Validação de Especificações Formais de Sistemas Dependentes de Tempo Através de Simulação[+]

Roberto Milton Scheffel e Murilo Silva de Camargo
E-mail: {roberto, murilo}@inf.ufsc.br

Universidade Federal de Santa Catarina
Curso de Pós-Graduação em Ciência da Computação
Departamento de Informática e de Estatística
Campus Universitário Trindade - Caixa Postal 476
88040-900 Florianópolis - SC
Agosto, 1997

Resumo

Este artigo apresenta uma abordagem baseada em simulação para a validação de sistemas dependentes de tempo, especificados numa extensão temporal da linguagem LOTOS denominada RT-LOTOS (Real-Time LOTOS). A primeira parte do artigo apresenta as principais características de RT-LOTOS, com uma discussão de sua sintaxe e semântica. A seguir, é apresentada uma metodologia de validação de especificações RT-LOTOS, baseada em simulação. Para tal, o presente artigo apresenta uma ferramenta que suporta a simulação interativa e automática das especificações, apresentando suas funcionalidades. Para uma melhor compreensão, no final do artigo é apresentado um estudo de caso, baseado na especificação de um protocolo, usando a ferramenta para a validação dos seus requisitos.

Palavras-chave *- Especificação e Validação de Sistemas, Sistemas Distribuídos Dependentes de Tempo, RT-LOTOS, Simulação.*

Abstract

This work presents a simulation-based approach for the validation of time dependent systems, which are specified in a temporal extension of the LOTOS language called RT-LOTOS (Real-Time LOTOS). The first part of this paper presents the RT-LOTOS main characteristics, with a discussion of its syntax and semantics. Next, a validation methodology for RT-LOTOS specifications, using simulation, is presented. To accomplish this, this paper presents a tool supporting interactive and automatic simulation of the specifications. The functionalities of the tool are discussed , and for a better understanding, in the last part of this article, a case study based on a protocol specification is analyzed, using the tool for the validation of the requirements of the protocol.

Keywords *- Systems Specification and Validation, Time Dependent Distributed Systems, RT-LOTOS, Simulation.*

1. Introdução

As técnicas de descrição formal (TDF) têm sido cada vez mais utilizadas para a especificação de sistemas. Isto se deve ao fato de que estas apresentam vantagens flagrantes na especificação de sistemas complexos, de uma forma livre de ambigüidades. As TDFs oferecem uma fundamentação teórica que permite a construção de ferramentas computacionais para realizar a verificação e a validação dos sistemas especificados.

Uma das técnicas formais mais aceitas tem sido LOTOS, padronizada pela ISO (International Organization for Standardization) [1]. LOTOS tem encontrado vasto campo de aplicações, mas se destaca na especificação de aplicações distribuídas. Várias ferramentas de verificação e validação encontram-se disponíveis para esta TDF. Contudo, LOTOS possui uma limitação grave: não há definidos na linguagem mecanismos ou operadores que permitam a especificação e o tratamento do tempo e de restrições temporais. Isto é, LOTOS fornece os mecanismos apropriados para se expressar a ordenação temporal em que as ações devem ocorrer, mas não há como representar o tempo

[+] Este trabalho foi desenvolvido no contexto do projeto *Design de Aplicações Multimídia Distribuídas - DAMD* - ProTeM - fase III.

A ação interna `i` pode ter intervalo de tempo associado, porém a semântica de RT-LOTOS faz com que a ação `i` ocorra, necessariamente, dentro do intervalo de tempo a ela associado. Ações observáveis podem ser ocultadas, como na TDF LOTOS. Porém, ações ocultadas devem ocorrer no primeiro instante de tempo possível. Esta é a *propriedade de progressão máxima*.

As ações de RT-LOTOS são atômicas e instantâneas, e compreendem:

- as ações clássicas de LOTOS, que incluem as *ações observáveis*, `a`, pertencentes ao conjunto *Act*, a ação interna `i` e a ação de término com sucesso δ. Define-se $Act^{i} = Act \cup \{i\}$, e $Act^{\delta} = Act \cup \{\delta\}$.
- as ações específicas de RT-LOTOS, que são as *violações temporais*, `a*`, que pertencem ao conjunto *Act**. Existe uma bijeção entre *Act* e *Act**. Ou seja, para cada elemento $a \in Act$, existe uma ação $a^* \in Act^*$, e vice-versa.

O domínio de tempo $\mathbb{D}^-$, para a temporização das ações de Act^{i}, pode ser esparso ou denso, mas deve ser enumerável.

2.1. Sintaxe da Linguagem

As expressões de comportamento das especificações RT-LOTOS são geradas por uma sintaxe que é uma extensão direta da linguagem LOTOS. Esta sintaxe é apresentada a seguir:

```
E ::=   stop                                  ⇒ inação
    |   exit                                  ⇒ término com sucesso
    |   [t_min, t_max]a; E                    ⇒ prefixação - ação observável
    |   [t_min, t_max]i; E                    ⇒ prefixação - ação interna
    |   E [] E                                ⇒ escolha
    |   hide L in E                           ⇒ ocultação
    |   E |[L]| E                             ⇒ composição paralela
    |   E >> E                                ⇒ composição seqüencial
    |   E [> E                                ⇒ preempção
    |   E <L> {a_1:Q_1, ..., a_n:Q_n}         ⇒ preempção temporal
    |   P[a_1, a_2, ..., a_n]                 ⇒ instanciação de processo
```

2.2. Semântica Operacional de RT-LOTOS

A tabela I apresenta a semântica operacional de RT-LOTOS, no estilo SOS (Structured Operational Semantics) de Plotkin [11], e inclui regras de inferência para as ações clássicas, para as violações temporais e para a passagem do tempo.

2.3. Comentários sobre a semântica de RT-LOTOS

Como dito anteriormente, existe uma diferença entre a urgência da ação interna `i` e das ações ocultadas pelo operador `hide`. A ação interna `i` tem a garantia de ocorrer no seu intervalo, tornando-se urgente e incontrolável no limite superior do seu intervalo. Já as ações ocultadas, mesmo vistas pelo ambiente como ações internas normais, têm a *propriedade de progressão máxima*, ou seja, devem ocorrer no primeiro momento do seu intervalo.

Desta forma, a expressão de comportamento (`[5,10]i;P`) e a expressão de comportamento (`hide [a] in [5,10]a;P`) apresentam diferenças semânticas relevantes. Ambas são vistas, externamente, como uma ação interna seguida do comportamento de `P`. Porém, no aspecto temporal, têm comportamentos diferentes. No primeiro exemplo a ação `i` irá ocorrer no intervalo [5, 10], tornando-se urgente e incontrolável no instante 10. Já no segundo exemplo a ação `a`, pela propriedade da progressão máxima, irá ocorrer no instante 5, sendo vista externamente como a ação `i`.

O operador de preempção temporal também deve ser utilizado com certo cuidado, para não gerar comportamentos indesejados. O problema ocorre no uso da recursividade do lado esquerdo do operador de preempção temporal. As regras semânticas fazem com que a recursão "acumule" o operador de preempção temporal na expressão de comportamento subseqüente. Este comportamento pode ser evitado, bastando declarar um novo processo, recursivo, do lado esquerdo do operador de preempção temporal. Uma discussão mais detalhada sobre estes aspectos pode ser encontrada em [12]. Uma descrição mais detalhada da semântica operacional de RT-LOTOS pode ser encontrada em [6].

quantitativamente. Em outras palavras, não permitem expressar *quando* uma ação[1] deve ocorrer ou as restrições de tempo que são impostas a uma ação.

Constatada essa limitação de LOTOS, várias extensões desta TDF foram propostas com objetivo de sanar essa deficiência. Os primeiros trabalhos surgiram ainda na década de 80 [2], seguidos de outros que incluíam as mais diversas características, entre os quais pode-se citar [3, 4, 5, 6]. A evolução destas propostas fez com que um novo padrão de LOTOS fosse estudado, agora com suporte a aspectos temporais da especificação. Estes estudos deverão resultar num novo padrão denominado E-LOTOS (*Enhancement to LOTOS*) [7]. Para a especificação de sistemas dependentes de tempo, neste trabalho será utilizada a TDF RT-LOTOS (*Real-Time LOTOS*), descrita em [6]. Esta TDF é uma extensão de LOTOS, que permite expressar restrições temporais na ocorrência das ações das especificações, bem como especificar o tratamento dos casos em que as restrições temporais de uma ação não são satisfeitas.

O primeiro passo da concepção de um sistema é a descrição detalhada, em linguagem natural, dos seus requisitos relevantes. Um segundo passo consiste em fazer uma descrição formal destes requisitos. Esta descrição formal é refinada várias vezes, com maiores níveis de detalhamento e modularização. Neste processo é necessário garantir que os requisitos do sistema são satisfeitos, e que uma versão mais refinada do sistema continua apresentando a mesma funcionalidade da versão anterior, mais genérica. Para isto são necessárias ferramentas automatizadas, já que a avaliação manual das especificações torna-se inviável. Estas ferramentas devem permitir ao projetista verificar propriedades desejadas de sua especificação, bem como permitir estabelecer equivalências entre diferentes especificações.

Para especificação de sistemas dependentes de tempo, descritas com o uso da TDF RT-LOTOS, a verificação de propriedades pode ser feita utilizando-se a metodologia descrita em [8]. A ferramenta apresentada naquele trabalho faz a tradução das especificações RT-LOTOS para um modelo denominado *Autômato Temporizado*. As propriedades do sistema, descritas através da lógica temporal TCTL [9], são verificadas sobre o autômato temporizado, com o uso da ferramenta KRONOS [10].

Neste trabalho será abordado o problema da validação de sistemas dependentes de tempo através de simulação. Ao contrário da metodologia acima descrita, a simulação envolve uma grande interação com o usuário. A simulação interativa permite que o projetista possa acompanhar de perto a evolução da especificação, determinando a conformidade do comportamento com a descrição dos requisitos. Permite ainda identificar o ponto exato de possíveis comportamento indesejados. A simulação automática, através da geração de traços, mostra a evolução do sistema sob as condições determinadas pelo projetista. E uma exploração automática exaustiva dos possíveis traços de simulação gera uma árvore de execução. Esta árvore, quando finita, possibilita a verificação do comportamento total do sistema.

O presente artigo está organizado da seguinte forma. Para uma familiarização do leitor com a linguagem RT-LOTOS, a seção 2 apresenta a sua sintaxe e semântica. A seção 3 apresenta uma abordagem do problema da validação de especificações RT-LOTOS, através da utilização das funcionalidades de uma ferramenta de simulação. Como exemplo e estudo de caso um protocolo de comunicação simples é tratado. A seção 4 apresenta descrição informal desse protocolo, seus requisitos de qualidade, e sua especificação RT-LOTOS. Na seção 5, seus requisitos são validados através da simulação. Na seção 6 são feitas considerações finais sobre a ferramenta, é feita uma comparação com outros trabalhos relacionados e são apresentadas sugestões para continuação deste trabalho.

2. A Técnica de Descrição Formal RT-LOTOS

RT-LOTOS é uma extensão de LOTOS, que apresenta mecanismos para a representação e o tratamento de restrições de tempo na ocorrência das ações da especificação. Intervalos de tempo, na forma $[t_{min}, t_{max}]$, podem ser associados às ações. A semântica das ações temporizadas determina que uma ação só poderá ocorrer dentro do intervalo de tempo associado. Caso uma ação a não ocorra dentro do seu intervalo de tempo, no limite superior do intervalo ocorre uma ação interna especial a*, chamada de *violação temporal*. Violações temporais podem ser tratadas pelo novo operador de *preempção temporal*.

[1] O termo "porta" denota o nome da entidade sintética nas especificações que, quando executada, provoca a observação de um "evento" de mesmo nome. Neste trabalho comete-se um abuso de linguagem identificando com o termo "ação" tanto a porta quanto o evento associado a esta. Isto não traz maiores problemas, visto que os contextos onde cada um dos termos é usado são bem distintos, não havendo risco de confusão.

Inação	$\dfrac{-}{stop \xrightarrow{t} stop}$
Término com Sucesso	$\dfrac{-}{exit \xrightarrow{\delta} stop}$ $\qquad$ $\dfrac{-}{exit \xrightarrow{t} exit}$
Prefixação	$\dfrac{-}{[0,t]a;E \xrightarrow{a} E}$ $\qquad$ $\dfrac{-}{[0,0]a;E \xrightarrow{a^*} stop}\ (a \neq i)$ $\dfrac{-}{[0,t+s]a;E \xrightarrow{s} [0,t]a;E}$ $\qquad$ $\dfrac{-}{[t_1+s,t_2+s]a;E \xrightarrow{s} [t_1,t_2]a;E}$
Escolha	$\dfrac{E \xrightarrow{a} E'}{E[]F \xrightarrow{a} E' \quad F[]E \xrightarrow{a} E'}$ $\qquad$ $\dfrac{E \xrightarrow{a^*} E'}{E[]F \xrightarrow{a^*} E'[]F \quad F[]E \xrightarrow{a^*} F[]E'}$ $\dfrac{E \xrightarrow{t} E' \quad F \xrightarrow{t} F'}{E[]F \xrightarrow{t} E'[]F'}$
Ocultação	$\dfrac{E \xrightarrow{a} E' \quad (a \notin L)}{hide\ L\ in\ E \xrightarrow{a} hide\ L\ in\ E'}$ $\qquad$ $\dfrac{E \xrightarrow{a} E' \quad (a \in L)}{hide\ L\ in\ E \xrightarrow{i} hide\ L\ in\ E'}$ $\dfrac{E \xrightarrow{a^*} E' \quad (a \notin L)}{hide\ L\ in\ E \xrightarrow{a^*} hide\ L\ in\ E'}$ $\qquad$ $\dfrac{E \xrightarrow{a^*} E' \quad (a \in L)}{hide\ L\ in\ E \xrightarrow{i} hide\ L\ in\ E'}$ $\dfrac{E \xrightarrow{t} E' \quad (\forall a \in L \neg(E \xrightarrow{a}))}{hide\ L\ in\ E \xrightarrow{t} hide\ L\ in\ E'}$
Composição Paralela	$\dfrac{E \xrightarrow{a} E' \quad (a \notin L \cup \{\delta\})}{E[[L]]F \xrightarrow{a} E'[[L]]F \quad F[[L]]E \xrightarrow{a} F[[L]]E'}$ $\dfrac{E \xrightarrow{a^*} E'}{E[[L]]F \xrightarrow{a^*} E'[[L]]F \quad F[[L]]E \xrightarrow{a^*} F[[L]]E'}$ $\dfrac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F' \quad (a \in L \cup \{\delta\})}{E[[L]]F \xrightarrow{a} E'[[L]]F'}$ $\qquad$ $\dfrac{E \xrightarrow{t} E' \quad F \xrightarrow{t} F'}{E[[L]]F \xrightarrow{t} E'[[L]]F'}$
Composição Seqüencial	$\dfrac{E \xrightarrow{a} E' \quad (a \neq \delta)}{E >> F \xrightarrow{a} E' >> F}$ $\qquad$ $\dfrac{E \xrightarrow{\delta} E'}{E >> F \xrightarrow{i} F}$ $\dfrac{E \xrightarrow{a^*} E'}{E >> F \xrightarrow{a^*} E' >> F}$ $\qquad$ $\dfrac{E \xrightarrow{t} E' \quad \neg(E \xrightarrow{\delta})}{E >> F \xrightarrow{t} E' >> F}$
Preempção	$\dfrac{E \xrightarrow{a} E' \quad (a \neq \delta)}{E[> F \xrightarrow{a} E'[> F}$ $\qquad$ $\dfrac{F \xrightarrow{a} F'}{E[> F \xrightarrow{a} F'}$ $\dfrac{E \xrightarrow{\delta} E'}{E[> F \xrightarrow{\delta} E'}$ $\qquad$ $\dfrac{E \xrightarrow{a^*} E'}{E[> F \xrightarrow{a^*} E'[> F}$ $\dfrac{F \xrightarrow{a^*} F'}{E[> F \xrightarrow{a^*} E[> F}$ $\qquad$ $\dfrac{E \xrightarrow{t} E' \quad F \xrightarrow{t} F'}{E[> F \xrightarrow{t} E'[> F'}$
Preempção Temporal	$\dfrac{E \xrightarrow{a} E' \quad (a \neq \delta)}{E < L\{a_1:Q_1,\ldots,a_n:Q_n\} \xrightarrow{a} E' < L\{a_1:Q_1,\ldots,a_n:Q_n\}}$ $\qquad$ $\dfrac{E \xrightarrow{\delta} E'}{E < L\{a_1:Q_1,\ldots,a_n:Q_n\} \xrightarrow{\delta} E'}$ $\dfrac{E \xrightarrow{a^*} E' \quad (a \notin L)}{E < L\{a_1:Q_1,\ldots,a_n:Q_n\} \xrightarrow{a^*} E' < L\{a_1:Q_1,\ldots,a_n:Q_n\}}$ $\qquad$ $\dfrac{E \xrightarrow{a_i^*} E' \quad (a_i \in L)}{E < L\{a_1:Q_1,\ldots,a_n:Q_n\} \xrightarrow{i} Q_i}$ $\dfrac{E \xrightarrow{t} E'}{E < L\{a_1:Q_1,\ldots,a_n:Q_n\} \xrightarrow{t} E' < L\{a_1:Q_1,\ldots,a_n:Q_n\}}$
Instanciação	$\dfrac{E[a_1/g_1,\ldots,a_n/g_n] \xrightarrow{a} E', \quad P[g_1,\ldots,g_n] := E}{P[a_1,\ldots,a_n] \xrightarrow{a} E'}$ $\dfrac{E[a_1/g_1,\ldots,a_n/g_n] \xrightarrow{a^*} E', \quad P[g_1,\ldots,g_n] := E}{P[a_1,\ldots,a_n] \xrightarrow{a^*} E'}$ $\dfrac{E[a_1/g_1,\ldots,a_n/g_n] \xrightarrow{t} E', \quad P[g_1,\ldots,g_n] := E}{P[a_1,\ldots,a_n] \xrightarrow{t} E'}$

3. Validação de Especificações RT-LOTOS Através de Simulação.

Quanto se fala de verificação e validação de especificações formais, algumas divergências são encontradas na literatura, principalmente quando os dois termos são confrontados. Na continuidade deste trabalho, o termo *verificação* será utilizado para designar uma prova formal das propriedades de uma especificação através de um método, geralmente baseado na manipulação de axiomas. Já o termo *validação* denotará uma análise não exaustiva do comportamento de uma especificação por experimentos. Esta abordagem segue a linha de [13].

Para as especificações de sistemas utilizando RT-LOTOS, uma metodologia de verificação foi desenvolvida e apresentada em [8]. Esta metodologia utiliza uma técnica baseada na verificação de modelos (*model checking*). Para verificação de propriedades de uma especificação em RT-LOTOS, são escritas fórmulas de uma Lógica Temporal Tempo-Real chamada TCTL [9] para representar essas propriedades. Em seguida é feita uma tradução das especificações RT-LOTOS para um formalismo denominado *Grafo Temporizado* que é o modelo a ser verificado. Este formalismo é um grafo estendido, com relógios e condições lógicas associados aos estados, que permitem uma redução da complexidade do modelo. A tradução é feita pela ferramenta apresentada em [8]. Em seguida, uma ferramenta auxiliar, denominada KRONOS [10], é utilizada sobre o grafo temporizado, para verificar a satisfação ou não das fórmulas TCTL que representam as propriedades especificadas para o sistema.

A ferramenta apresentada neste artigo, denominada RTLS (RT-LOTOS Simulator), complementa a ferramenta descrita em [8], no sentido em que completa um ambiente computacional para análise de sistemas dependentes de tempo descritos em RT-LOTOS. A ferramenta RTLS apresenta basicamente duas funcionalidades, a serem utilizadas na validação de especificações RT-LOTOS, descritas a seguir:

Simulação Interativa: em cada estado da especificação, o usuário pode escolher as ações a disparar, entre as ações clássicas, as violações temporais e a passagem de tempo. A cada disparo é alcançado um novo estado. A ferramenta permite a marcação de estados da especificação. Pode-se retornar a estes estados posteriormente, e explorar outro traço de simulação. Permite-se visualizar o estado corrente da especificação, dado na forma de especificação RT-LOTOS. A simulação interativa é crucial para a depuração das especificações, já que permite acompanhar a evolução da especificação passo a passo.

Simulação Automática: permite que se gere um traço de simulação da especificação, com diferentes parâmetros. Estes parâmetros definem o momento de disparo das ações, a ocorrência ou não das violações temporais, o tamanho do traço de simulação a ser gerado, e também o tipo de traço a ser mostrado. Pode-se assim analisar a evolução da especificação sob diversas condições. A simulação automática também permite a geração de uma árvore de execução da especificação, de modo que todas as possíveis execuções até uma determinada profundidade de execução sejam mostradas. Tem-se assim uma representação de todos os comportamentos possíveis da especificação para determinadas faixas de tempo.

3.1. Implementação do Simulador

Para a simulação de uma especificação RT-LOTOS, a mesma é compilada, sendo traduzida para uma representação interna ao simulador. Quando da realização de ação de portas ou de passagem de tempo, é feita a atualização desta representação, de forma a expressar o novo estado da especificação. Estas transformações são implementadas no núcleo do simulador. As funcionalidades de simulação interativa e automática são implementadas em módulos independentes. A figura 3.1 mostra a arquitetura da ferramenta RTLS. Os três módulos principais são descritos a seguir.

3.1.1. Núcleo de Simulação

O núcleo de simulação faz a compilação da especificação, com sua tradução para a representação interna. Na construção do compilador de especificações foi utilizada a ferramenta SINTAX [14, 15]. Esta ferramenta, com base numa gramática que descreve a linguagem, constrói automaticamente os analisadores léxico e sintático. Gera também um programa fonte na linguagem C, que permite percorrer a árvore abstrata do arquivo fonte compilado. Com base nesta árvore abstrata o núcleo de simulação constrói a representação interna da especificação.

A representação interna da especificação é baseada diretamente na árvore abstrata. Consiste numa árvore, onde cada nó pode ser:

- um operador da linguagem, e os seus filhos os operandos;

- um processo básico de RT-LOTOS (`exit` e `stop`)
- uma instanciação de processo.

A construção desta árvore obedece a precedência de operadores de LOTOS, sendo que o operador de preempção temporal, introduzido em RT-LOTOS, tem a menor precedência, ou seja, as precedências são:

Prefixação > Escolha > Composição Paralela > Preempção > Composição Seqüencial > Ocultação > Preempção Temporal.

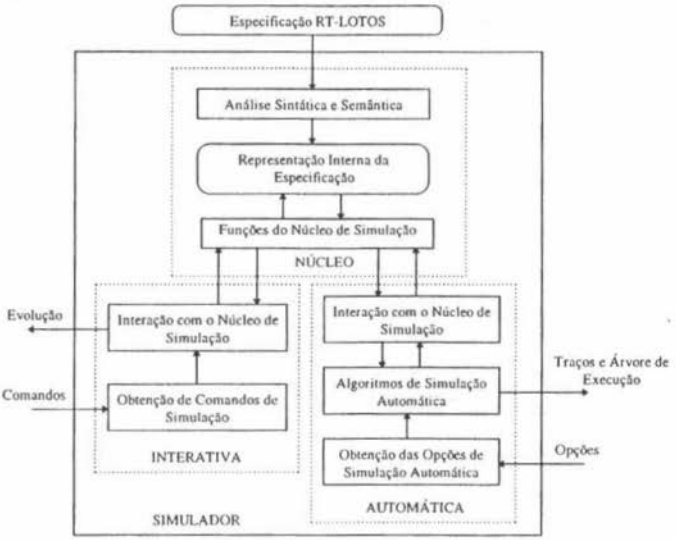

Figura 3.1 - Arquitetura interna da ferramenta RTLS

Para ilustrar a construção da árvore que representa uma expressão de comportamento RT-LOTOS, será considerado o seguinte processo:

```
Exemplo := a; exit ||| c; exit [] b; Q[a]
```

Ao final da análise da árvore abstrata do processo `Exemplo`, a representação interna da expressão de comportamento será uma árvore como a representada na figura 3.2.

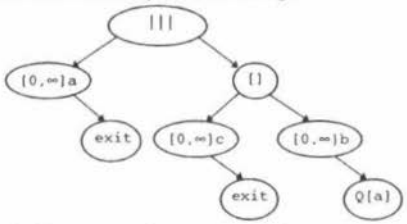

Figura 3.2 - Representação interna completa do processo `Exemplo`.

A representação interna é utilizada pela implementação das funções do núcleo do simulador. Antes da aplicação das funções, as instanciações que não forem guardadas (com no mínimo uma ação antecedendo-as) são substituídas recursivamente pela expressões de comportamento que as definem. Caso seja detectada uma recursão nesta substituição, a simulação é interrompida, pelo fato de haver expressões não guardadas na especificação, o que é proibido.

A manipulação da representação interna da especificação também é de responsabilidade do núcleo de simulação. Estas funções estão divididas em dois grupos: funções que manipulam a evolução da especificação pela ocorrência de ações, e funções que manipulam a evolução da especificação pela passagem do tempo.

De maneira bastante resumida, pode-se dizer que estas funções são responsáveis por determinar o conjunto da ações oferecidas pela especificação, determinar quais destas ações podem ocorrer num certo instante, e atualizar a representação interna em decorrência de uma ação. As funções determinam ainda o tempo mínimo até que haja mudança no conjunto de ações a ocorrer, o tempo máximo até que uma ação tenha que ocorrer, e atualizar a representação interna em decorrência da passagem do tempo.

As funções responsáveis pela evolução da especificação pela passagem do tempo simplesmente percorrem a árvore e atualizam os intervalos de tempo associados às ações. Já as funções responsáveis pela atualização da árvore, refletindo a ocorrência de ações clássicas e de violações temporais, têm um mecanismo mais complexo. Para cada ação possível, é associado o nó da árvore onde a mesma ocorre (para as ações da composição paralela, são armazenados todos os nós onde a ação ocorrerá simultaneamente). Cada nó, depois de atualização, determina como a ação que ocorreu é vista pelo operador acima, e notifica-o.

Por exemplo, a ocorrência da ação c na árvore da figura 3.2 substitui a prefixação pelo comportamento subseqüente (`exit`), e notifica o nó pai (escolha) que uma ação ocorreu na sub-árvore da esquerda. Este nó, de acordo com a sintaxe de RT-LOTOS, deve substituir toda a escolha pelo comportamento resultante do lado onde ocorreu a ação. Assim, a escolha será substituída pelo processo `exit`, e o nó da composição paralela será notificado que ocorreu a ação c do lado direito do operador. Como c não faz parte do conjunto de sincronização, e a composição paralela é o nó raiz, a atualização da árvore termina, e a representação interna do processo passa a ser a representada na figura 3.3.

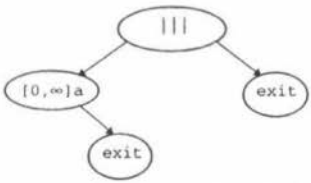

Figura 3.3 - Representação interna do processo `Exemplo` depois da ação c.

3.1.2. Simulação Interativa

Este módulo é bastante simples, mostrando para o usuário um menu com as ações que podem ocorrer na especificação, bem como o tempo que pode transcorrer. A partir disso, obtém-se do usuário a ação que deve ocorrer ou o tempo que deve passar, e esta informação é passada para o núcleo, que faz a atualização da representação interna da especificação. Os disparos de ações e passagem de tempo podem ser desfeitos por um comando *undo*.

Este módulo aceita ainda a marcação de estados, e o posterior retorno a estados previamente marcados. Mostra ainda para o usuário o estado atual da especificação, e a especificação original. Uma lista das ações e passagens de tempo é mantida por este módulo, para inspeção pelo usuário.

3.1.3. Simulação Automática

O módulo de simulação automática é responsável pela implementação dos algoritmos de geração automática de um *traço de simulação* e de uma *árvore de execução*.

A geração de um traço de simulação consiste na obtenção de algumas opções do usuário, que definem:

a) o instante, dentro do intervalo de tempo associado, em que a ação irá ocorrer;

b) a probabilidade da ocorrência de violações temporais;

c) a condição de parada da geração do traço de simulação; e

d) as ações que devem aparecer no traço de simulação.

Com base nos parâmetros acima, o algoritmo obtém do núcleo de simulação o conjunto de ações que podem ocorrer, bem como as possibilidades de passagem de tempo. É decidido então entre a ocorrência de ações e a passagem de tempo. O traço de simulação é atualizado a cada disparo de ação

ou passagem de tempo.

Os disparos são feitos de forma aleatória, respeitando as opções do usuário. O momento de disparo de uma ação, dentro do seu intervalo de tempo, obedece uma distribuição de probabilidade uniforme. Outras distribuições estão sendo estudadas para implementação posterior. A geração dos números aleatórios é feita baseada numa "semente", um número informado pelo usuário. A geração de dois traços baseados numa mesma semente, com as mesmas opções, faz com que os resultados sejam iguais. Assim, para diferentes testes, pode-se dar a mesma semente e diferentes opções.

Já a geração da árvore de execução tem o objetivo de montar uma representação de todas as possibilidades de evolução da especificação. A partir do estado inicial, o seguinte algoritmo de exploração é aplicado:

1. Para cada ação possível no estado, a mesma é disparada, e o estado resultante é explorado.
2. Se houver a possibilidade de passagem do tempo, é calculado o tempo máximo que pode passar no estado atual. Este cálculo obedece o seguinte critério: "*É o menor tempo para que se habilite uma nova ação, ou para que uma ação urgente ou uma violação temporal não permita a ocorrência de ações de passagem do tempo.*" Porém, o tempo máximo pode ser infinito, por exemplo, quando existem apenas ações observáveis não temporizadas. Neste caso, a exploração do estado pára depois do primeiro passo. Caso contrário, é feita a passagem do tempo máximo, e o estado resultante é explorado.

Este algoritmo, pelas características das funções do núcleo do simulador, garante que ações não temporizadas sejam disparadas uma vez, no primeiro instante em que são habilitadas. Garante também que ações temporizadas sejam disparadas no mínimo duas vezes, no limite inferior e no limite superior do intervalo de tempo associado.

Ações sincronizáveis na composição paralela são disparadas nos limites inferior e superior da interseção dos intervalos de sincronização. O *interleaving* de ações, resultante da semântica da composição paralela, pode fazer com que ações temporizadas sejam disparadas mais do que duas vezes.

A existência de ações não temporizadas não interfere na sincronização, uma vez que estas são disparadas logo que são habilitadas, não alterando os tempos das ações temporizadas. Porém, possíveis atrasos que possam levar à sincronizações, não possíveis sem estes atrasos, não são determinados. Em [12] encontra-se uma análise mais criteriosa das possibilidades de sincronização de ações temporizadas e da influência das ações não temporizadas nas sincronizações.

Na geração da árvore de execução, cada ramo da árvore é explorado até ser encontrada uma das três condições de parada:

1- chegada a um estado de *deadlock*, no qual a especificação não pode mais evoluir;

2- uma profundidade definida pelo usuário é alcançada;

3- um estado equivalente a outro já presente na árvore é encontrado. Se o estado estiver no ramo que leva ao estado sendo analisado, tem-se uma *recursão*. Senão, tem-se um estado equivalente ao atingido por outro traço de execução. Em ambos os casos, aponta-se o local onde o estado equivalente é explorado.

Assim, se todos os ramos da árvore de execução terminarem num *deadlock* ou apontando para um estado em outro lugar da árvores, todos os comportamentos possíveis da especificação estarão representados na árvore, podendo o usuário analisar a sua especificação em todos os aspectos comportamentais e temporais, validando os requisitos da mesma.

É evidente que comportamentos infinitos, que não apresentem recursão de comportamento (as transições acabam levando de volta para um estado já explorado), não podem ser analisados, nem mesmo utilizando a árvore de execução. Isto se deve ao fato de que, para comportamentos infinitos não recursivos, a árvore de execução também é infinita. Note-se que comportamentos infinitos recursivos têm uma representação finita do seu comportamento. Já para comportamentos infinitos não-recursivos não é possível gerar uma representação finita, o que inviabiliza sua análise.

A seguir, todas as funcionalidades da ferramenta serão descritas, de modo mais detalhado. A utilização de cada funcionalidade é analisada do ponto de vista da validação de sistemas dependentes de tempo descritos em RT-LOTOS. Um protótipo desta ferramenta está disponível para avaliação na Internet.

3.2. Simulação Interativa

A simulação interativa permite a validação da especificação através do acompanhamento, passo a

passo, da evolução do comportamento da mesma. Os disparos de todas as transições são controlados pelo projetista, permitindo assim um controle total da evolução da especificação. Este controle e acompanhamento permite que os pontos exatos da especificação onde ocorram possíveis erros sejam encontrados. Depois dos ajustes necessários, valida-se o novo comportamento da especificação, novamente por simulação. Este procedimento é repetido até que se obtenha o comportamento desejado.

Na simulação interativa, a ferramenta RTLS permite que o usuário escolha as ações da especificação que devem ocorrer, entre ações clássicas e violações temporais. Se não houver nenhuma ação urgente, o usuário também pode permitir a passagem de tempo. Os disparos de ações e a passagem de tempo podem ser desfeitos. A cada disparo de ação ou tempo, o estado da especificação é atualizado. O estado atual pode ser visualizado, na forma de uma especificação RT-LOTOS. Também a especificação original pode ser visualizada. O traço de disparo de ações, que levou a especificação até o estado atual, pode ser inspecionado. Em qualquer ponto da especificação, o estado atual pode ser marcado para posterior retorno, de modo a permitir a exploração de outros caminhos de simulação.

3.3. Simulação Automática

A simulação automática permite que se faça a exploração de um ou mais caminhos da especificação sem a intervenção do usuário. Permite que se automatizem alguns testes da especificação, de modo que o usuário não tenha que fazer a simulação manualmente.

Dois tipos de simulação automática são possíveis: a *geração de um traço de simulação*, ou a *geração de uma árvore de execução*. A geração de um traço é mais rápida, já que uma seqüência de disparos é gerada, sendo que sua análise fica a cargo do usuário.

Já a árvore de execução percorre todos os caminhos possíveis de execução para uma especificação, detectando recursividade. Esta funcionalidade faz uma exploração, a partir do estado inicial da especificação, de todos os traços de simulação possíveis. Cada ramo da árvore é explorado até se encontrar uma recursão, um *deadlock*, ou a profundidade especificada pelo usuário.

As duas formas de simulação automática são descritas com mais detalhes a seguir.

3.3.1. Geração de Traços

Para a geração de traços de ações da especificação, o usuário pode determinar parâmetros relacionados ao momento de ocorrência das ações temporizadas, à ocorrência ou não das violações temporais, à condição de parada da simulação, bem como ao traço a ser fornecido pelo simulador. Estas opções são descritas sucintamente a seguir.

Com relação ao momento de ocorrência de uma ação, o usuário tem a opção de fazer com que as ações ocorram no limite inferior do intervalo, no limite superior, ou num momento aleatório do intervalo, segundo uma distribuição de probabilidade uniforme. Pode-se assim determinar o tempo mínimo e o tempo máximo de evolução da especificação, bem como tempos intermediários aleatórios.

Também é oferecido o controle sobre a ocorrência das violações temporais. O usuário pode especificar se deseja que as violações temporais nunca ocorram, sempre ocorram ou ocorram aleatoriamente. Pode-se simular assim o funcionamento sem falhas da especificação, a falha total das sincronizações da especificação com seu ambiente, ou ainda o caso em que o ambiente que pode estar pronto ou não para sincronizar-se com as ações temporizadas.

A simulação automática, na geração dos traços de simulação, irá gerar uma seqüência de ações, até encontrar uma condição de parada. Esta condição pode ser um estado de *deadlock*, no qual nenhuma ação está pronta para ocorrer, nem existem ações que estão esperando para que o limite mínimo do seu intervalo seja alcançado. O usuário pode ainda especificar o comprimento do traço que deseja gerar, em número de disparos de ações. Pode ainda determinar que a simulação termine depois que um limite de tempo da especificação seja alcançado. Ou ainda pode dizer que deseja simular indefinidamente a especificação. Neste caso a simulação terminará quando for encontrado um *deadlock*, ou quando for estourado o limite de memória da máquina que estiver executando a simulação.

Para o acompanhamento do traço de simulação, o usuário pode ajustar o tipo de saída que deseja que o simulador forneça. Pode-se acompanhar todas as ações e as passagens de tempo, o que fornece uma visão completa da evolução da especificação. Outra opção é acompanhar somente as ações observáveis e a passagem do tempo, o que dá uma visão de como o comportamento da especificação é percebido pelo ambiente no qual a mesma está inserida. Uma terceira opção mostra somente as ações observáveis e o momento no qual elas ocorrem. Uma última opção permite acompanhar o traço de

apenas algumas ações. Esta opção é importante para analisar a periodicidade de uma ação, por exemplo.

3.3.2. Árvore de Execução

A árvore de execução permite representar, de uma forma simples, todos os comportamentos possíveis de uma especificação. Ao contrário da simulação interativa e da geração de traços, a árvore de execução permite uma análise exaustiva do comportamento.

A análise da árvore consiste em observar, para cada estado, as transições possíveis. Cada linha da árvore representa uma transição possível, ou o estado completo, quando o mesmo for equivalente a um estado já explorado, ou quando for um estado de *deadlock*. Em todas as linhas da árvore é indicado, entre parênteses, o tempo transcorrido desde o estado inicial da especificação. As linhas da árvore de execução podem ser apresentadas das seguintes formas:

Nº do estado - ação disparada (tempo): indica que, a partir do estado indicado pelo número, pode-se disparar a ação indicada. O estado resultante é explorado nas linhas seguintes da árvore.

Nº do estado - recursion detected - estado (tempo): indica que o estado é equivalente ao outro estado indicado, sendo que o mesmo encontra-se no traço de execução que leva do estado inicial da especificação ao estado atual.

Nº do estado - Analized elsewhere - estado (tempo): indica também que o estado é equivalente ao outro estado indicado, porém não há um traço de execução que leve de um estado para outro.

Nº do estado - Deadlock (tempo): indica que o estado é de *deadlock*, ou seja, não é possível o disparo de ações em nenhum momento futuro. A especificação pode somente deixar o tempo passar, indefinidamente.

Por exemplo, uma especificação que deveria ter um comportamento infinito, ou seja, livre de *deadlocks*, pode ter um traço de simulação em que um *deadlock* ocorre. Na simulação interativa, mesmo que o usuário de esforce ao máximo para cobrir todas as possibilidades de execução, pode acabar deixando de lado justamente o caminho que leve a um *deadlock*. Na geração da árvore de execução este problema não ocorre, pois todas as possibilidades de execução são examinadas.

4. Especificação de um Mecanismo de Sincronização Intra-Fluxo

Como estudo de caso e para uma breve demonstração da ferramenta de simulação, esta seção irá apresentar a especificação de um pequeno protocolo de comunicação. Este exemplo foi descrito inicialmente em [16] e verificado também em [17]. Aparece também em [8], que apresentou uma especificação e verificação de propriedades deste protocolo em RT-LOTOS. Outros exemplos podem ser encontrados em [12].

O protocolo de comunicação aqui descrito compõem-se de três entidades principais: *sender*, *receiver* e *service*. A validação se dará somente sobre a especificação de *service*, um vez que aí estão presentes as restrições temporais básicas do protocolo.

4.1. Descrição Informal do Protocolo

O envio de informações através da entidade *service* se dá através da interação das entidades *sender* e *receiver*, nos pontos de comunicação SS-SAP e SR-SAP. A figura abaixo ilustra o protocolo, de forma simplificada:

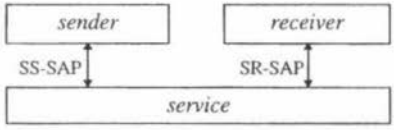

Figura 4.1 - O protocolo Tick-Tock

A entidade *service* aceita dados da entidade *sender*, e os transmite para a entidade *receiver*. As trocas de dados nos SAPs são feitas de forma atômica e instantânea. Assume-se que uma célula de dados está associada a cada interação nos SAPs.

Na figura 4.2 é mostrada a especificação formal do protocolo, escrita em RT-LOTOS. Após uma explanação sobre esta especificação, serão validadas as propriedades da mesma, descritas a seguir.

```
specification Service[SS_SAP, DELIVER, SR_SAP]
behaviour
            Isochronous[SS_SAP]
            |[SS_SAP]|
            (hide [DELIVER] in ((Trans_Dels[SS_SAP, DELIVER]
            |[DELIVER]|
            Spacing_Deliveries[DELIVER])
            |[DELIVER]|
            Imm_Accept[DELIVER, SR_SAP]))
where
    process Isochronous[SS_SAP]:=
            ([0]SS_SAP; [100]i; exit
             <SS_SAP>(SS_SAP: [100]i; exit)) >> Isochronous[SS_SAP]
    endproc
    process Trans_Dels[SS_SAP, DELIVER]:=
            SS_SAP; [50, 80]i; DELIVER; Trans_Dels[SS_SAP, DELIVER]
    endproc
    process Spacing_Deliveries[DELIVER]:=
            DELIVER; [90]i; Spacing_Deliveries[DELIVER]
    endproc
    process Imm_Accept[DELIVER, SR_SAP]:=
            (DELIVER;[0]SR_SAP; exit
             <SR_SAP>(SR_SAP: exit)) >> Imm_Accept[DELIVER, SR_SAP]
    endproc
endspec
```

Figura 4.2 - Especificação RT-LOTOS do protocolo Tick-Tock

No que tange aos aspectos temporais, a entidade *service* deverá satisfazer as seguintes condições:

- **Isocronismo:** As células de dados são aceitas pela entidade *service* em intervalos regulares, de π unidades de tempo. Somente uma célula pode ser enviada de cada vez Esta restrição está representada no processo *Isochronous*. A sincronização com a entidade *sender* (*SS_SAP*) é oferecida somente num instante. Após a sincronização, ou sua falha, é necessário um intervalo de π unidades de tempo. No exemplo acima, $\pi = 100$.
- **Atraso de Transmissão:** Após o envio pela entidade *sender*, a informação será oferecida para a entidade *receiver* dentro de um intervalo de tempo $[\tau_{min}, \tau_{max}]$. O processo *Trans_Dels* trata desta restrição da especificação. No exemplo acima, $\tau_{min} = 50$ e $\tau_{max} = 80$. Depois deste atraso, o processo *Trans_Dels* sinaliza que pode ser feita a entrega, através da ação *DELIVER*.
- **Espaços entre Entregas:** Existirá um atraso de, pelo menos, α unidades de tempo entre duas ofertas sucessivas de células para SR-SAP. O comportamento do processo *Spacing_Deliveries* garante o atendimento desta restrição. Após uma entrega, outra somente poderá ser feita depois de α unidades de tempo. No exemplo acima, $\alpha = 90$ unidades de tempo.
- **Aceitação Imediata:** Se uma célula de dados for oferecida para a entidade *receiver*, mas não for aceita imediatamente, esta célula será perdida.. O processo *Imm_Accept* irá garantir a restrição da aceitação imediata do dado recebido. Uma vez que o processo sincroniza na ação *DELIVER*, o que indica um novo dado está pronto para a entidade *receiver*, a mesma deverá sincronizar no mesmo instante. Após esta sincronização, o processo novamente fica esperando um novo sinal em *DELIVER*. Caso *receiver* não esteja pronto para receber o dado, o intervalo pontual de *SR_SAP* garante o dado não fique disponível por mais tempo. Neste caso o operador de preempção temporal garante que *Imm_Accept* volte a esperar um sinal em *DELIVER*.
- **Transmissão sem Falhas:** considera-se que a entidade *service* é um meio de transmissão confiável. Ou seja, uma vez que a célula de dados é recebida pela entidade, a mesma será oferecida para a entidade *receiver*, sendo perdida somente no caso da impossibilidade da aceitação imediata.

Os valores atribuídos às constantes da especificação (π, τ_{min}, τ_{max} e α) são os mesmo atribuídos por [17].

5. Simulação da Especificação

Nesta seção serão utilizadas as funcionalidades da ferramenta de simulação descrita na seção 3 para a validação das propriedades da especificação. A simulação interativa não será mostrada, pela dificuldade de demonstrá-la em texto. Serão utilizados traços de simulação e a árvore de execução.

As propriedades serão demonstradas na ordem em que foram apresentadas. Para isto serão utilizados traços de simulação. Para a geração dos traços de simulação será utilizada sempre a mesma semente. Ao final a árvore de execução será mostrada a árvore de execução gerada pela ferramenta, na qual as propriedades também podem ser mostradas.

5.1. Traços de Simulação

Isocronismo

Será gerado um traço de simulação, observando apenas a ação *SS_SAP*, que é a ação que recebe os dados da entidade *sender*. As violações podem ou não ocorrer. Ou seja, a entidade *sender* pode estar pronta para transmitir ou não. O traço de simulação será uma intercalação da ação *SS_SAP* e da sua violação temporal.

```
<0 - SS_SAP*>      <500 - SS_SAP*>     <1000 - SS_SAP*>
<100 - SS_SAP>     <600 - SS_SAP>      <1100 - SS_SAP>
<200 - SS_SAP*>    <700 - SS_SAP*>     <1200 - SS_SAP*>
<300 - SS_SAP*>    <800 - SS_SAP>      <1300 - SS_SAP*>
<400 - SS_SAP>     <900 - SS_SAP*>     <1400 - SS_SAP>
```

Analisando o traço acima, pode-se deduzir que a ação *SS_SAP* será oferecida à entidade *sender* em intervalos regulares de tempo, de 100 unidades de tempo. Ocorrendo ou não a transmissão, a entidade *sender* terá nova chance de transmitir somente depois de transcorridas 100 unidades de tempo. Valida-se assim o requisito de isocronismo.

Atraso na transmissão

Para a validação desta propriedades, dois traços serão gerados, observando-se a ação *DELIVER*, permitindo-se a ocorrência aleatória de violações temporais. O primeiro traço será gerado disparando-se as ações no limite inferior de seu intervalo de tempo.

```
<150 - i(DELIVER)>     <850 - i(DELIVER)>     <1650 - i(DELIVER)>
<450 - i(DELIVER)>     <1150 - i(DELIVER)>    <1750 - i(DELIVER)>
<650 - i(DELIVER)>     <1450 - i(DELIVER)>    <1850 - i(DELIVER)>
```

No segundo traço as ações serão disparadas no seu limite superior de tempo.

```
<180 - i(DELIVER)>     <880 - i(DELIVER)>     <1680 - i(DELIVER)>
<480 - i(DELIVER)>     <1180 - i(DELIVER)>    <1780 - i(DELIVER)>
<680 - i(DELIVER)>     <1480 - i(DELIVER)>    <1880 - i(DELIVER)>
```

Pela análise dos traços gerados acima, pode-se ver que o atraso mínimo na transmissão é de 50 unidades de tempo, pelo disparo das ações da especificação no limite inferior. Disparando as ações no limite superior, pode-se ver que o atraso máximo da ação DELIVER em relação ao recebimento do dado é de 80 unidades de tempo. Assim, pode-se considerar validado o requisito do atraso de transmissão. Note-se que não há violações temporais da ação *DELIVER*, uma vez que a mesma é urgente, por ser ocultada pelo operador *hide*.

Aceitação imediata

Nesta propriedade, o que se pretende verificar é: assim que o meio termina a transmissão (ação *DELIVER*), não há passagem de tempo antes que a mensagem seja entregue (*SR_SAP*) ou perdida (*SR_SAP**). Será gerado um traço, no qual ações irão ocorrer no limite superior de seu intervalo de tempo. O traço a ser observado é o das ações *DELIVER* e *SR_SAP*.

```
<180 - i(DELIVER)>     <680 - i(DELIVER)>     <1180 - i(DELIVER)>
<180 - SR_SAP>         <680 - SR_SAP*>        <1180 - SR_SAP*>
<480 - i(DELIVER)>     <880 - i(DELIVER)>     <1480 - i(DELIVER)>
<480 - SR_SAP>         <880 - SR_SAP*>        <1480 - SR_SAP*>
```

Pelo traço pode-se ver que entre o sinal de que uma transmissão terminou (ação *DELIVER*) e o oferecimento do dado à entidade *receiver* não há passagem de tempo. Isto valida o requisito da aceitação imediata. Por outro lado implica que se a entidade *receiver* não estiver pronta para receber o dado assim que o mesmo chegar, este será perdido.

Transmissão sem Falhas

Para a validação desta propriedade será gerado um traço de simulação, disparando as ações em momentos aleatórios de seus intervalos, permitindo violações temporais. O traço a ser avaliado é o das ações *SS_SAP* e *DELIVER*.

```
<0 - SS_SAP*>          <300 - SS_SAP*>        <600 - SS_SAP>
<100 - SS_SAP>         <400 - SS_SAP>         <675 - i(DELIVER)>
<157 - i(DELIVER)>     <470 - i(DELIVER)>     <700 - SS_SAP>
<200 - SS_SAP*>        <500 - SS_SAP*>        <765 - i(DELIVER)>
```

Analisando o traço acima, pode-se perceber que a cada vez que há um dado é recebido da entidade *sender* (ação *SS_SAP*), existe uma ação *DELIVER* num tempo compreendido no intervalo de atraso de transmissão. Valida-se assim requisito de transmissão sem falhas. Toda vez que um dado é colocado na entidade *service*, a mesma acusa a sua transmissão, para oferecer o dado para a entidade *receiver* (quando então o dado pode ser perdido por uma não recepção por parte da mesma).

5.2. Árvore de Execução

Depois de se validar os requisitos pela geração de traços de simulação, será gerada a árvore de execução. Pela natureza recursiva de todos os processos da especificação, pode-se deduzir que o comportamento da mesma seja infinito recursivo. Esta dedução mostra-se verdadeira, pois, com uma profundidade adequada, pode-se gerar a árvore de execução completa da especificação.

A seguir é apresentada uma parte da árvore de execução da especificação acima. Por motivos de espaço, a árvore toda não será mostrada. A árvore de execução completa da especificação apresenta 87 estados, sendo 69 estados diferentes. A partir da árvore completa as propriedades da especificação podem ser validadas.

```
1 - SS_SAP - (0)
 2 - time(50) - (0)
  3 - i (i) - (50)
   4 - i (DELIVER) - (50)
    5 - SR_SAP - (50)
     6 - i (EXIT) - (50)
      7 - time(50) - (50)
       8 - i (i) - (100)
        9 - i (EXIT) - (100)
         10 - SS_SAP - (100)
          11 - time(40) - (100)
           12 - i (i) - (140)
            13 - time(10) - (140)
             14 - Recursion detected - 3 (150)
         10 - i (SS_SAP*) - (100)
          15 - time(40) - (100)
           16 - i (i) - (140)
            17 - time(60) - (140)
             18 - i (i) - (200)
              19 - i (EXIT) - (200)
               20 - Recursion detected - 1 (200)
    5 - i (SR_SAP*) - (50)
     21 - i (EXIT) - (50)
      22 - Analized elsewhere - 7 (50)
```

A análise da árvore de execução é um pouco mais complicada do que a análise dos traços de simulação. Esta dificuldade é resultante da representação não linear do comportamento da especificação. Porém alguma propriedades são mais fáceis de se validar, como a ocorrência ou não de *deadlocks*. A árvore completa da especificação da figura 4.2 não apresenta *deadlocks*.

Se a ocorrência ou não de *deadlocks* é facilmente verificada na árvore de execução, há outras propriedades que são mais fáceis de se verificar através de traços. Tome-se, por exemplo, a transmissão sem erros. Esta propriedade exige que, depois da ação *SS_SAP*, sempre ocorra uma ação *DELIVER*. E que depois da violação temporal de *SS_SAP* (*SS_SAP**) não ocorra *DELIVER*. Para validar esta propriedade, é necessário encontrar todas as ocorrências de *SS_SAP* e ver que existe um *DELIVER* no ramo da árvore que segue, antes de outra ocorrência de *SS_SAP* ou de sua violação temporal. E para cada ocorrência da violação temporal de *SS_SAP* (*SS_SAP**), deve-se verificar que não existe uma ocorrência de *DELIVER* no ramo abaixo, antes de uma nova ocorrência de *SS_SAP*.

Todas as propriedades validadas com os traços de simulação podem ser validadas utilizando-se a árvore de execução, porém com mais dificuldade. Métodos automáticos de validação de propriedades, baseados na árvore de execução, estão sendo estudados para implementação futura na ferramenta RTLS.

6. Conclusões e Trabalhos Futuros

Neste trabalho apresentou-se uma abordagem de validação de sistemas dependentes do tempo especificados em uma extensão temporal de LOTOS chamada RT-LOTOS, através da ferramenta de simulação RTLS. Esta ferramenta apresenta funcionalidades de simulação interativa e simulação automática. A simulação interativa permite um acompanhamento detalhado da especificação. É bastante útil para depuração de especificações. A simulação automática apresenta as opções de se gerar um traço de simulação da especificação, com diversas opções de acompanhamento. Também permite a geração de uma árvore de execução, que tenta cobrir todas as possibilidades de evolução da especificação.

As funcionalidades da ferramenta foram descritas, após o que foi feita a validação de uma especificação de um protocolo de comunicação simples. Para tal foram utilizados traços de simulação e a geração da árvore de execução completa da especificação.

O tratamento de exemplos tem mostrado que a ferramenta é útil na validação de certas características das especificações, tais como o oferecimento de certas ações em determinados

instantes, ou o funcionamento da especificação de acordo com o esperado pelo seu projetista. A simulação interativa pode ser utilizada para detectar a causa de comportamentos indesejados. A geração de traços de simulação, com combinações de opções, pode avaliar como a especificação se comporta sob certas circunstâncias (se o ambiente nunca estiver pronto para sincronizar-se, por exemplo). Também permite acompanhar os instantes de ocorrência de algumas ações somente.

Já a árvore de execução permite que se teste se a especificação tem uma representação finita de seu comportamento. Ou mesmo se a especificação é livre de *deadlocks*. Porém apresenta o problema da explosão de estados para especificações complexas, o que pode inviabilizar a sua utilização.

Por outro lado, a ferramenta suporta apenas a parte básica da TDF RT-LOTOS. A parte de dados não foi implementada, podendo assim apresentar limitações para sua utilização em especificações que necessitem a parte de dados.

Como continuação deste trabalho, para incrementar as funcionalidades do simulador, pretende-se implementar as seguintes funcionalidades e características:

- Definição de um *menu* de funções de distribuição de probabilidade para o momento da ocorrência das ações no seu intervalo de tempo. Neste *menu* está prevista a possibilidade de escolha e configuração das principais funções de distribuição de probabilidade e também a definição de outras pelos usuários.
- Permitir que o usuário defina probabilidades (ou distribuições) para a sincronização do ambiente com a especificação. A partir disto poderia-se calcular, por exemplo, a probabilidade da especificação falhar em decorrência de um falta do ambiente.
- Geração de gráficos de ocorrência de ações no tempo. Esta funcionalidade está baseada na apresentada em [18] para uma outra extensão temporizada de LOTOS homônima à nossa.
- Inclusão de algumas verificações durante a geração da árvore de execução. Tais verificações podem incluir análise de alcançabilidade, probabilidade de falhas, avaliação de desempenho, entre outras.
- Composição da especificação com processos teste, que expressem propriedades na forma de uma especificação RT-LOTOS.

A ferramenta apresentada neste trabalho, juntamente com a ferramenta apresentada em [8], tem seu desenvolvimento inserido no projeto ProTeM III intitulado DAMD - Design de Aplicações Multimídia Distribuídas. A atual situação dessa ferramenta é apresentada na figura 6.1.

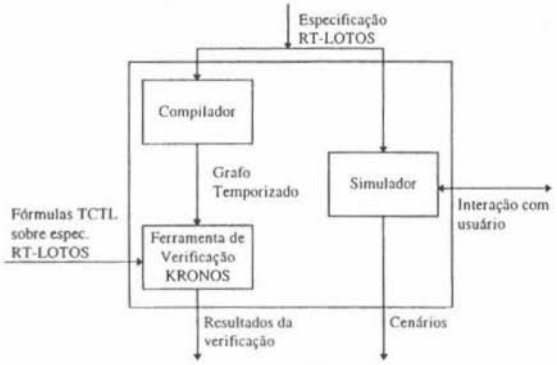

Figura 6.1 - Ferramentas de verificação e validação de especificações RT-LOTOS

Deve-se ressaltar que existem outras ferramentas com funcionalidades próximas àquelas apresentadas aqui para outras extensões temporais de LOTOS. Contudo, nossa abordagem faz parte de um contexto que inclui desde o desenvolvimento de uma linguagem extensão temporal de LOTOS até as ferramentas baseadas nela. A conclusão deste trabalho deve encerrar todo um ciclo que começou pela definição da linguagem [19, 20], definiu as abordagens para verificação de sistemas especificados na linguagem [21] e as ferramentas para verificação [22] e validação de sistemas para a linguagem RT-LOTOS que foi apresentada neste trabalho.

Entre outras ferramentas para simulação de especificações de sistemas dependentes de tempo, pode-se citar a apresentada no anexo C de [23]. Esta ferramenta utiliza a extensão temporal de LOTOS apresentada em [5, 24], denominada TE-LOTOS (Time Extended LOTOS). A simulação das especificações com restrições de tempo é feita através da tradução das especificações TE-LOTOS para LOTOS, representando os intervalos de tempo associados às ações através do oferecimento de uma ação especial *time*, com valores associados representando o intervalo de tempo. A seguir é utilizada a ferramenta de simulação para LOTOS denominada LOLA. A tradução de TE-LOTOS para LOTOS implica em algumas restrições, fazendo com que algumas características temporais de TE-LOTOS não possam ser utilizadas, como o operador *Wait* e domínios de tempo densos.

Um outro exemplo é a ferramenta apresentada em [18] utiliza uma extensão temporal de LOTOS, também denominada RT-LOTOS, a qual apresenta características diferentes das apresentada neste trabalho. Tal ferramenta gera uma série de gráficos para a análise do comportamento da especificação em relação ao tempo. A geração de gráficos está prevista para a implementação na continuidade deste trabalho.

Recentemente, foi divulgada uma *Draft Version* da linguagem E-LOTOS (*Enhancement to LOTOS*) [7] que incorpora nessa nova linguagem todos os elementos necessários para representação de processos dependentes do tempo, além de outros elementos que enriqueceram muito essa nova linguagem. E-LOTOS está em processo de padronização por um comitê especial da ISO e já se encontra em *status* de *Draft Proposal* . Tendo em vista esses avanços, já iniciamos um trabalho de adequação e portagem das ferramentas para E-LOTOS.

7. Referências Bibliográficas

[1] Information Processing Systems - Open Systems Interconnection - *LOTOS - A formal description technique based on the temporal ordering of observational behaviour*. IS8807, 1988.

[2] Quemada, J.; Fernandez, A.: *Introduction of Quantitative Relative Time into LOTOS*. In Protocol Specification, Testing and Verification VII, p. 105-121, 1987.

[3] Azcorra, A. S.: *Formal Modeling of Synchronous Systems*. Ph.D. Thesis, Dpto. de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Madrid, Espanha, 1990.

[4] Regan, T.: *Multimidia in Temporal LOTOS*. In IFIP - Protocol Specification, Testing and Verification, XIII (C-16), Dantine, A.; Leduc, G.; Wolper, P. (editors), Elsevier Science Publishers B. V. (North-Holland), p. 127-143, 1993.

[5] Leduc, G.; Léonard, L.: *A Formal Definition of Time in LOTOS*. Research Report, Université de Liège, Institut d'Electricité Montefiori, Bélgica, 1994.

[6] Camargo, M. S. de: *Tornando a Linguagem LOTOS Apta para Especificar Sistemas Dependentes do Tempo*. Tese de Doutorado LCMI/UFSC, Florianópolis, 1995.

[7] *Working Draft on Enhancement to LOTOS*, ISO/IEC JTC1/SC21/WG7, Project WI 1.21.20.2.3, January, 1997, Juan Quemada Ed.

[8] Martins, R. F.: *Verificação de Sistemas Dependentes de Tempo a partir de Especificações escritas em RT-LOTOS*. Dissertação de Mestrado LCMI/UFSC, Florianópolis, 1996.

[9] Alur, R.; Henzinger, T. A.: *Logics and Models for Real Time: A Survey*. Proceedings of the REX Workshop, Mook, The Netherlands, June, 1991, Lecture Notes in Computer Science No.600, Springer-Verlag, 1992.

[10] Olivero, A.; Yovine, S.: *KRONOS: A Tool for Verifying Real-Time Systems - User's Guide and Reference Manual*. Montbonnot Soint Martin, França, agosto 1993.

[11] Plotkin, G. D.: *A Structural Approach to Operational Semantics*, Report DAIMI-FN19, Computer Science Dept., Århus University, Dinamarca, 1981.

[12] Scheffel, R. M.: *Um Simulador para Validação de Sistemas Dependentes de Tempo Descritos em RT-LOTOS*. Dissertação de Mestrado CPGCC/UFSC, Florianópolis, 1997.

[13] Kirkwood, C. E.: *Verification of LOTOS Specifications using Term Rewriting Techniques*. Submitted for the Degree of Doctor of Philosophy. Department of Computing Science, University of Glasgow, June, 1994.

[14] Boullier, P.; Deschamp, P.: *Le Système SYNTAX: Manuel d'Utilisation et de Mise en Oeuvre sous UNIX*. Project Languages et Traducteurs, INRIA, Setembro, 1988.

[15] Boullier, P.; Deschamp, P.: *SYNTAX: User Commands and C Library Functions*. Project

Languages et Traducteurs, INRIA, Junho, 1989.

[16] Leduc, G.; Léonard, L.; Danthine, A.: *The Tick-Tock case study for the assessment of timed FDTs*. In the ISO95 transport service with multimidia support on HSLAN's and B-ISDN, 1994.

[17] Daws, C.; Olivero, A.; Yovine, S.: *Verifying ET-LOTOS programs with KRONOS*. In Proceedings of the FORTE'94, Berne, Switzerland, October, 1994.

[18] Courtiat, J. P.; Oliveira, R. D. de: *On RT-LOTOS and its application to the formal design of multimedia protocols*. In Annals of Telecommunications, vol 50, no. 11-1, p. 888-906, 1995.

[19] Courtiat, J.P. and de Camargo, M.S. and Saïdouni, D.E: *RT_LOTOS: LOTOS Temporisé pour la Spécification de Systèmes Temps Réel*, CFIP'93 - Ingénierie des Protocoles, Dssouli, R. e Bochmann, G. von (Editeurs), HERMES, Paris, 1993

[20] Camargo, M.S. de; Farines, J.-M.: *Tornando LOTOS Apta para Especificar Sistemas Tempo-Real*, Anais do 12 Simpósio Brasileiro de Redes de Computadores, Curitiba, maio, 1994.

[21] Camargo, M.S. de; Farines, J.-M.: *Uma abordagem para especificação e verificação de sistemas dependentes do tempo*, Anais do IX Simpósio Brasileiro de Engenharia de Software, Recife, Outubro de 1995.

[22] Martins, R.F.; Camargo, M.S. de; Farines, J.-M.: *Uma ferramenta para auxílio no processo de verificação de especificações em RT-LOTOS*, Anais do X Simpósio Brasileiro de Engenharia de Software, São Carlos, Outubro de 1996.

[23] Pavón, S.; Larrabeiti, D.; Rabay, G.: *Lotos Laboratory - User Manual (version 3R6)*, LOLA/N5/V10, Departamento de Ingenieria Telematica, Universidad Politécnica de Madrid, Madrid - Espanha, 1995.

[24] Leduc, G.; Leonard, L.; Frutos, D. de; Llana, L.; Miguel, C.; Quemada, J. e Rabay, G.: *Belgian-Spanish Proposal for a Time Extended LOTOS*. In J. Quemada, editor "Working Draft on Enhancements to LOTOS", ISI/IEC JTC1/ SC21/WG1. October, 1994.