PRÉ-IMPLEMENTANDO ESPECIFICAÇÕES LOTOS

QUEIROZ, José[1], SERHROUCHNI, Ahmed[2], NAJM, Elie[2] e CUNHA, Paulo[1]

[1]Departamento de Informática da UFPE
Av. Prof. Luiz Freire, S/N - Edifício do CCEN
CP 7851 / Cidade Universitária - 50 739 Recife-PE

[2]INRIA
Domaine de Voluceau - Rocquencourt
BP 105 / 78 153 Le Chesnay Cedex / França

queiroz@di.ufpe.br ahmed@wagner.inria.fr najm@inria.inria.fr cunha@di.ufpe.br

Sumário

A pré-implementação de especificações LOTOS se baseia em um modelo/linguagem chamado PRIMOL. PRIMOL é um modelo formal para a transformação de especificações LOTOS em pré-implementações PRIMOL que estão mais próximas do que podemos considerar como implementações. PIL é uma ferramenta interativa que implementa o modelo PRIMOL. Depois de traduzir LOTOS para PRIMOL, a pré-implementação PRIMOL pode ser transformada em outras pré-implementações PRIMOL. PIL pode ser interfaceada com diversas ferramentas de exibição gráfica, verificação e testes. PIL executa em estações de trabalho Sun.

Abstract

The pre-implementation of LOTOS is based on a model/language called PRIMOL. PRIMOL is a formal model for the transformation of LOTOS specification in PRIMOL pre-implementations that are closer to we can consider implementations. PIL is an interacting tool that implements the PRIMOL model. After the LOTOS to PRIMOL translation, the PRIMOL pre-implementation can be transformed on PRIMOL pre-implementations. PIL can interfaces with several graphical exhibition, verification and testing tools. PIL runs on Sun workstations.

1 Introdução

A idéia de que um sistema deve ter a sua especificação realizada separadamente de sua implementação é amplamente defendida. Segundo essa abordagem um sistema deve ser especificado inteiramente em uma linguagem formal que apresente um nível muito alto de abstração. A implementação pode, então, ser derivada a partir dessa especificação.

A problemática de derivação de implementação é um assunto muito amplo que envolve, em particular, a transformação de objetos abstratos em objetos reais. É necessário considerar que os objetos abstratos são objetos formais enquanto os objetos reais são objetos com um baixo nível de formalização. Esses dois extremos fazem da derivação de implementação uma tarefa complexa. Conseqüentemente, é interessante decompor essa

etapa de desenvolvimento de maneira a obter uma etapa intermediária onde os objetos abstratos são derivados em objetos ainda formais e que estão mais próximos dos objetos reais.

A abordagem pré-implementação introduz um nível intermediário, chamado PRIMOL, que está colocado entre os níveis de especificação e de implementação, é o nível Pré-Implementação. PRIMOL é um modelo formal, apresentado na seção 2, que está mais próximo do que podemos considerar como implementações: as Máquinas de Estados Finitos Estendidas que Cooperam. Os elementos de PRIMOL são chamados de pré-implementações.

PRIMOL tem o mesmo domínio semântico de LOTOS (os sistemas de transição etiquetados) e é uma primeira etapa formal na trajetória que vai de LOTOS para a implementação. Sendo PRIMOL a controle finito e a um número finito de autômatas, se faz necessário definir um sub-conjunto de LOTOS que possa ser traduzido nessa linguagem.

A flexibilidade do modelo de transformações de PRIMOL em PRIMOL permite que diferentes tipos de aplicações, discutidas na seção 4, sejam alcançados:

- a simulação ou interpretação de especificações;
- a derivação dos testes baseados nas máquinas de estados finitos;
- a implementação em diferentes tipos de configurações (centralizadas ou distribuídas); e
- a prova de equivalência entre duas expressões de comportamento abertas.

A aceitação das Técnicas de Descrição Formal (TDFs) está ligada ao desenvolvimento de metodologias e de ferramentas que possam facilitar o uso dessas técnicas. LOTOS serviu de base para um grande número de trabalhos de pesquisa e de desenvolvimento para que uma implementação seja derivada formalmente de uma especificação [09,12,15,07]. As ferramentas para o suporte de LOTOS se estendem sobre os editores, analisadores, compiladores, simuladores e verificadores [11,06,18].

Na seção 3 será apresentada PIL (Pre-Implementation of LOTOS) que é uma ferramenta de Pré-Implementação de LOTOS baseada no modelo/linguagem PRIMOL.

PIL é uma ferramenta interativa que traduz especificações LOTOS em pré-implementações PRIMOL e permite transformar estas pré-implementações em outras que possam atender a aplicação desejada.

Complementando a apresentação de PIL se faz necessário a apresentação de um exemplo de tradução de LOTOS em PRIMOL e de transformações de PRIMOL em PRIMOL. Esse exemplo é dado na seção 5. Conclusões sobre o desenvolvimeto de PIL e direções de trabalhos futuros são apresentados na seção 6.

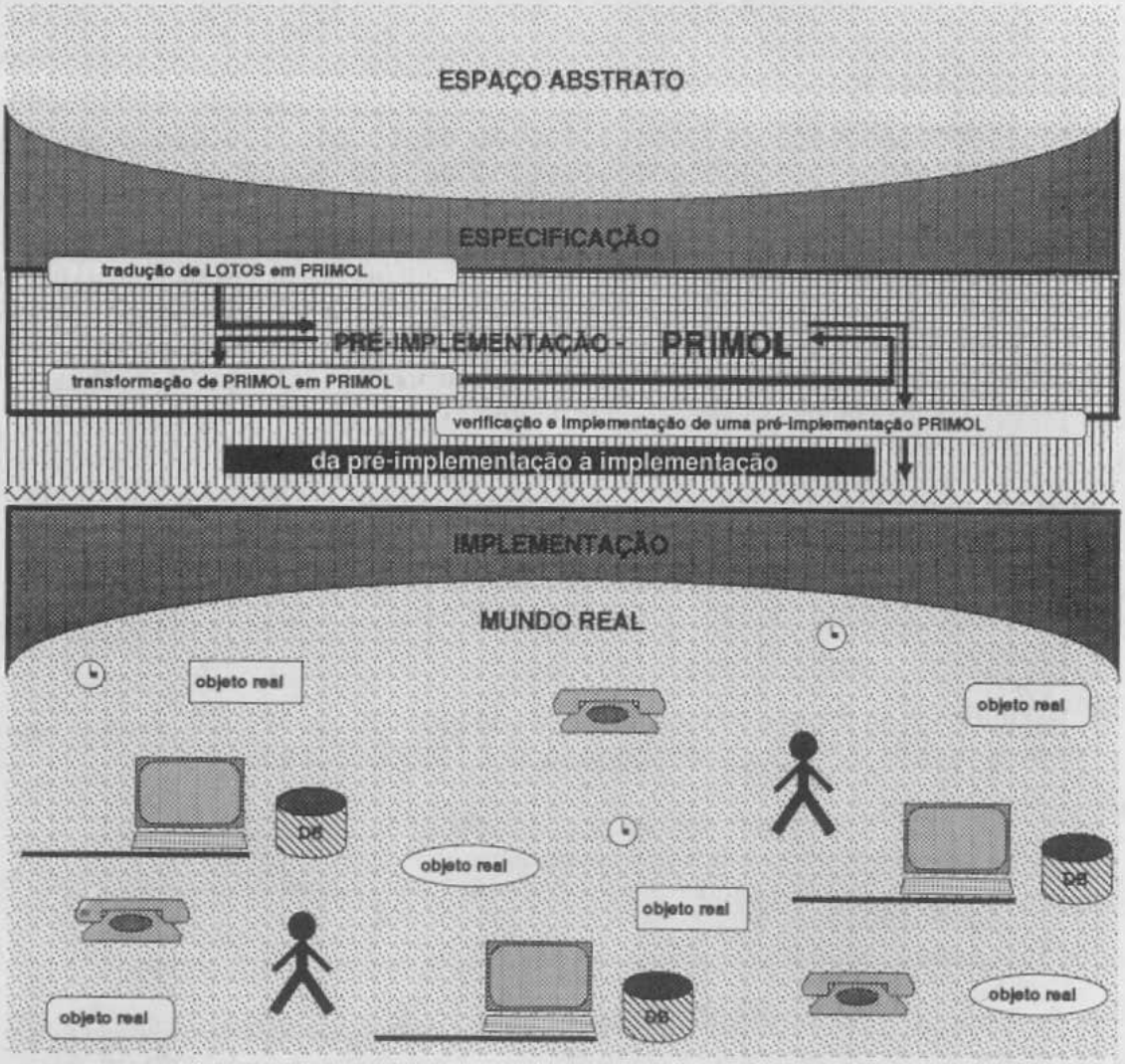

Figura 1 - Domínios do desenvolvimento de sistemas baseado em modelo de pré-implementação

2 O Modelo PRIMOL

O desenvolvimento de um sistema informático se articula em duas fases principais: a fase de especificação (concepção) e a fase de implementação. A fase de especificação é aquela onde o comportamento desejável de um sistema é descrito. O objetivo dessa fase é servir de base a implementação do sistema especificado. A implementação é a fase do sistema onde uma especificação é transformada em um conjunto de programas que serão implantados em máquinas reais.

Uma implementação pode ser derivada de uma especificação ou manualmente, pela utilização de metodologias descritas nos domínios de estudo da engenharia de software,

ou semi-automaticamente, através das derivações pelas transformações de especificação, ou automaticamente, pela compilação de uma especificação (linguagens automáticas).

O vazio formal que separa uma especificação de sistema da sua implementação pode ser reduzido pela adição de um mecanismo de transformação de especificações que produza outras especificações mais próximas de suas implementações. A figura 1 mostra o nível intermediário que foi inserido entre os níveis de especificação e de implementação, o nível de Pré-Implementação. Efetivamente a etapa de transformação de uma especificação em direção de uma implementação é reduzida pela transformação de uma pré-implementação (autômatos de estados finitos) em implementação.

Um mecanismo de derivação de implementação a partir de especificações LOTOS é apresentado. Esse mecanismo contribui para a realização de sistemas através da complementação do ciclo de desenvolvimento com uma tradução formal de especificações em em outras especificações (pré-implementações) baseadas num modelo orientado implementação. A adição desse mecanismo a um ambiente LOTOS aumenta a aplicabilidade dessa técnica formal, conseqüentemente essa linguagem de especificação pode ter uma utilização mais efetiva nos meios de desenvolvimento de sistemas distribuídos.

O nível intermediário introduzido serve para transformar especificações LOTOS em pré-implementações PRIMOL (**PR**e-**IM**plentation **O**f **L**OTOS) [13], ele é um modelo formal que tem como objetivo a transformação de especificações LOTOS em especificações que estejam mais próximas do que pode se considerar como sendo implementações: as Máquinas de Estados Finitos Estendidas que Cooperam.

Uma característica interessante de PRIMOL é a sua flexibilidade de representação que lhe permite servir como entrada para diversas ferramentas de verificação e teste.

A abordagem Pré-Implementação oferece um quadro transfomacional elegante. Com essa abordagem, uma especificação LOTOS é transformada em PRIMOL depois, pela aplicação do mecanismo de transformação de PRIMOL em PRIMOL, essa pré-implementação será transformada em novas pré-implementações. A pré-implementação resultante pode ser verificada e servir de base a uma implementação. Uma propriedade muito importante de PRIMOL é poder realizar transformações sobre especificações (LOTOS ou PRIMOL) preservando a semântica das especificações.

A implementação em um sistema real de um sistema especificado é realizada a partir de uma pré-implementação PRIMOL gerada pela fase de tradução em pré-implementação e depois pelas transformações (sucessivas) sobre essa pré-implementação que deve ter sido verificada e que servirá para a realização de testes de conformidade.

A tradução de LOTOS em PRIMOL se baseia sobre um mecanismo que mantém a estrutura hierarquica da árvore abstrata LOTOS e substitui os operadores LOTOS por suas formas de autômatas, um processo recursivo sendo traduzido em um único nó/autômata.

Um sub-conjunto de LOTOS é traduzível em PRIMOL. A retrição imposta às especificações LOTOS é de não se poder criar um número não limitado de processos. Essa restrição é garantida por um sub-conjunto sintático de LOTOS onde para todo par de processos mutualmente recursivos P e Q, nenhuma instanciação de Q na definição de P terá lugar no escopo de um operador de paralelismo, ou no escopo do argumento esquerdo de um operador de preempção ou de composição seqüencial.

PRIMOL é um modelo de Máquinas de Estados Finitos Estendidas Hierárquicas e Cooperantes. Mais precisamente, uma pré-implementação PRIMOL é uma hierarquia de elementos de pré-implementação, os elementos que compõem essa hierarquia, os nós, são

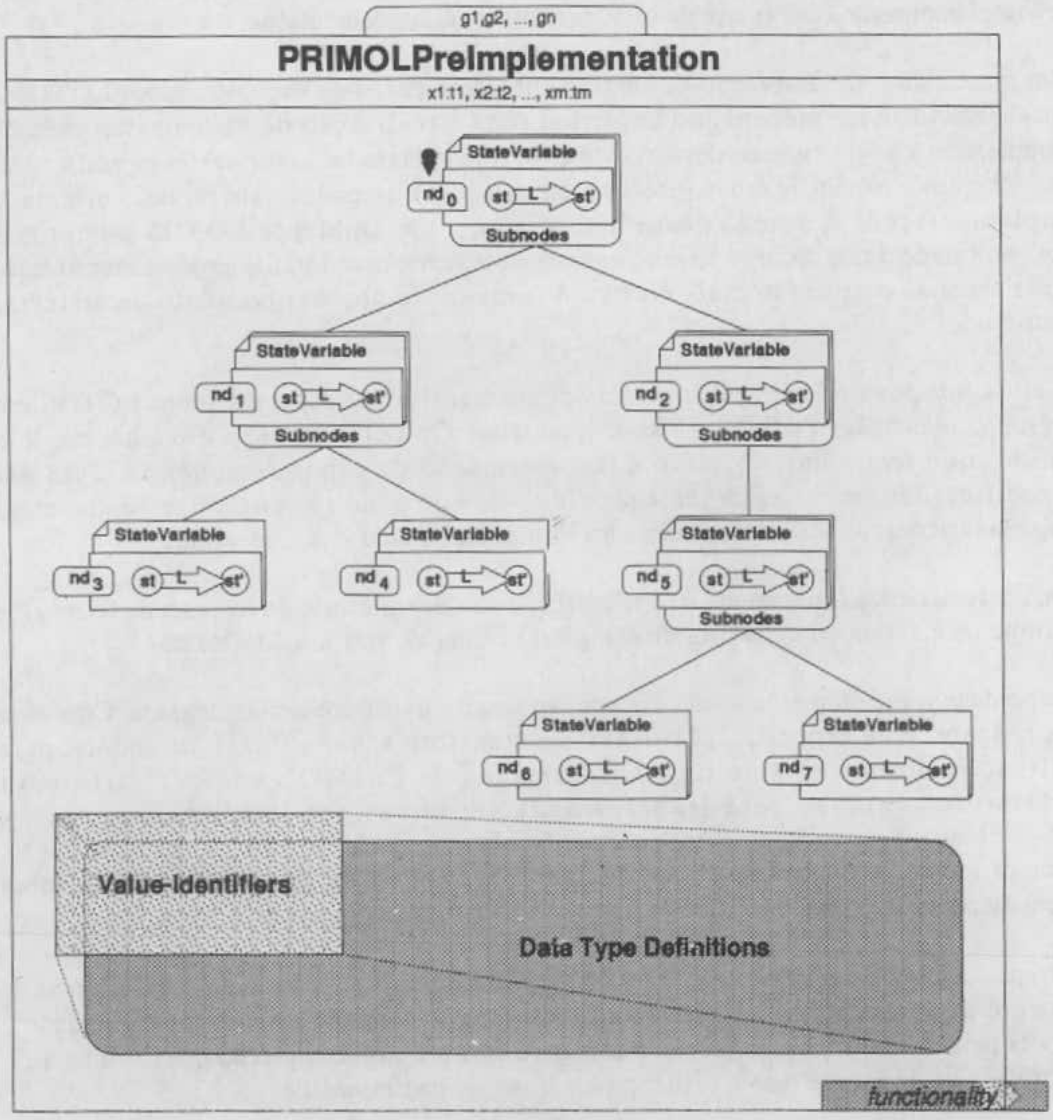

Figura 2 - Estrutura de uma Pré-implementação PRIMOL

um tipo especial de Máquina de Estados Finitos Estendida que compartilham uma coleção de definições de tipos de dados e um conjunto de *value-identifiers* tipados. Cada nó dessa hierarquia é responsável pela sincronização e pelo controle dos seus sub-nós. A figura 2 apresente a estrutura de uma pré-implementação PRIMOL dada. Os nós dessa pré-implementação são representados por uma hierarquia de icones.

A hierarquia em PRIMOL foi herdada da estrutura algébrica das expressões de comportamento LOTOS. Quando uma tradução de especificação LOTOS em pré-implementação PRIMOL é feita, cada nó é, em grosso modo, o resultado do mapeamento de um operador LOTOS da especificação fonte.

Uma vantagem da estrutura hierárquica é que ela formula um quadro de transformações de PRIMOL em PRIMOL, com preservação da semântica, simples e fácil: uma transformação é o resultado da aplicação de um conjunto de transformações elementares e uma transformação elementar de PRIMOL em PRIMOL é definida simplesmente como o resultado da fusão de um nó com um de seus sub-nós, resultando num novo nó.

3 PIL: Uma Ferramenta para a Pré-implementação de LOTOS

PIL é uma ferramenta interativa que traduz especificações LOTOS em pré-implementações PRIMOL. Depois de realizada a tradução as pré-implementações podem ser transformadas de PRIMOL em PRIMOL. PRIMOL é um modelo/linguage que é dotado de um mecanismo de transformações formal, simples e flexível que pode ser controlado pelo especificador.

Os módulos principais de PIL são:

- o analisador sintático e semântico da linguagem LOTOS como ele foi definido em [08];
- o módulo de verificação da propriedade de finitude de uma especificação, e o nivelamento dessa especificação;
- o tradutor de uma especificação LOTOS em pré-implementação PRIMOL; e
- o módulo de transformações de PRIMOL em PRIMOL.

PIL recebe como entrada uma especificação LOTOS e produz uma pré-implementação PRIMOL. As pré-implementações geradas pelo tradutor e pelas transformações podem ser visualizadas permitindo ao usuário seguir passo-a-passo a evolução de uma derivação de pré-implementação. Uma interface usuário ergonômica foi definida.

A especificação LOTOS fornecida como entrada para PIL é analisada sintática e semanticamente. Depois da análise, se essa especificação atende a propriedade de finitude então a função de nivelamento é aplicada à especificação LOTOS e uma especificação LOTOS URL (Unique Reference LOTOS) é gerada. Depois do nivelamento a tradução de uma especificação LOTOS em pré-implementação PRIMOL é realizada. Por fim, uma pré-implementação PRIMOL pode ser transformada (sucessivamente) em outras pré-implementações PRIMOL.

PIL suporta também os "pretty-printings" das especificações LOTOS, LOTOS URL e PRIMOL (numa forma compacta ou expandida). Uma outra função do "pretty-printer" é gerar arquivos de saída em diversos formatos que possam ser fornecidos como entrada para as ferramenta de exibição gráfica e de verificação, respectivamente AUTOGRAPH [16] e AUTO [19,10].

PIL foi desenvolvido na linguagem de programação "C" embaixo do sistema operacional UNIX 4.3 BSD, PIL executa na família de estações de trabalho Sun.

3.1 A Arquitetura de PIL

A arquitetura de PIL é apresentada em função dos seus principais componetes de software. A figura 3 ilustra essa aquitetura.

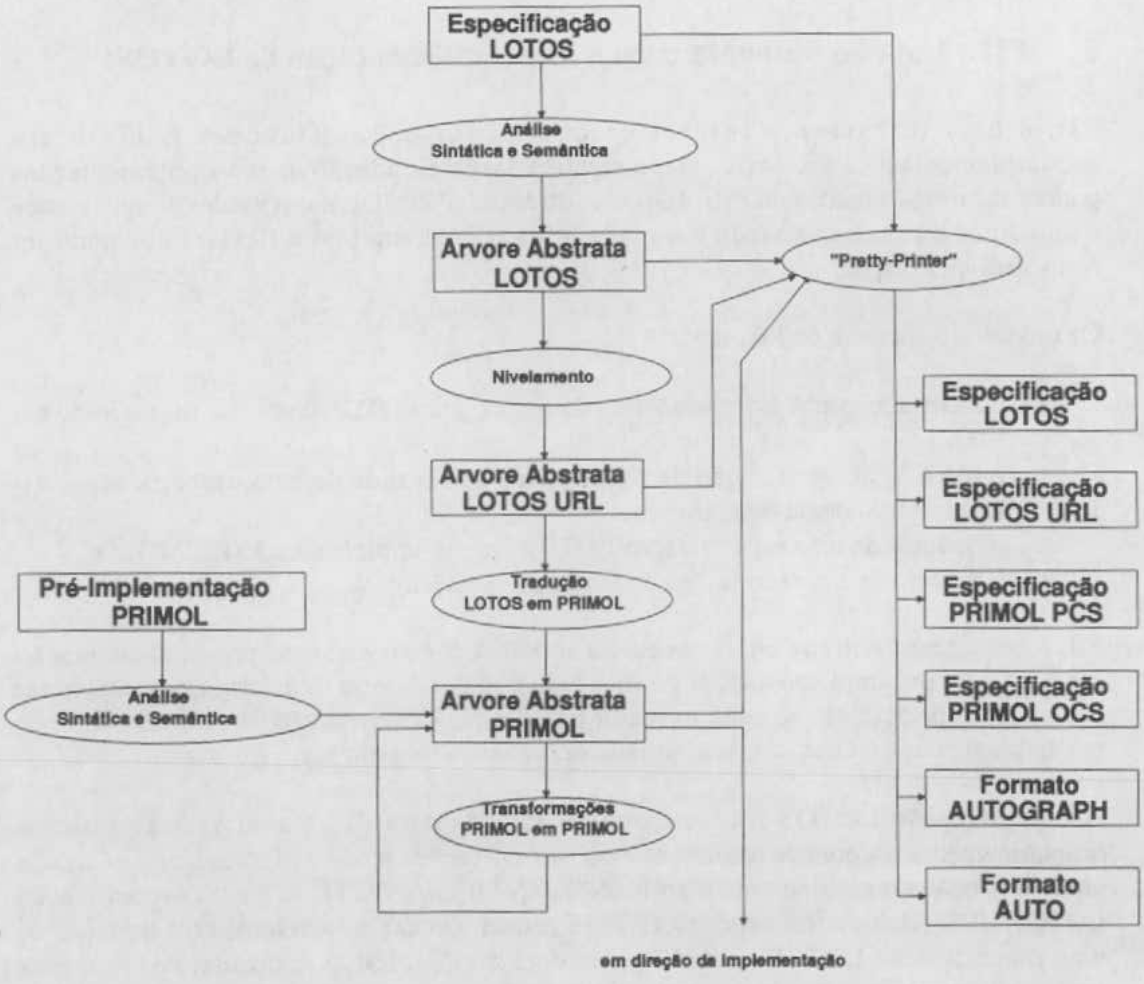

Figura 3 - Arquitetura de PIL

3.1.1 O Analisador de LOTOS

O analisador LOTOS é o primeiro componente de PIL, ele recebe como entrada uma especificação LOTOS, faz a análise sintática e de semântica estática e gera uma árvore de sintaxe abstrata da especificação LOTOS fornecida como entrada.

O analisador foi construído automaticamente pelo sistema gerador de compilador SYNTAX desenvolvido no INRIA [02]. Uma gramática de LOTOS, enriquecida com as funções de síntese de atributos é fornecida a esse gerador.

A análise sintática e de semântica estática de uma especificação LOTOS é realizada em mesmo tempo que às funções seguintes:

- construção da árvore de sintaxe abstrata que está muito próxima da sintaxe da especificação LOTOS;
- recuperação dos comentários;
- determinação da semântica estática parcial onde as instanciações de processos não são substituídas pelas suas expressões de comportamento correspondentes e as portas não são renomeadas; e
- ligação de uma ocorrência de referência de um identificador a uma ocorrência de definição correspondente.

3.1.2 Tradução de LOTOS em LOTOS URL

O tradutor de uma especificação LOTOS em uma especificação LOTOS URL (com referência única) é o segundo componente de PIL. O objetivo dessa fase é de preparar uma especificação LOTOS para que ela possa ser traduzida em PRIMOL.

Este componente toma como entrada a árvore abstrata de LOTOS e trata essa árvore gerando uma árvore de sintaxe abstrata URL que está completamente nivelada, isto é, todas as definições de processos e de tipos de dados encapsuladas são substituídas pelas definições desses processos e tipos de dados num único nível de especificação.

A geração da árvore de sintaxe abstrata é feita para as especificações LOTOS que verificam a propriedade de finitude de especificações.

O nivelamento de uma expressão de comportamento LOTOS consiste a substituir as instanciações de processos não-recursivos pelas expressões de comportamento que definem esses processos e a tratar as definições de processos recursivos segundo o método descrito por [17].

Na árvore de sintaxe abstrata de LOTOS URL todas as referências a identificadores são únicas.

3.1.3 Tradução de LOTOS (URL) em PRIMOL

A tradução de LOTOS (URL) em PRIMOL é a fase final da pré-compilação de uma especificação LOTOS. Nessa fase a árvore de sintaxe abstrata URL é recebida como

entrada e traduzida em pré-implementação PRIMOL. Essa tradução gera uma árvore de sintaxe abstrata de uma pré-implementação PRIMOL.

Essa tradução faz um mapeamento entre cada operador da árvore de comportamento LOTOS e um nó PRIMOL. Cada processo recursivo que foi tratado na fase de nivelamento, é traduzido em um só nó da pré-implementação PRIMOL.

3.1.4 Transformações de PRIMOL em PRIMOL

O módulo de transformação de PRIMOL em PRIMOL é o componente de PIL responsável das transformações sintáticas de uma pré-implementação (especificação) PRIMOL.

Esse componente de software toma como entrada a árvore de sintaxe abstrata de PRIMOL e aplica nela o mecanismo de transformações de PRIMOL em PRIMOL gerando uma árvore transformada de sintaxe abstrata de PRIMOL.

Uma pré-implementação PRIMOL é transformada, preservando a sua semântica [14], em outra pré-implementação PRIMOL pela fusão de um nó dado da especificação fonte com um de seus sub-nós.

O método de fusão de nós de uma pré-implementação foi descrito em [14]. O usuário pode, através da utilização do modo interativo de PIL, dirigir a transformação de uma pré-implementação PRIMOL. Para isso uma interface amigável foi construída.

3.2 Transformando Pré-implementações PRIMOL em PRIMOL

As transformações de uma pré-implementação PRIMOL podem ser dirigidas pelo especificador para que ele possa obter a pré-implementação desejada para a sua aplicação O mecanismo de condução de transformações se baseia na tipagem de nós e na execução interativa das transformações.

Uma pré-implementação é uma hierarquia de nós onde cada um desses nós podem ser marcados explicitamente com um tipo. Essa marcação deve ser efetuada pelo usuário e é ela quem dirigirá o mecanismo de transformações. Um nó ou sub-nó de uma pré-implementação PRIMOL pode ter um dos três tipos seguintes:

- **ordinário** - o **nó ordinário** é um nó da pré-implementação que pode absover um dos seus sub-nós ou que pode ser absorvido pelo seu nó pai;
- **controle** - o **nó de controle** é um nó da pré-implementação que pode absorver os seus sub-nós mas que não pode ser absorvido pelo seu nó pai; ou
- **genérico** - o **nó genérico** é um tipo de nó que não pode absorver os seus sub-nós nem mesmo ser absorvido pelo seu nó pai.

Inicialmente, os nós de uma pré-implementação são marcados implicitamente com os seguites tipos: o nó raiz com o tipo **controle** e todos os outros (sub-)nós com o tipo **ordinário**. O tipo (controle) do nó raiz deve ser constante.

3.2.1 Usando Nós Ordinários

Uma pré-implementação composta unicamente de nós ordinários (exceto o nó raiz) pode ser transformada em uma pré-implementação com um único nó. O resultado dessa transformação pode ser analisado na figura 4. Esse tipo de transformação pode ser utilizado para servir como base de uma implementação sobre um único processo (seqüencial).

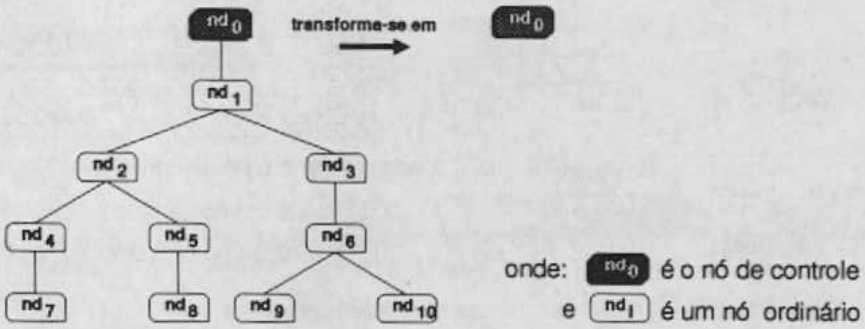

Figura 4 - Transformação de uma hierarquia de nós ordinários

3.2.2 Usando Nós de Controle

Uma pré-implementação é uma hierarquia de nós que pode ser transformada em uma outra hierarquia (vide figura 5) onde cada nó pode servir de base para a implementação sobre processos seqüenciais cooperantes. Essa abordagem pode ser utilizada como técnica para a distribuição de processos na implementação de um sistema.

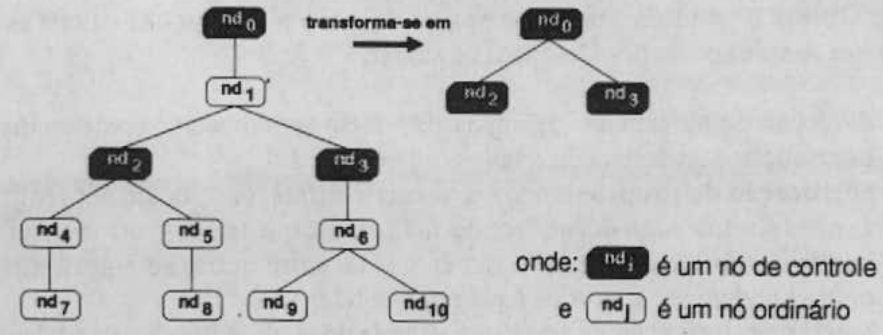

Figura 5 - Transformação de uma hierarquia marcada com nós de controle

3.2.3 Usando Nós Genéricos

A definição de nós genéricos em uma hierarquia de nós PRIMOL indica as sub-árvores dessa hierarquia sobre as quais o especificador não quer fazer nenhuma suposição sobre os seus comportamentos. Nesse caso é introduzido o conceito de expressão de

comportamento aberta, isto é, uma expressão de comportamento que utiliza *variáveis de ação* genéricas. O especificador pode se utlizar dessa técnica de marcação de nós para provar a equivalência de expressões de comportamento LOTOS (vide seção 4). A figura 6 apresenta uma pré-implementação marcada com nós genéricos que é transformada numa outra hierarquia onde os nós genéricos e seus sub-nós não são absorvidos.

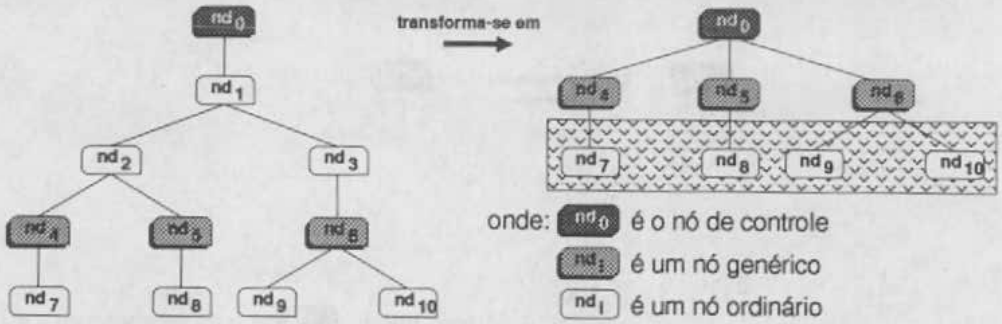

Figura 6 - Transformação de uma hierarquia marcada com nós genéricos

4 Aplicações

O modelo/linguagem PRIMOL possui uma estrutura de transformações flexível que permite a derivação de diferentes implementações a partir de uma pré-implementação dada. Por isso, diveros domínios de aplicação podem ser cobertos por essa estrutura de transformações. O mecanismo de transformações PRIMOL tem como objetivo produzir pré-implementações que possam ser verificadas e implementadas em sistemas (distribuídos) reais, e testar as implementações realizadas.

Uma pré-implementação pode ser transformada em uma outra pré-implementação que contém um único nó. Nesse caso essa pré-implementação terá uma única máquina de estados finitos estendida que pode servir de base a aplicações diversas. As mais importantes dessas aplicações são citadas a seguir:

- **detecção de blocagem** - os impasses podem ser detectados pela análise da parte de controle da máquina de estados finitos estendida;
- **verificação de propriedades** - diversas verificações podem ser realizadas pela expansão dos *value-identifiers* da máquina de estados finitos estendida se o autômato obtido é finito. Ferramentas [19 entre outras] e algoritmos [01 entre outros] podem ser aplicados à máquina obtida;
- **teorias de derivação de teste** - a máquina de estados finitos estendida pode servir de base para que a implementação de teorias de derivação de teste [04,05,03] e de teorias de testes baseadas nas máquinas de estados finitos estendidas sejam aplicadas;
- **implementação da especificação** - a estrutura de controle de PRIMOL permite que uma implementação eficaz possa ser realizada. Essa estrutura de controle serve de base para a geração de código ou para a simulação ou para uma interpretação eficaz.

Uma pré-implementação pode ser transformada em uma outra pré-implementação tendo mais de um nível hierarquico de nós. Essa pré-implementação com diversos níveis de nós é um tipo de pré-implementação onde os nós que a compõem sejam ou nós que controlam seus sub-nós ou sub-nós que são controlados por seus nós pais. Algumas das aplicações mais importantes são citadas a seguir:

- **mapeamento sobre máquinas reais** - os múltiplos níveis de uma pré-implementação podem ser mapeados sobre múltiplos processos onde os processos resultantes do mapeamento do nó raiz e dos nós intermediários são processos tipo ***kernel*** e os outros processos, resultantes do mapeamento dos nós terminais, são processos tipo ***operacional*** ;
- **prova de teorema** - a estrutura em múltiplos níveis de PRIMOL pode servir de base à prova de teorema. Essa prova pode ser implementada pela transformação das pré-implementações para a prova, em outras que são estruturadas em dois níveis. A prova se faz pela comparação dos nós raiz.

5 Um Exemplo

O exemplo que vai ser apresentado a seguir mostra como uma expressão de comportamento LOTOS é traduzida em PRIMOL e como essa pré-implementação PRIMOL pode ser transformada com vistas à prova de que o operador ***disable*** de LOTOS é associativo.

Consideremos B_1, B_2, B_3 três expressões de comportamento LOTOS quaisquer, pode-se mostrar que $(B_1 [> B_2) [> B_3$ é fortemente equivalente a $B_1 [> (B_2 [> B_3)$.

Essa prova pode ser realizada da seguinte maneira: inicialmente as expressões de comportamento LOTOS ($(B_1 [> B_2) [> B_3$ e $B_1 [> (B_2 [> B_3)$) são traduzidas para PRIMOL (figura 7 e figura 8, respectivamente); em seguida as pré-implementações geradas são transformadas em pré-implementações com dois níveis (figura 9) onde o primeiro nível corresponde ao nó de controle e o segundo nível correspode às expressões de comportamento B_1, B_2 e B_3; e, por fim, a prova é feita pela verificação da equivalência forte entre os nós de controle das duas pré-implementações.

Os nós de pré-implementação são aqueles que resultam da tradução de LOTOS em PRIMOL. Nessas pré-implementações os nós nd_{B1}, nd_{B2} e nd_{B3} são nós genéricos (com expressões de comportamento quaisquer) que não são descritos. Nas figuras 7 e 8, os nós nd_0 são os nós raiz dessas pré-implementações e os nós nd_1 são sub-nós respectivos de nd_0. A declaração **STATE-INITIALIZE** indica o estado inicial do autômata do nó e a declaração **SUBNODES-INITIALIZE** inicializa os sub-nós do nó.

É importante observar o autômata gerado pela tradução do operador ***disable*** (nós nd_0 e nd_1). A interpretação desse autômata é imediata. O estado inicial é aquele marcado pela seta preta. As *variáveis de ação* ***action(ndi)***, onde nd_i representa um dos sub-nós do nó observado, estão contidas em *expressões de ação abertas* e representam qualquer ação proveniente do nó nd_i. Os predicados sobre as ações **IS-IN {exit}** e **NOT-IN {exit}** verificam se a ação ***action(ndi)*** é uma operação ***exit*** ou não, respectivamente.

Os nós **nd0** e **nd1** são resposáveis pela "**sincronização**" e pelo "**controle**" dos seus sub-nós.

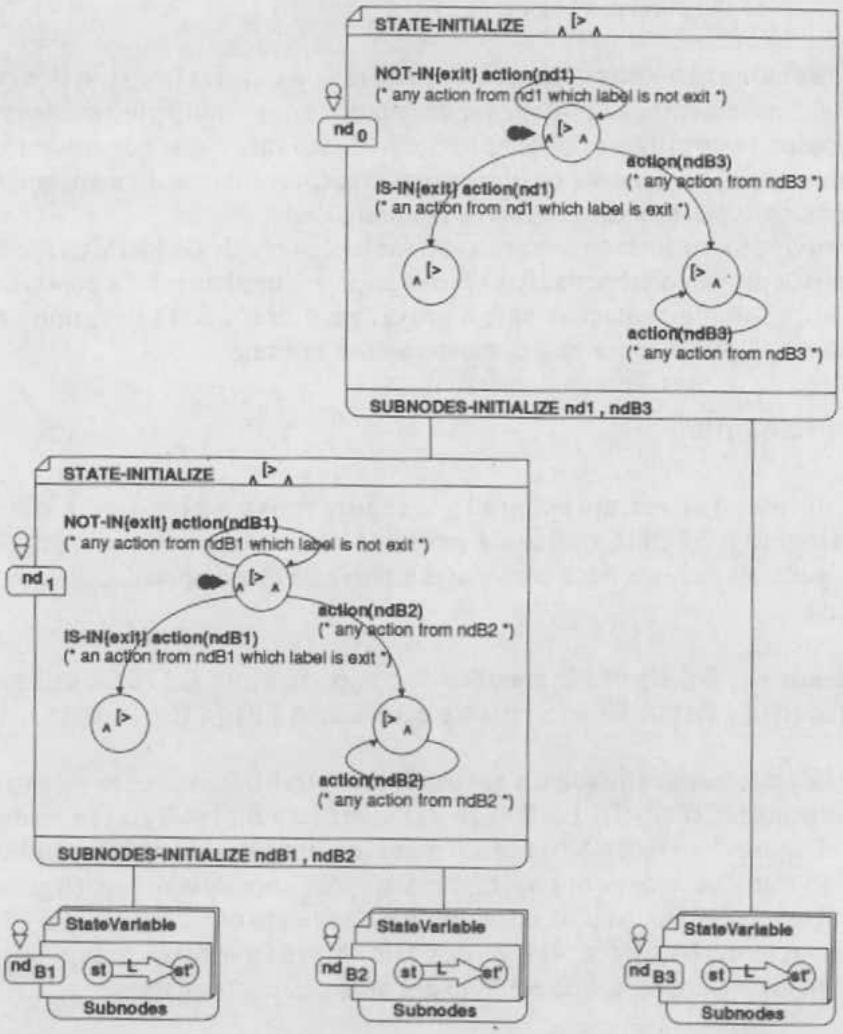

Figura 7 - Representação PRIMOL não transformada do comportamento LOTOS: $(B_1 [> B_2) [> B_3$

As transformações realizadas nessas duas pré-implementações são obtidas pelas fusões dos nós **nd0** e **nd1**, tanto na pré-implementação representada na figura 7 quanto naquela representada na figura 8. Depois de transformadas, as pré-implementações se apresentam em dois níveis: o nó raiz (de controle) e os seus sub-nós (genéricos).

As pré-implementações resultantes das transformações são comparadas para provar suas equivalências. A prova se dá pela comparação dos autômatas dos nós raiz. Verifica-se que existe uma equivalência forte entre os dois autômatas, então as expressões $(B_1 [> B_2) [> B_3$ e $B_1 [> (B_2 [> B_3)$ são equivalentes para quaisquer que sejam os nós genéricos nd_{B1}, nd_{B2} e nd_{B3}. Como os autômatas resultantes das transformações são fortemente equivalentes será apresentada, para efeito de simplicidade, unicamente a

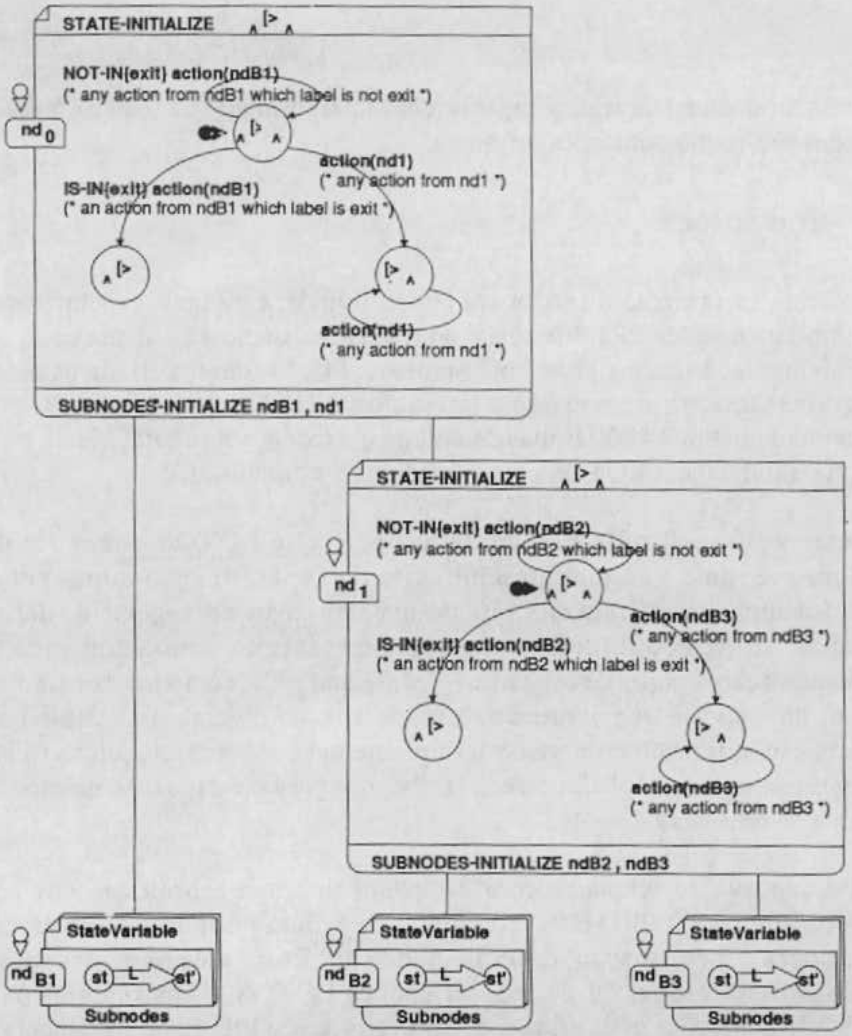

Figura 8 - Representação PRIMOL não transformada do comportamento LOTOS: B_1 [> (B_2 [> B_3)

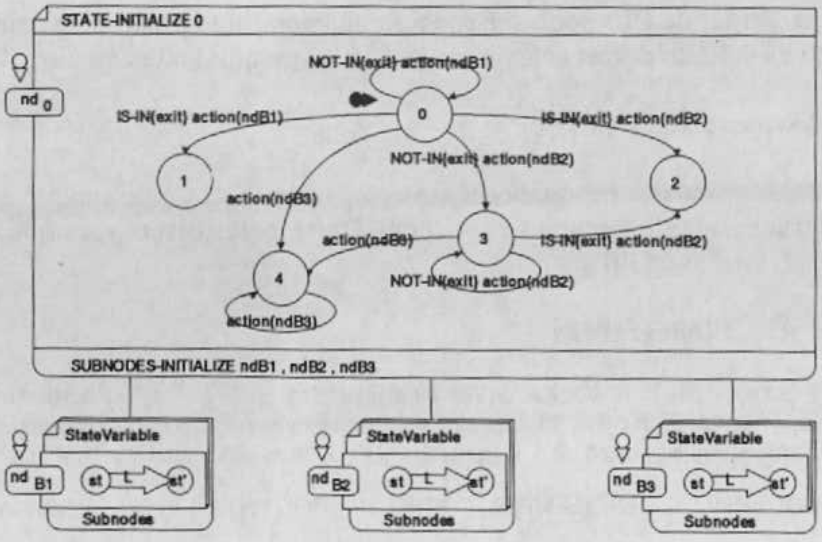

Figura 9 - Representação PRIMOL transformada dos comportamentos LOTOS:
(B_1 [> B_2) [> B_3 e B_1 [> (B_2 [> B_3)

pré-implementação da figura 9 que representa o resultado da transformação sobre qualquer uma das pré-implementações originais.

6 Conclusões

PIL é uma ferramenta de software construída para traduzir especificações LOTOS em pré-implementações PRIMOL visando reduzir a distância existente entre a especificação e a implementação de sistemas informáticos. PIL foi desenvolvida usando a lingugem de programação C sobre o sistema operacional UNIX. A versão atual de PIL comporta aproximadamente 24.000 linhas de código e executa sobre estações de trabalho SUN. PIL suporta a linguagem LOTOS como definida na norma [08].

O desenvolvimento de PIL contribui para que LOTOS possa ser difundida mais facilmente, uma vez que um ambiente de especificação amigável é oferecido ao especificador. Essa feramenta suporta um ciclo mais abrangente de desenvolvimento de sistemas, através do fornecimento de um mecanismo semi-automático de derivação de implementação a partir de especificações. Usando PIL como instrumento de especificação, tem-se um suporte que permite ao usuário ensaiar diversas pré-implementações de uma mesma especificação com vistas a implementações sobre máquinas (distribuídas) reais. Além dessa aplicação, PIL oferece facilidades para a verificação de especificações e para a prova de teoremas.

Outros aspectos relacionados com a implementação de especificações e com extensões do modelo/linguagem PRIMOL são objetos de estudo visando reduzir ainda mais o espaço que separa especificação de implementação. Entre eles podemos citar: a extesão do sub-conjunto traduzível de especificações LOTOS, as estratégias para obtenção de pré-implementações mais eficientes e a avaliação de PIL como ferramenta apropriada para a verificação.

A versão atual de PIL pode ser considerada como um ponto de partida promissor em direção da solução desses e de outros tópicos de pesquisa interessantes.

Agradecimentos

Os autores gostariam de agradecer aos colegas do INRIA pelo suporte que lhes foi dado. Um agradecimento especial para Jacky Barré pelas discussões que enriqueceram o desenvolvimento de PIL

Referencias Bibliográficas

[01] BOLOGNESI, T., SMOLKA, S. A. Fundamentals Results for the Verification of Observational Equivalence: A Survey. In: International Symposium on Protocol Specification, Testing and Verification, 8. Edited by S. Aggarwal and K. Sabnani. Atantic City, New Jersey-USA, 1988.

[02] BOULIIER, P., DESCHAMP, P., LORHO, B. The SYNTAX Reference Manual. INRIA, 1987.

[03] BRINSKMA, H. A Theory for the Derivation of Tests. In: International Symposium on Protocol Specification, Testing and Verification, 8. Edited by S. Aggarwal and K. Sabnani. Atantic City, New Jersey-USA, 1988.

[04] BRINSKMA, H., SCOLLO, G., STENNBERGEN, C. LOTOS Specification, their Implementation and their Tests. In: International Workshop on Protocol Specification, Testing and Verification, 6. North-Holland, Edited by H. Brinskma and G. V. Bochmann. Amsterdam, Holland, 1987.

[05] de MEER, J. Derivation of Test Scenarios Based on the Formal Specification Language LOTOS. In: International Workshop on Protocol Specification, Testing and Verification, 6. North-Holland, Edited by H. Brinskma and G. V. Bochmann. Amsterdam, Holland, 1987.

[06] EIJK, P. H. V. The Design of a Simulator Tool. In: The Formal Description Technique LOTOS: Results of ESPRIT/SEDOS Project. Elsevier Science Publishers, Edited by P. H. V. Eijk, C. A. Vissers, M. Diaz. Amsterdam, Holland, 1989. pp.351-90.

[07] GARAVEL, H., SIFAKIS, J. Compilation and Verification of LOTOS Specification. In: International Symposium on Protocol Specification, Testing and Verification, 10. Ottawa, Canadá, 1990.

[08] Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on the Temporal Ordering of Obsevational Behaviour. International Standard - IS 8807, 1988.

[09] KARJOTH, G. A LISP Based LOTOS Environment. In: International Conference on Formal Description Techniques - FORTE'88, 1. Elsevier Science Publishers, Edited by K.J.Turner. Amsterdam, Holland, 1988.

[10] MADELAINE, E. AUTO: A Verification Tool for Distributed Systems Using Reductions of Finite State Automata Networks. In: International Conference on Formal Description Techniques - FORTE'89, 2. Vancouver, Canadá, 1989.

[11] MADELAINE, E. de SIMONE, R. ECRINS, un Laboratoire de Preuve pour le Calcul de Processus. INRIA, RR 672. 1987.

[12] MAÑAS, J., de MIGUEL, T. From LOTOS to C. In: International Conference on Formal Description Techniques - FORTE'88, 1. Elsevier Science Publishers, Edited by K.J.Turner. Amsterdam, Holland, 1988.

[13] NAJM, E., QUEIROZ, J., SERHROUCHNI, A. The Pre-implementation and Verification of LOTOS. In: IFIP TC6 International Conference on Computer Networking. Budapeste, Hungria, Maio de 1990.

[14] QUEIROZ, J.A.M. Représentations Graphiques, Transformations et Pré-Implémentation de LOTOS. Université Pierre et Marie Curie - Paris 6. Novembro 90. (Tese de Dotourado)

[15] QUEMADA, J., PAVON, S., FERNANDEZ, A. Transforming LOTOS Specifications with LOLA: The Parameterized Expansion. In: International Conference on Formal Description Techniques - FORTE'88, 1. Elsevier Science Publishers, Edited by K.J.Turner. Amsterdam, Holland, 1988.

[16] ROY, V., de SIMONE, R. AUTO/AUTOGRAPH. In: Workshop on Computer Aided Verification. Rutgers, USA, Junho de 1990.

[17] SERHROUCHNI, A. PIL: Un Outil de Pré-Implémentation de LOTOS. Université Pierre et Marie Curie - Paris 6. Novembro 90. (Tese de Dotourado)

[18] TRETMANS, I. HIPPO - A LOTOS Simulator. In: The Formal Description Technique LOTOS: Results of ESPRIT/SEDOS Project. Elsevier Science Publishers, Edited by P. H. V. Eijk, C. A. Vissers, M. Diaz. Amsterdam, Holland, 1989. pp.391-408.

[19] VERGAMINI, D. Verification by Means of Observational Equivalence on Automata. INRIA, TR 501, 1986.