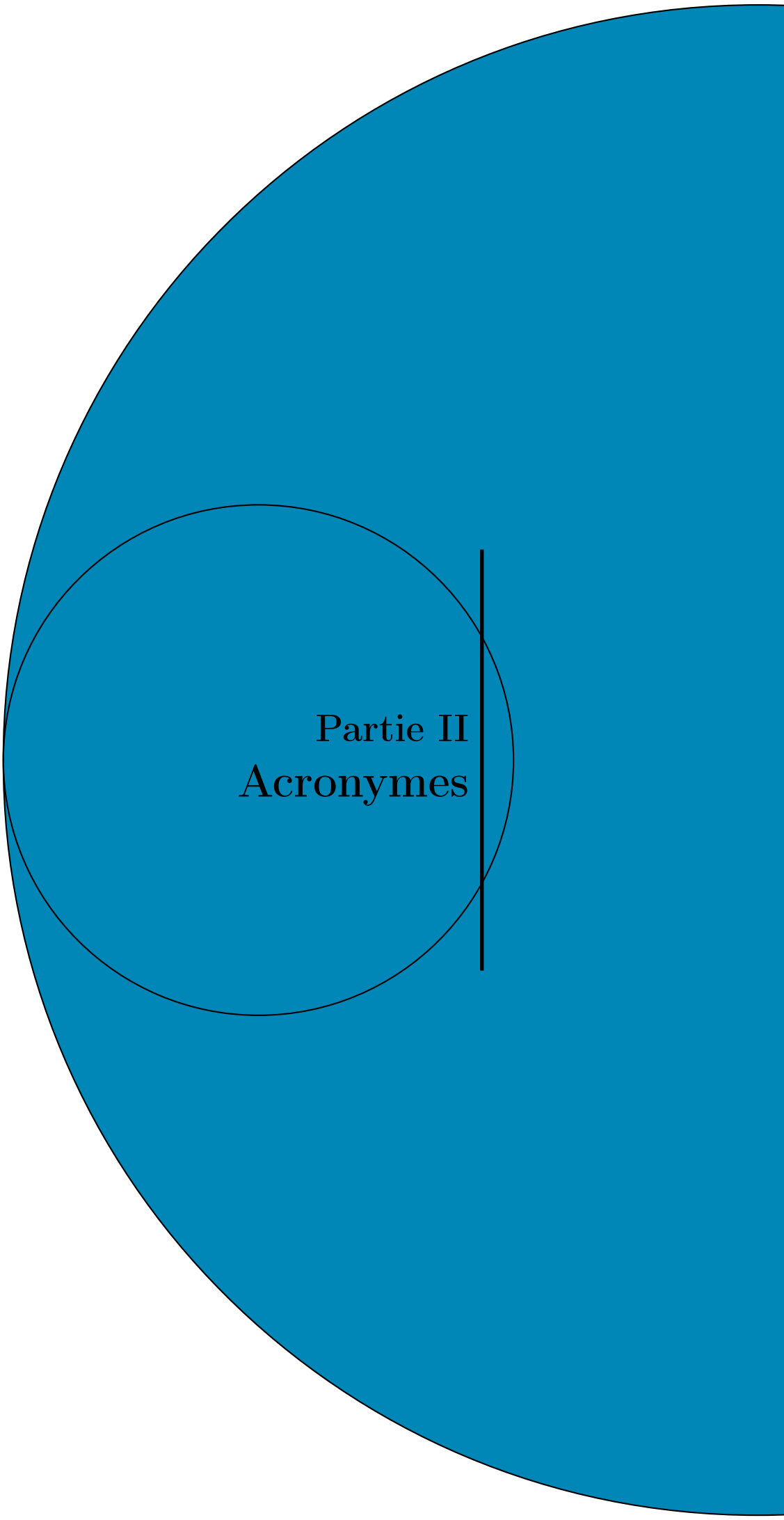
[Ra-3] Micaela Mayero, B. Monsuez et F. Védrine. How an Incoherent Behavior inside generic hardware component characterizes functional errors. Rapport Technique, 2005.

[Ra-2] F. Kordon et L. Petrucci. Proposal for an addendum to ISO/IEC 15909-1, novembre 2004. NWI submitted by AFNOR to ISO/IEC JTC1-SC7-WG19, ref. MAL-012.

[Ra-1] G. Reggio, E. Astesiano et C. Choppy. CASL-LTL : A CASL extension for dynamic reactive systems version 1.0 – summary. Rapport Technique DISI-TR-03-36, Univ. of Genova, 2003.

**Outils**

[Lo-1] S. Bardin, A. Finkel, J. Leroux et L. Petrucci. FAST : Fast Acceleration of Symbolic Transition systems, juin 2005. Tool presentation at ACSD'05.


Partie II
Acronymes

**A³** Apprentissage Artificiel et Applications

**ANR** Agence National de la Recherche

**ARC** Australian Research Council

**CEDRIC** Centre d'Etudes et de Recherche en Informatique du CNAM - EA 1395

**EA** Equipe d'Accueil

**Inria** Institut National de la Recherche en Informatique et Automatique

**LIPN** Laboratoire d'Informatique de Paris-Nord - UMR 7030

**LRI** Laboratoire de Recherches en Informatique, université Paris 11 - UMR 8623

**MAE** Ministère des Affaires Etrangères

**MENESR** Ministère de l'Education Nationale, de l'Enseignement Supérieur et de la Recherche

**ONERA** Office National d'Etudes et de Recherches Aérospatiales

**PAI** Programme Actions Intégrées

**UMR** Unité Mixte de Recherche

**UniSA** University of South Australia