Le LOTOS: Théorie, Outils, Applications.

Souheil Gallouzi ●■

Luigi Logrippo ●

Abdellatif Obaid ❑

● *Université d'Ottawa, Département d'informatique*
Ottawa, Ont. Canada K1N 6N5

■ *Recherches Bell-Northern, BP. 3511, Station C*
Ottawa, Ont. Canada K1Y 4H7

❑ *Université du Québec à Hull, Département d'informatique*
Hull, Qué, Canada J8Y 3G4

Résumé. On donne un bref aperçu de la situation de la recherche sur le langage LOTOS, en ce qui concerne les travaux théoriques, le développement d'outils logiciels, et les applications.

Mots Clé. Langages de spécification, systèmes répartis, protocoles, LOTOS.

0. Introduction

Le LOTOS est une norme internationale depuis 1989 [ISO 89], mais sa définition est (plus ou moins) stable depuis 1987. Il y a désormais un bon nombre d'articles qui parlent de la théorie, des outils, et des applications du langage.

Il est difficile de fournir un compte rendu complet des activités concernant le LOTOS, car ce langage est utilisé par un grand nombre d'équipes sur plusieurs continents. Nous nous limitons à citer les travaux les plus intéressants desquels nous sommes au courant, surtout travaux publiés. De plus, nous ne parlons que des travaux qui parlent du langage dans sa forme courante. Nous avons essayé d'être plus complets dans la bibliographie. Les sources les plus importantes ont été les livres cités dans la section 0 de la bibliographie. Pour un aperçu du langage, voir [ISO

89][Bolognesi 89][Logrippo 92].

1. L'évolution de la théorie du LOTOS.

1.1 Origines théoriques.

Robin Milner est le premier à avoir utilisé une méthode algébrique pour décrire les comportements asynchrones des systèmes communicants comme termes d'un calcul. Différentes notions d'équivalences entre comportements ont été définies dans ce calcul. Ce travail a mené Milner à la découverte de CCS [Milner 80][Milner 89], un langage algébrique de processus. La sémantique de ce langage est décrite en fonction de règles d'inférences. Ces règles rendent possible la définition d'un ensemble de relations d'équivalences entre comportements et un calcul qui permet de raisonner sur les processus en se basant sur un ensemble d'équations. D'où le lien établi, pour la première fois, entre une théorie équationnelle et une relation d'équivalence entre comportements, notamment l'équivalence observationnelle, en se basant sur une sémantique opérationnelle. Il existe une version plus élégante de cette relation d'équivalence, appelée équivalence de bisimulation, qui est définie dans [Park 81][Milner 89]; ses applications sont illustrées dans [Milner 83][Milner 89].

L'action interne, notée τ, est un constructeur important de CCS qui permet la représentation des comportements non-déterministes. L'action τ est cachée de son environnement et lorsqu'un processus est prêt à s'engager dans τ, il lui est permis de le faire. Ce constructeur joue un rôle clé dans le CCS, pour obtenir un pouvoir d'expression maximal avec un nombre minimal d'opérateurs.

Une autre école de pensée est basée sur CSP, qui depuis son apparition [Hoare 78] a joué un rôle important dans l'évolution des théories des systèmes concurrents. Cette école s'est beaucoup inspirée de l'approche algébrique de Milner pour changer CSP d'une façon radicale [Brookes 84][Hoare 85]. La communication binaire entre processus concurrents, comme dans CCS, est remplacée par une synchronisation multiple et la communication entre les valeurs a été remplacée par des actions symboliques. La nouvelle version de CSP, aussi nommée TCSP ("Theoretical CSP"), consiste en un ensemble d'opérateurs et une définition récursive des processus. Sa sémantique est représentée à l'aide d'un modèle de dénotation appelé le modèle d'échec ("Failure model") [Brookes 84][Hoare 85], qui est utilisé pour justifier les équations entre les opérateurs. Chaque échec possible d'un processus décrit un comportement dans lequel le processus pourrait s'engager dans une séquence d'actions visibles à un moment donné, appelée trace, après laquelle le processus pourrait refuser de participer dans un ensemble d'actions, appelé refus. Cependant, au lieu d'utiliser τ pour représenter les comportements non-déterministes, CSP illustre ces comportements de la façon suivante: Après qu'un processus s'engage dans une tra-

ce, il peut refuser certains ensembles d'actions. Ces refus représentent les conséquences disponibles, pour l'étape suivante, des différentes décisions non-déterministes que le processus peut prendre. Une notion d'équivalence d'échec (*failure equivalence*) permet d'identifier les processus ayant le même ensemble d'échecs comme dénotation.

Des études comparatives entre CCS et CSP ont été élaborées dans [Brookes 83][vanGlabbeek 86]. En particulier, le travail de [Brookes 83] identifie explicitement les différences entre les deux systèmes et montre que les modèles sémantiques de CCS et CSP sont comparables.

LOTOS n'est autre que le fruit du couplage de CCS et CSP. Plus précisément, CCS est utilisé comme la base sémantique de la partie contrôle de LOTOS (la partie qui traite la description des comportements des processus et des interactions). CSP a aussi influencé la conception de LOTOS, surtout dans la définition de l'opérateur de composition parallèle. En séparant les opérateurs d'effacement et de composition parallèle (qui sont reliés en CCS), plusieurs processus peuvent s'engager dans une seule action. D'où l'introduction du style de spécification orienté-contraintes [Vissers 88][Turner 88].

1.2 Modèles sémantiques.

Les opérateurs de LOTOS ont été choisis de telle façon qu'il est possible de démontrer un ensemble de règles algébriques sur ces derniers, comme celles de CCS. Cependant, il faut noter que ni CCS ni LOTOS n'offre une théorie algébrique complète qui permet le traitement et l'interprétation de tous les comportements possibles. Le fait que LOTOS utilise l'action interne i, équivalente de τ dans CCS, pour représenter les comportements non-déterministes et que le modèle sémantique de LOTOS a hérité du caractère opérationnel de CCS, a permis l'utilisation du concept d'équivalence de bisimulation pour dériver ces règles algébriques et avoir une définition équationnelle des comportements LOTOS. Le modèle le plus complet dans la littérature est LOTCAL [Brinksma 88] qui identifie l'ensemble minimal d'opérateurs suffisant à l'interprétation formelle de LOTOS. LOTCAL est en grande partie basé sur CCS.

Il existe d'autres modèles sémantiques basés sur la théorie de CSP qui proposent un traitement de dénotation des comportements LOTOS [Gallouzi 89][Gallouzi 91a][Gallouzi 91b][Leduc 90]. Ces modèles interprètent les comportements des processus LOTOS à l'aide de leurs ensembles d'échecs. On montre que les ensembles d'échecs des processus composés peuvent être construits à partir des ensembles d'échecs de leurs composantes. Le résultat est une sémantique alternative pour LOTOS ayant un lien solide avec l'interprétation opérationnelle des comportements. Le travail de [Leduc 90] se distingue du nôtre par le fait qu'il propose une extension du modèle d'échec. Cette extension n'identifie pas une expression de comportement

qui ne peut pas s'engager dans une action interne au départ avec une autre qui a le même ensemble d'échecs comme dénotation, mais peut s'engager dans une action interne au départ. Cette extension résout le problème de la monotonie des opérateurs LOTOS par rapport à la relation d'ordre partiel induite par le modèle d'échec. Cependant, le traitement des processus divergents dans un tel modèle n'est pas complètement résolu. D'où la difficulté de donner une dénotation appropriée aux processus récursifs.

Les processus divergents ont été le sujet de plusieurs recherches. Dans [Brookes 84][Brookes 85] la divergence est définie à l'aide d'un processus spécial, nommé CHAOS, qui ne permet pas de distinguer les comportements des processus après qu'ils commencent à diverger. Dans CSP, on définit un système de dénotation qui permet de calculer les traces après lesquelles un processus est capable de diverger, c.-à-d. se comporter comme CHAOS. L'adaptation de cette méthode à LOTOS n'est pas évidente, surtout parce que la divergence en LOTOS n'est pas équivalente à CHAOS, puisqu'elle est représentée par une séquence infinie d'actions internes (i) dans le modèle opérationnel. Un autre traitement de la divergence basé sur le modèle d'échec a été proposé dans [Bergstra 86]. Ce dernier représente la divergence par un processus spécial nommé DELAY, une idée inspirée de SCCS [Milner 83]. Ce traitement diffère du premier [Brookes 84][Brookes 85] par le fait qu'il permet de distinguer les comportements des processus après qu'ils commencent à diverger. Cette approche est plus conforme à l'interprétation opérationnelle des comportements LOTOS.

Le modèle d'échec de CSP ainsi que le modèle d'acceptation [DeNicola 84] [Hennessy 85][Hennessy 88] ont été les sources d'inspiration du travail de Brinksma et al. [Brinksma 87][Brinksma 89], qui proposent une extension des théories d'équivalences d'échec et de test de CSP et CCS, respectivement, à LOTOS en se basant sur une sémantique opérationnelle. Ce traitement a mené à une relation d'équivalence qui est plus faible que la bisimulation et qui identifie des comportements ayant différents points de divergences. Les expressions de comportements LOTOS sont interprétées comme des arbres d'échecs, dont les noeuds sont étiquetés avec les ensembles de refus représentant les ensembles d'actions que le processus peut rejeter à un moment donné. Le même traitement pourrait se faire à l'aide des arbres d'acceptations proposés par Hennessy [Hennessy 85].

D'autres notions importantes ont aussi été introduites telles que les relations d'extension et de conformité et le concept des testeurs canoniques. Ces derniers ont joué un rôle clé dans la définition de la théorie de dérivation des tests pour tester la conformité d'une implantation donnée par rapport à sa spécification. Dans ce contexte, une implantation est un processus LOTOS relativement plus élaboré que la spécification (un autre processus LOTOS). L'idée de base est de dériver un processus tes-

teur à partir de la spécification et de le composer en parallèle avec le processus implantation afin de démontrer qu'on ne peut arriver à un état d'impasse. D'où la conformité de l'implantation par rapport à sa spécification. L'algorithme de dérivation des tests (ou processus testeurs) est exprimé à l'aide des arbres d'échecs. Cette méthode s'applique à toutes les méthodes de descriptions formelles qui permettent une interprétation sémantique des spécifications en fonction des systèmes de transitions étiquetés munis d'actions internes.

Le travail de [Fantechi 90][Fantechi 90a] propose une approche différente pour interpréter les comportements LOTOS. Cette approche utilise la logique temporelle pour définir un modèle sémantique pour LOTOS. Un tel modèle permet l'interprétation de chaque expression de comportement LOTOS comme formule de la logique temporelle.

D'autres tentatives proposant des modèles sémantiques alternatifs pour LOTOS sont rapportés dans [van Eijk 89].

1.3 Méthodes de preuve.

En général, on est intéressé à prouver deux genres de propriétés des systèmes distribués: solidité ("safety") et vivacité ("liveness"). Les propriétés de solidité décrivent les comportements désirés des systèmes et les propriétés de vivacité décrivent des comportements non-désirés. D'où le besoin d'une ou plusieurs méthodes qui permettent de démontrer qu'un processus LOTOS donné satisfait une ou plusieurs propriétés de ce genre. De plus, la conception des systèmes distribués se fait sur plusieurs niveaux d'abstractions. D'où le besoin de démontrer qu'un processus LOTOS est équivalent à un autre processus LOTOS plus élaboré afin de valider sa conception.

La méthode de preuve la plus utilisée pour prouver qu'un processus LOTOS satisfait certaines propriétés est basée sur l'équivalence de bisimulation. Un rapport sur les algorithmes et outils associés à cette méthode peut être trouvé dans [Bolognesi 89a]. En utilisant la bisimulation, on peut prouver qu'un processus LOTOS est équivalent à un autre processus LOTOS. Des exemples de cette façon de procéder sont présentés dans [Najm 87][Shiratori 90], où on montre que des processus services sont équivalents à la composition des processus protocoles avec le service fournisseur associé. On peut aussi se servir de l'équivalence de test si on veut prouver qu'il n'existe pas d'expérience qui peut distinguer deux processus donnés.

Cette méthode a beaucoup de mérite, surtout qu'elle joue un rôle important dans la conception des systèmes distribués, sauf que plusieurs propriétés ne peuvent être formulées ou prouvées de cette façon.

Une méthode complémentaire est proposée dans [Gallouzi 89][Gallouzi 91a][Gal-

louzi 91b]. Cette méthode utilise la relation de satisfaction de Hoare [Hoare 85] et se base sur le modèle d'échec de CSP. Le système de preuve proposé permet de construire des preuves d'un processus composé à partir des preuves de ses composantes. Les propriétés d'un processus sont exprimées sous forme de propositions logiques de premier ordre, en fonction des propriétés des ensembles de traces ou d'échecs du processus. Les propriétés des ensembles de traces peuvent être utilisées pour exprimer les propriétés de solidité, vu qu'elles contiennent des informations positives sur les comportements, et celles des ensembles d'échec peuvent être utilisées pour exprimer les propriétés de vivacité, vu qu'elles contiennent des informations négatives sur les comportements. L'inconvénient de cette méthode est relié à sa complexité et au fait qu'il n'existe aucun principe général d'induction et de point fixe qu'on puisse utiliser dans les preuves.

Une approche similaire est proposée dans [Fantechi 90]. Elle utilise la logique temporelle pour exprimer les propriétés des processus et se base sur une sémantique temporelle des comportements LOTOS. L'application de cette méthode est encore limitée, vu sa complexité.

2. Les outils de LOTOS.

Depuis le début des travaux sur le langage LOTOS, un accent a été mis sur l'élaboration d'outils. Bien que la plupart de ceux-ci ont été produits sous forme de prototypes, on peut y constater une certaine qualité et la rigueur (surtout sémantique) de leur analyse et de leur mise en oeuvre. On peut classifier les outils disponibles en quatre catégories: - les outils de simulation et d'exécution, - les outils de transformation, - les outils de validation, - les outils de représentation graphique.

2.1. Outils de simulation et d'exécution.

Ces outils font en général l'analyse syntaxique, la vérification de la sémantique statique et la génération du code intermédiaire en vue d'une exécution. Récemment, il y a eu des travaux sur des compilateurs ayant pour fonction de générer des processus parallèles et de gérer la communication par rendez-vous multiples. Les méthodes d'implantation de ces outils reposent soit sur des fonctions du système d'exploitation sur lequel s'est effectuée cette implantation (généralement Unix) soit sur des modèles d'exécution pouvant être implantés sur toutes sortes de machines (modèles abstraits, langages spécialisés, ...). Parmi ces outils, nous citons les plus connus dont:

LOTOS/OTTAWA: Développé à l'Université d'Ottawa [Logrippo 90b]. C'est le

précurseur des interpréteurs du langage LOTOS. La version la plus récente porte le nom de ISLA [Guillemot 88]. Ce système prend comme entrée une spécification en LOTOS et l'analyse syntaxiquement et sémantiquement pour en produire un code qui sera interprété. Le système couvre aussi la partie des données abstraites de LOTOS ainsi que la librairie standard des types de données. L'exécution se fait soit en mode pas à pas, soit en mode de génération d'arbre symbolique dans lequel les valeurs des variables sont traitées symboliquement. Il comprend un grand nombre de commandes permettant de naviguer facilement à travers le comportement. En plus il permet de manipuler des prédicats complexes. En effet, le système offre la possibilité de définir une base de données des valeurs sur lesquelles ces prédicats peuvent être vérifiés et de désigner les expressions de valeurs complexes par des noms de constantes. Récemment, une version de ce système basée sur X-Windows a été développée rendant l'interface plus conviviale et permettant même d'afficher des spécifications selon un mode graphique proche de ce qui est proposé à l'ISO [ISO 89a].

HIPPO: C'est un système développé dans le cadre du projet ESPRIT/ SEDOS [Van Eijk 89a] [Tretmans 89] en Europe. Sa fonctionnalité est sembable à celle de ISLA. Il permet la simulation après avoir effectué l'analyse syntaxique et sémantique. Il exécute les spécifications de comportements et effectue l'évaluation des expressions de valeurs. Il permet de construire des arbres de communication de la spécification qui contient l'historique de l'exécution jusqu'au point courant. Il est muni d'un éditeur dirigé par la syntaxe.

SMILE: Explore les comportements de LOTOS, fait une exécution symbolique et résout les équations conditionnelles sur les données de façon symbolique. Les comportements sont considérés sans instanciation des variables (d'une façon similaire à celle utilisée dans [Guillemot 88]). La principale utilisation de ce système est la construction des arbres symboliques [Van Eijk 90].

LOTOS/GIPE: Utilisé sur le système CENTAUR du projet GIPE (Generation of Interactive Programming Environment) pour la génération d'un environnement pour LOTOS [de Jager 90]. Etant donné les spécifications lexicales, syntaxiques et sémantiques d'un langage, CENTAUR fournit un analyseur et un simulateur. Pour LOTOS, l'environnement consiste en un éditeur dirigé par la syntaxe, un paragrapheur, etc. Le simulateur proprement dit est encore en cours de développement.

OBJ/LOTOS: Ce système permet de simuler des comportements décrits en LOTOS [Ohmaki 90]. Il est basé sur la sémantique opérationnelle telle que celle utilisée

pour LOTOS mais décrite en OBJ, un langage formel de description basé sur la logique équationnelle. L'avantage d'un tel formalisme est de permettre une implantation des types de données abstraits de façon directe en plus de permettre la construction d'un système de preuves de théorèmes.

JAPON/LOTOS: Un compilateur pour LOTOS est décrit dans [Nomura 90]. Ce système gère la communication et l'ordonnancement entre processus parallèles. Cet ordonnanceur est implanté en C et prend en charge la gestion des événements et de leurs files d'attente. Un modèle abstrait sous forme d'arbre de synchronisation est utilisé. Il définit la hiérarchie entre processus ainsi que leurs points de synchronisation.

MONTREAL/LOTOS: L'équipe de Montréal s'intéresse à une implantation répartie du langage LOTOS [Bochmann 90]. Les processus LOTOS sont construits sur la base d'un modèle abstrait d'un arbre de synchronisation permettant de faire circuler l'information entre processus parallèles qui participent à une communication. En plus de cela, un algorithme d'implantation de rendez-vous multiples a été introduit. Il permet de faire communiquer des processus répartis en réseau. L'intérêt d'un telle implantation est qu'elle pourrait être la base d'une implantation de spécifications LOTOS dans un environnement physique réparti.

LOTOS/C: Ce système est l'un des premiers "Compilateurs" de LOTOS [Mañas 89]. Après avoir effectué l'analyse syntaxique et sémantique, ce système génère du code en langage C exécutable sur une plate-forme Unix. On associe aux actions des annotations (sous forme de commentaires) qui indiquent la procédure en C à exécuter lorsque cette action est atteinte par l'exécuteur. Cette procédure peut être un appel à une primitive de communication, une opération de "delay" ou l'attente d'une condition:

```
send! m (*! send (m) ! *); (*! wait isdata () !*) read? d: data; . . .
```

Le système permet de manipuler les types de données abstraits en produisant une représentation basée entre autres sur les types du langage C.

Pour exécuter une spécification, on doit définir les processus ainsi que leur environnement. Le système alors peut évoluer de lui-même en effectuant les communications internes au système ainsi spécifié.

TETRA: Cet outil [Bochmann 89] est bâti sur ISLA. Il peut être utilisé pour déterminer si une trace d'exécution est une trace légale pour une spécification donnée. Si

la trace n'est pas légale, TETRA essaie de donner une explication de l'erreur. Cet outil peut donc être utilisé pour valider des séquences de tests pour un protocole ou service spécifié en LOTOS [Bochmann 90a].

2.2.Outils de transformation.

LOLA: LOLA est un outil de transformation de spécifications LOTOS [Quemada 88]. Il effectue l'expansion d'expressions paramétrisées et élimine les boucles d'événements internes apparaissant dans le comportement. Un des problèmes de l'expansion est le risque d'explosion de l'ensemble des états. Pour résoudre ce problème, les développeurs ont introduit la notion d'expansion paramétrisée où certaines variables "contextuelles" sont introduites pour rendre fini les comportements dynamiques des systèmes, en utilisant une représentation récursive paramétrisée des processus.

L'objectif visé par ce système est de faciliter le processus de vérification en éliminant la composition parallèle des processus, et d'aider à l'implantation des systèmes spécifiés. De plus, elle permet de transformer les spécifications écrites dans le style orienté-contraintes en spécifications dans un style monolithique.

PIL: PIL [Queiroz 90] est un système interactif de transformation de spécifications LOTOS vers un formalisme inspiré des réseaux hiérarchisés d'automates (appelé PRIMOL). Le but du système est en fait de permettre une pré-implantation de spécifications LOTOS et d'étudier certaines propriétés des spécifications. PRIMOL exécute ces automates en permettant à chaque noeud parent d'activer l'exécution des noeuds fils correspondant aux processus formant une composition (parallèle, interruption, ...). Les variables états sont introduites pour indiquer les changements à la suite de l'exécution des transitions et pour indiquer la composante à exécuter. Le langage PRIMOL permet de contrôler les transitions en utilisant des conditions qui doivent être vérifiées pour qu'une transition soit exécutée.

2.3.Outils de vérification.

Les outils de vérification sont les moins nombreux. Ces outils sont très souvent basés sur les notions d'équivalence observationnelle et bisimulation (section 1).

SQUIGGLES [Bolognesi 89a]: C'est un outil de vérification basé sur la notion d'équivalence observationnelle. Il a été implanté en C et en Prolog. Il comprend plusieurs composantes, dont un analyseur syntaxique et un traducteur vers une représentation interne. D'autres modules vérifient la bisimulation forte, la bisimula-

tion faible et l'équivalence de traces. Le système commence par transformer la spécification donnée en spécification régulière et l'algorithme de décision d'équivalence est linéaire. On ne considère que le LOTOS *pur*.

L'interface est assez conviviale donnant le choix entre différents types de tests d'équivalence.

CAESAR-ALDEBARAN [Garavel 91]: Cet outil traduit une spécification LOTOS dans un réseau de Petri (partie contrôle) et dans des procédures C (partie données). La spécification traduite est exécutée avec le but d'obtenir un graphe d'états fini. Si un tel graphe peut être obtenu, il peut être vérifié en utilisant ou bien la logique temporelle (*model checking*), ou bien des relations d'équivalences d'automates.

BOYER-MOORE/LOTOS: Dans [Aujla 88] est décrit un système de vérification selon la méthode Boyer-Moore appliquée à LOTOS. Le but du système est de permettre de vérifier des prédicats sur les transitions d'une spécification LOTOS. Afin de pouvoir appliquer cette méthode, la sémantique du langage a été écrite en LISP. Ceci permet de pratiquer des preuves par induction basées sur des axiomes proprement construits pour LOTOS (tel que l'axiome de commutativité de l'opérateur de choix par exemple). Le système a été conçu comme une tentative d'utilisation de la technique Boyer-Moore pour LOTOS. Les types de données abstraits ne sont pas traités.

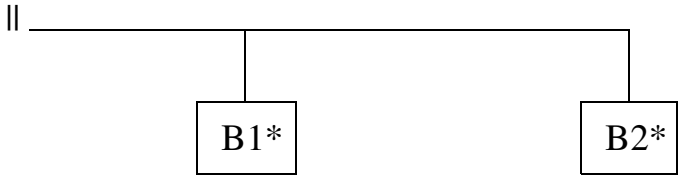
2.4.Outils graphiques.

Plusieurs outils ont été développés suite aux travaux sur la représentation graphique de LOTOS dans le cadre de l'ISO [ISO 89a]. Cependant, quelques uns de ces outils adoptent leurs propres notations.

G-LOTOS: A partir d'une spécification LOTOS, ce système affiche une représentation graphique (selon la proposition de G-LOTOS de l'ISO). La représentation utilise une description externe de haut niveau à l'aide d'un langage déclaratif appelé GSL basé sur les attributs sémantiques pour décrire des objets graphiques emboîtés. L'interface utilisateur est assez souple en ce qu'elle permet de naviguer à travers la représentation à l'aide de menus, fenêtres, ... La spécification d'un dessin se fait au moyen de boîtes typées (par les différents types d'expressions de comportements), de contraintes graphiques et d'attributs géométriques [Franchi 90].

UO-GLOTOS: Ce système a été développé à l'Université d'Ottawa [Cheung 90]. Son but est de fournir un environnement de simulation de comportements basé sur

une représentation graphique des spécifications LOTOS. Les développeurs de ce système ont choisi un représentation simple sous forme d'arborescence des expressions de comportement. Par exemple l'expression: B1 || B2 est représentée par:



où B1* et B2* sont les représentations graphiques de B1 et B2 respectivement. Le système admet quatre modes d'exécution:

- exécution action par action.
- exécution d'une séquence d'actions qu'on applique à une spécification.
- exécution automatique dans laquelle l'utilisateur doit intervenir pour faire un choix lorsque celui-ci se présente (en cas de choix non-déterministe par exemple).
- exécution automatique dans lequel l'utilisateur intervient pour fournir des données lorsqu'elles sont nécessaires.

3. Les applications de LOTOS.

Le domaine d'application le plus important de LOTOS est celui pour lequel le langage a été conçu, c'est-à-dire le domaine des protocoles de communication, et surtout des protocoles de la famille de l'OSI.

La spécification du service du transport [Scollo 89] [ISO 90] [ISO 90a] a été particulièrement importante. Elle a été élaborée (ou, mieux, elle est encore en train d'être élaborée) par un comité de l'ISO, dont le rapporteur est G. Scollo. Avant tout, elle traite un service très bien connu et souvent utilisé comme exemple, parce qu'il a été un des premiers services OSI a avoir été standardisé. Ce service est simple et présente des caractéristiques typiques de plusieurs services OSI. De plus, cette spécification a été une des premières spécifications de taille. Donc, elle a été beaucoup étudiée et a influencé les spécifications développées par la suite.

Le style utilisé est celui dit orienté-contraintes. Au premier niveau, le service du transport est décrit comme la conjonction parallèle de quatre contraintes, ou processus principaux:

- un processus qui gère la multiplicité des connections
- un processus qui gère l'identification des connections
- un processus qui règle l'acceptation de nouvelles connections
- un processus qui règle le débit d'acceptation de données (contrôle de flux).

Cette structure se répète aux niveaux plus bas: par exemple, le premier processus est à son tour défini comme la composition parallèle de deux processus: un processus qui représente les contraintes locales à une connection, et un autre processus qui représente les contraintes de bout en bout pour chaque connection. Cette spécification a illustré pour la première fois l'application du style orienté-contraintes.

Le même groupe de l'ISO a aussi produit une spécification des classes 0-3 du protocole de transport. Pendant la préparation de ce rapport une limitation du langage fut découverte. En LOTOS, tout processus qui participe à une synchronisation est mis au courant des valeurs établies pendant cette dernière. Pourtant, dans le protocole de transport il y a des situations dans lesquelles il serait beaucoup plus convenable d'avoir un partage partiel des valeurs. Par exemple, les actions

g ?x: Octet ?n: nat et g ?y:Octet !true

ne peuvent pas synchroniser en LOTOS. Cependant, il serait parfois convenable s'ils le pouvaient et si le résultat de cette synchronisation était l'événement

g ?z:Octet ?n:nat !true,

où seulement le processus de la première action connait la valeur de n, et seulement le processus de la deuxième connait la valeur true.

Dans la spécification du protocole de transport, on change la sémantique du LOTOS comme si une telle synchronisation était permise. À partir de cette spécification, qui proprement est erronée, il est possible d'obtenir une spécification correcte par un processus d'expansion.

Ce point a été développé ultérieurement dans [Brinksma 89], où l'auteur décrit une extension du langage dans laquelle une action peut avoir plusieurs portes, ce qui permet de spécifier que des processus différents peuvent synchroniser sur différentes parties d'une même action.

Les spécifications du niveau session existantes [Ajubi 89a][Ajubi 89b][van Sinderen 89a][van Sinderen 89c][ISO 89b][ISO89c] sont semblables en méthodologie et structure.

De nombreuses autres spécifications de protocoles, standard ou non, existent aujourd'hui.

[Freestone 89] propose une spécification au niveau application. [Kouyzer 91] discute l'application du langage pour la spécification d'un "objet de gestion".

[Gueraichi 89] [Gueraichi 90] donnent une spécification complète du protocole LAP-B (X.25 niveau 2), et montrent comment une suite de tests de conformité a pu être obtenue à partir de cette spécification. [Navarro 91] traite de l'application du LOTOS à la spécification d'un service ISDN niveau 3 (canal D).

[van Sinderen 91] a utilisé le langage dans une application de courrier électronique. [Varadharajan 89] discute son application à un problème d'authentification dans un système de courrier électronique, avec un mécanisme d'échange de clés de chiffrement.

Récemment, on a vu l'application du langage à des protocoles de type différent: [Iglewski 91] discute la spécification d'un protocole d'accès à un réseau local "token-ring", un sous-ensemble de la norme IEEE 802.5. Il faut noter qu'il s'agit d'un protocole "de bas niveau" et que le LOTOS a été suffisant pour spécifier le protocole presque dans sa totalité.

Une autre application importante du LOTOS est dans le domaine téléphonique. [Faci 91] [Faci 90] décrivent deux façons différentes de spécifier un service téléphonique de base (POTS). [Boumezbeur 91] s'occupe de la spécification de services téléphoniques plus compliqués que POTS, comme renvoi automatique, garde, transfert d'appel, conférence, et rappel automatique, tandis que [Quemada 89a] discute de la spécification d'un noeud de commutation, et [Cam 90] s'occupe de la spécification de systèmes cellulaires.

Quelques applications spécialisées ont aussi été traitées: [Logrippo 90a] discute de l'application du langage dans une application industrielle concernant la documentation d'un système de PBX distribué, déjà existant à la compagnie Gandalf. [Pecheur 90] présente la spécification d'un système d'exploitation réparti. [Fernandez 89] discute du développement d'une architecture de protocole sur un système intégré et distribué en temps réel. Le système concerné est un terminal mobile de communication par satellite, destiné à être installé sur un avion, camion ou navire. Une application reliée est celle de [Taylor 91], qui concerne la spécification et la validation d'une famille de protocoles développés dans le cadre du Comité Consultatif pour les Systèmes de Données Spatiales, pour un système de communication des données dans l'espace (Advanced Orbiting Systems).

Il est à noter que quelques unes de ces applications ont été développées sans l'utilisation d'outils. Les auteurs ont conclu que le développement de la spécification exacte d'un système a permis par lui-même de clarifier plusieurs problème de conception et d'obtenir une qualité de conception qui n'aurait pas pu être atteinte avec les méthodes conventionnelles.

En conclusion, le langage LOTOS s'est montré à la hauteur de la tâche comme langage de spécification d'applications générales, non seulement dans le domaine OSI,

mais bien au-delà. Cependant, quelques limitations existent, une desquelles a déjà été mentionnée. Evidemment, l'application du langage à des protocoles de niveau physique paraît être inappropriée. L'application à la spécification de systèmes en temps réel, dans lesquels les contraintes temporelles jouent un rôle important, est aussi inappropriée. Elle le restera au moins jusqu'au jour où il sera possible de spécifier les contraintes temporelle dans le langage (ceci a été le sujet de plusieurs articles: v. entre les plus récents [Quemada 90][van Hulzen 90]). Par ailleurs, la séparation nette qui existe dans le langage entre expressions de valeur et expressions de comportement rend ardue la représentation de certaines situations, comme la création de services définis par un usager [Stefani 91].

Bibliographie

SECTION 0

[Aggarwal 89] S. Aggarwal, K. Sabnani (eds.) *Protocol Specification, Testing, and Verification, VIII.* North-Holland, 1989.

[Bolognesi 89] T. Bolognesi, E. Brinksma. *Introduction to the ISO Specification Language LOTOS.* Dans [van Eijk 89] 23-73.

[Brinksma 90] E. Brinksma, G. Scollo, C. Vissers (eds.) *Protocol Specification, Testing, and Verification IX.* North-Holland, 1990.

[ISO 89] IS 8807. LOTOS, A Formal Description Technique Based on the Temporal Ordering of Observational Behavior. 1989.

[Logrippo 90] L. Logrippo, R.L. Probert, H. Ural (eds.) *Protocol Specification, Testing, and Verification, X.* North-Holland, 1990.

[Logrippo 92] L. Logrippo, M. Faci, M. Haj-Hussein. An Introduction to LOTOS: Learning by Examples. À paraître dans COMNET.

[Pehrson 91] B. Pehrson, B. Jonsson, J. Parrow (eds.) *Protocol Specification, Testing, and Verification, XI.* North-Holland (à paraître)

[Quemada 91] J. Quemada, J. Mañas, E. Vazquez (eds.) *Formal Description Techniques, III.* North-Holland (à paraître).

[Rudin 87] H. Rudin, C.H. West (eds.) *Protocol Specification, Testing, and Verification, VII.* North-Holland, 1987.

[Sarikaya 87] B. Sarikaya, G. v. Bochmann (eds.) *Protocol Specification, Testing, and verification, VI.* North-Holland, 1987.

[Turner 89] K.J. Turner (ed.) *Formal Description Techniques.* North-Holland, 1989.

[van Eijk 89] P.H.J. van Eijk, C.A. Vissers, and M. Diaz (eds.). *The Formal Des-*

cription Technique LOTOS. North-Holland, 1989.

[Vuong 90] S.T. Vuong (ed.) *Formal Description Techniques, II*. North-Holland, 1990.

SECTION 1

[Bergstra 86] J.A. Bergstra, J.W. Klop, E.-R. Olderog. Failures without CHAOS: A New Process Semantics for Fair Abstraction. Rapport Technique CS-R8625, Centrum voor Wiskunde en Informatica, Amesterdam, Pays-Bas, 1986.

[Bolognesi 89a] T. Bolognesi, M. Caneve. Equivalence Verification: Theory, Algorithms and a Tool. Dans [vanEijk 89] 303-326.

[Brinksma 87] E. Brinksma, G. Scollo, C. Steenbergen. LOTOS Specifications, their Implementations and their Tests. Dans [Bochmann 87] 349-360.

[Brinksma 88] E. Brinksma. On the Design of Extended LOTOS. Thèse de doctorat, Université de Twente, Pays-Bas, 1988.

[Brinksma 89] E. Brinksma. A Theory for the Derivation of Tests. Dans [Aggarwal 89] 63-74.

[Brookes 83] S.D. Brookes. On the relation of CCS and CSP. Dans G. Goos and J. Hartmanis, eds, Automata, Languages and Programming, 10th Colloqium. LNCS volume 154, Springer-Verlag, 1983.

[Brookes 84] S.D. Brookes, C.A.R Hoare, A.W. Roscoe. A Theory of Communicating Sequential Processes. JACM,31(3):560-599, 1984.

[Brookes 85] S.D Brookes, A.W. Roscoe. An Improved Failures Model for Communicating Sequential Processes. Dans S.D. Brookes, A.W. Roscoe, and G. Winskel, eds, NSF-SERC Seminar on Concurrency. LNCS volume 197, Springer-Verlag, 1985.

[DeNicola 84] R. DeNicola, M. Hennessy. Testing Equivalence for Processes. TCS, 34: 83-133, 1984.

[Fantechi 90] A. Fantechi, S. Gnesi, C. Caneve. Dans [Vuong 90], 261-276.

[Fantechi 90a] A. Fantechi, S. Gnesi, G. Ristori. Dans [Logrippo 90] 365-378.

[Gallouzi 89] S. Gallouzi. Trace analysis of LOTOS behaviours. Thèse de Maîtrise, Université d'Ottawa, Canada, 1989.

[Gallouzi 91a] S. Gallouzi, L. Logrippo, A. Obaid. A Hoare style proof system for LOTOS. Dans [Quemada 91].

[Gallouzi 91b] S. Gallouzi, L Logrippo, A. Obaid. An Expressive Trace Theory for LOTOS. Dans [Pehrson 91].

[Hennessy 85] M. Hennessy. Acceptance Trees. JACM, 32(4), 1985: 896-928.

[Hennessy 88] M. Hennessy. *Algebraic Theory of Processes.* Foundations of Computing Series. The MIT Press, 1988.

[Hoare 78] C.A.R. Hoare. Communicating Sequential Processes. Communications of the ACM, 21(8), 1978.

[Hoare 85] C.A.R. Hoare. *Communicating Sequential Proceeses.* Prentice-Hall, 1985.

[Leduc 90] G. Leduc. On the Role of Implementation Relations in the Design of Distributed Systems using LOTOS. Thèse de doctorat, Université de Liège, 1990.

[Milner 80] R. Milner. *A Calculus of Communicating Systems,* volume 92 Lecture Notes in Computer Science. Springer-Verlag, 1980.

[Milner 83] R. Milner. Calculi for Synchrony and Asynchrony. TCS, 25(3):267-310, 1983.

[Milner 89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.

[Najm 87] E. Najm. A Verification Oriented Specification in LOTOS of the Transport Protocol. Dans [Rudin 87] 181-203.

[Park 81] D. Park. Concurrency and Automata on Infinite Sequences. Proc. 5th GI Conference, LNCS 104 (1981), 167-183.

[Shiratori 90] N. Shiratori, H. Kaminaga, K. Takahashi, S Noguchi. A Verification Method for LOTOS Specifications and its Applications. Dans [Brinksma 90] 59-70.

[Turner 88] K. Turner. Constraint-oriented Style in LOTOS. Dans proceedings of British Computing Society, Workshop on Formal Methods in standards, Dicot, 1988.

[van Glabbeek 86] R.J. van Glabbeek. Notes on CCS and CSP. Rapport Technique CS-R8624, Centrum voor Wiskunde en Informatica, Amsterdam, Pays-Bas, 1986.

[Vissers 88] C. Vissers, G. Scollo, M. van Sinderen. Architecture and Specification Style in Formal Distributed Systems. Dans [Aggarwal 89].

SECTION 2

[Aujla 88] S.S. Aujla. The Boyer-Moore Theorem-Prover and LOTOS. Dans [Turner 88] 169-184.

[Bochmann 89] G.v. Bochmann, O. Bellal. Test Result Analysis in Respect to Formal Specifications, Proc. 2nd Int. Workshop on Protocol Test Systems, Berlin, Oct.

1989, pp.272-294.

[Bochmann 90] G. v. Bochmann, Q. Gao,C. Wu. Distributed Implementation of LOTOS. Dans [Vuong 90] 133-146.

[Bochmann 90a] G.v. Bochmann, D. Desbiens, M. Dubuc. D. Ouimet, F. Saba. Test Result Analysis and Validation of Test Verdicts. À Paraître dans les Proceedings of the Workshop on Protocol Test Systems, McLean, Virginia (Oct. 1990).

[Bolognesi 89a] T. Bolognesi, M. Caneve. Equivalence Verification: Theory, Algorithms, and a Tool. Dans [van Eijk 1989] 303-326.

[Cheung 90] T.Y. Cheung, Y.C. Ye, X. Ye, C.Q. Wang. UO-GLOTOS: A Syntax/ System for Representing, Editing, and Translating Graphical LOTOS. In [Vuong 90] 31-41.

[de Jager 90] P. de Jager, W. Jonker, A. Wammes, J. Wester. An Interactive Programming Environment for LOTOS. Dans [Quemada 90].

[Franchi 90] P. Franchi-Zannettacci, A. Zarli. An Incremental and Graphical Structure-oriented Editor for G-LOTOS. Dans [Quemada 90].

[Garavel 90] H. Garavel, J. Sifakis. Compilation and Verification of LOTOS Specifications. Dans [Logrippo 90] 379-394.

[Guillemot 88] R. Guillemot, M. Haj-Hussein, L. Logrippo. Executing Large LOTOS Specifications. Dans [Aggarwal 88] 399-410.

[ISO 89b] ISO/IEC JTC1/SC21 N3253. G-LOTOS: A Graphical Syntax for LOTOS (1989).

[Logrippo 90b] L. Logrippo, The University of Ottawa LOTOS Toolkit. Dans [Quemada 90].

[Mañas 89] J.A. Mañas, T. de Miguel, H. van Thienen. The Implementation of a Specification Language for OSI Systems. Dans [van Eijk 89] 409-421.

[Nomura 90] S. Nomura, T. Hasegawa, T. Takizuka. A LOTOS Compiler and Process Synchronization Manager. Dans [Logrippo 90], 165-184

[Ohmaki 90] K. Ohmaki, K. Takahashi, K. Futatsugi. A LOTOS Simulator in OBJ. Dans [Quemada 90].

[Queiroz 90] J.Queiroz, A. Sehrouchni, P. Cunha, E. Jajm. PIL: A Tool for Pre-implementation of LOTOS. Dans[Quemada 90].

[Quemada 88] J. Quemada, S. Pavon, A. Fernandez. Transforming LOTOS specifications with LOLA, The Parametrized Expansion. Dans Turner 88] 45-54.

[Tretmans 89] J. Tretmans. HIPPO: A LOTOS Simulator. Dans [van Eijk 89] 391-396.

[van Eijk 89a] P.H.J. van Eijk. The Design of a Simulator Tool. Dans [van Eijk 90] 351-390.

[van Eijk 90] P. van Eijk, H. Eertink, Design of the LOTOSPHERE Symbolic LOTOS Simulator. Dans [Quemada 90].

SECTION 3

[Ajubi 89a] I. Ajubi, G. Scollo, M. van Sinderen. Formal Description of the OSI Session Layer: Introduction. Dans [van Eijk 89] 89-96.

[Ajubi 89b] I. Ajubi. Formal Description of the OSI Session Layer: Session Protocol. Dans [van Eijk 89], 153-210.

[Boumezbeur 91] R. Boumezbeur, L. Logrippo. Specification and Validation of Telephone Systems in LOTOS. Université d'Ottawa, Département d'informatique, à paraître.

[Brinksma 88] E. Brinksma. On the Design of Extended LOTOS. Thèse de doctorat, Université de Twente, Pays-Bas, 1988.

[Cam 90] R.C. Cam, S.T. Vuong. A Formal Specification, in LOTOS, of a Simplified Cellular Mobile Communication System. Dans [Vuong 90] 485-499.

[Faci 90] M. Faci, L. Logrippo, B. Stepien. Formal Specifications of Telephone Systems in LOTOS. Dans [Brinksma 90] 25-34.

[Faci 91] M. Faci, L. Logrippo, B. Stepien. Formal Specification of Telephone Systems in LOTOS: The Constraint-Oriented Approach. Computer Networks and ISDN Systems 21 (1991) 53-67.

[Fernandez 89] A. Fernandez, J. Quemada, L. Vidaller, C. Miguel. PRODAT - The Derivation of an Implementation from a LOTOS Specification. Dans [Aggarwal 89] 411-419.

[Freestone 89] D. Freestone, S. Aujla. Specifying ROSE in LOTOS. Dans: [Turner 89] 231-245.

[Haj-Hussein 91] M. Haj-Hussein, L. Logrippo. Specifying Distributed Algorithms in LOTOS. A paraître dans Proceedings of the Computer Networks Conference, Wroclaw, 1991

[Iglewski 91] M. Iglewski, A. Obaid. Specification of the IEEE 802.5 Token Ring LAN in LOTOS. Université du Québec à Hull, RR 91/01-2, 1991.

[ISO 89b] ISO/IEC TR 9571. LOTOS Description of the Session Service, 1989.

[ISO 89c] ISO/IEC TR 9572. LOTOS Description of the Session Protocol, 1989.

[ISO 90] ISO/IEC DTR 10023. Formal Description of ISO 8072 in LOTOS, 1990.

[ISO 90a] ISO/IEC JTC1/SC 6/WG 4 N651. Formal Description of ISO 8073 in LOTOS, 1990.

[Kouyzer 91] A.J. Kouyzer, A.K. van Boogaart. The LOTOS Framework for OSI Management. In; I. Krishan and W. Zimmer (eds.) Integrated Network Management, II. North-Holland, 147-156.

[Logrippo 90a] L. Logrippo, T. Melanchuck, R.J. DuWors. The Algebraic Specification Language LOTOS: An Industrial Experience. Dans: M. Moriconi (Ed.) Proceedings of the ACM SIGSOFT International Workshop on Formal Methods in Software Development (Napa, CA., 1990), 59-66.

[Gueraichi 89] D. Gueraichi. Derivation of Test Cases for LAP-B From a Formal Specification in LOTOS. Thèse de Maîtrise, Université d'Ottawa, Département d'informatique, 1989.

[Gueraichi 90] D. Gueraichi, L. Logrippo. Derivation of Test Cases for LAP-B from a LOTOS Specification. Dans [Vuong 90] 361-374.

[Navarro 91] J. Navarro, P. San Martin. Experience in the Development of an ISDN Layer 3 Service in LOTOS. Dans [Quemada 91].

[Pecheur 90] C. Pecheur. An Overview of the LOTOS Specification of Chorus V3. Report No. S.A.R.T. 89 - 03 -13, Université de Liège, B28, Département de systèmes et Automatique, Mai 1989.

[Quemada 89a] J. Quemada, A. Azcorra, A Constraint-Oriented Specification of Al's Node. Dans [van Eijk 89] 83-88.

[Quemada 89b] J. Quemada, F. Fournon. Connection Less Data Link Service. Dans [van Eijk 89] 221-226.

[Quemada 90] J. Quemada, A. Azcorra, D. Frutos. TIC: A Timed Calculus for LOTOS. Dans [Brinksma 90] 195-209.

[Scollo 89] G. Scollo. Formal Description of the OSI Session Layer: Transport Service. Dans [van Eijk 89] 97-116.

[Stefani 91] J.B. Stefani. Article Invité dans [Quemada 91].

[Taylor 91] C.R. Taylor, M. Gamble. The CCSDS Protocol Validation Programme Inter Agency Testing Using LOTOS. Dans [Quemada 91]

[van der Lagemaat 1988] J. v.d. Lagemaat, G. Scollo. On the Use of LOTOS for the Formal Description of a Transport Protocol. Dans [Turner 89] 247-261.

[van Hulzen 1990] W.H.P. van Hulzen, P.A.J. Tilanus, H. Zuidweg. LOTOS Extended with Clocks. Dans [Brinksma 90] 179-193.

[van Sinderen 89a] M. van Sinderen. Formal Description of the OSI Session Layer:

Session Service. Dans [van Eijk 89] 117-151.

[van Sinderen 89b] M. van Sinderen. The OSI Transaction Processing Service: A Formal Framework. Dans [van Eijk 89] 211-220.

[van Sinderen 89c] M. van Sinderen, I. Ajubi, F. Caneschi. The Application of LOTOS for the Formal Description of the ISO Session Layer. Dans [Turner 89] 263-277.

[van Sinderen 91] M. van Sinderen, I. Widya. On the Design and Formal Specification of a Transaction Processing Protocol. Dans [Quemada 91].

[Varadharajan 89] V. Varadharajan. Use of a Formal Description Technique in the Specification of Authentication Protocols. Computer Standards and Interfaces 9 (1989/90) 203-215.

Souheil Gallouzi est un diplômé de l'Université d'Ottawa avec un B.Sc et M.Sc. en Informatique. Présentement, il est membre du personnel scientifique de Recherche Bell-Northern où il travaille sur la vérification et le test des protocoles de communication. De plus, il poursuit ses études de doctorat a l'Université d'Ottawa. Ses intérêts de recherches incluent la théorie de validation et spécification des systèmes distribués.

Luigi Logrippo a une *laurea* de l'Université de Rome, une Maîtrise en informatique de l'Université du Manitoba, et un doctorat en informatique de l'Université de Waterloo. Depuis 1973 il est à l'Université d'Ottawa. Il dirige une équipe qui développe outils, applications, et concepts théoriques pour le LOTOS, dans le cadre du Telecommunications Research Institute of Ontario et du l'Institut canadien de recherches sur les télécommunications.

Abdellatif Obaid a un diplôme de troisième cycle de l'Université de Bordeaux, et un doctorat en génie électrique de l'Université d'Ottawa. Depuis quelques années il est à l'Université du Québec à Hull, où il est directeur du département d'informatique. Il s'intéresse au développement de la théorie, des outils, et des applications du LOTOS.