Tornando a Linguagem LOTOS Apta para Especificar Sistemas Dependentes do Tempo

Murilo Silva de Camargo

27 de Janeiro de 1995

Universidade Federal de Santa Catarina Programa de Pós-Graduação de Engenharia Elétrica

Tornando a Linguagem LOTOS Apta para Especificar Sistemas Dependentes do Tempo

Tese submetida à Universidade Federal de Santa Catarina para obtenção do grau de Doutor em Engenharia, área Engenharia Elétrica, área de concentração Sistemas de Informação.

Murilo Silva de Camargo

Florianópolis, 27 de Janeiro de 1995.

TORNANDO A LINGUAGEM LOTOS APTA PARA ESPECIFICAR SISTEMAS DEPENDENTES DO TEMPO

Murilo Silva de Camargo

ESTA TESE FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE

DOUTOR EM ENGENHARIA

ÁREA ENGENHARIA ELÉTRICA, ÁREA DE CONCENTRAÇÃO SISTEMAS DE INFORMAÇÃO, E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO.

JEAN-MARIE FARINES, Dr.Ing.
(Orientador)

ENIO VALMOR KASSICK, Dr.
(Coordenador do Curso)

BANCA EXAMINADORA

JEAN-MARIE FARINES, Dr.Ing.
(Presidente)

WANDERLEY LOPES DE SOUZA, Dr.Ing.
(Relator)

JONIDA SILVA FRAGA, Dr.

JOSÉ ANTÔNIO MONTEIRO DE QUEIROZ, Dr.

MAURÍCIO FERREIRA MAGALHÃES, Dr.Ing.

ROSVELTER JOÃO COELHO DA COSTA, Dr.

Em memória de minha filha Isis

Isinha, você passou tão pouco tempo conosco
e nos deixou esse amor tão grande!

Já sei que você está no céu e que é o nosso anjinho,
mas essa saudade continua a doer imensamente.

Se soubesse que você partiria tão cedo,
eu não teria deixado que esse trabalho
roubasse tanto tempo do nosso convívio...

Para Clarice, com amor

Agradecimentos

Agradeço à Capes e ao CNP pelo suporte financeiro recebido em forma de bolsa durante grande parte do curso de doutorado.

À Universidade Federal de Santa Catarina, pelo afastamento concedido para conclusão do curso.

Ao Professor Júlio Felipe Szeremeta, chefe do Departamento de Informática e de Estatística da UFSC, pelo apoio integral e confiança durante os quase três anos em que estive afastado das atividades didáticas do departamento.

Ao meu orientador, Prof. Jean-Marie Farines, pela orientação, crítica construtiva, encorajamento e dedicação durante o desenvolvimento deste trabalho.

Ao Dr. Jean-Pierre Courtiat, e aos demais pesquisadores do *Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS - LAAS*, em Toulouse, pelas discussões frutíferas e pela acolhida durante meu estágio neste laboratório.

Aos professores Joni da Silva Fraga, José Antônio Monteiro de Queiroz, Maurício Ferreira Magalhães, Rosvelter João Coelho da Costa e Wanderley Lopes de Souza, por terem aceitado julgar este trabalho e pelos valiosos comentários que fizeram.

Aos Professores Rosvelter João Coelho da Costa, José Mazzucco Júnior e Fábio Teixeira Campos pelo incentivo, apoio e por terem me substituído em aulas quando da minha impossibilidade.

Aos funcionários do LCMI e PPGEEL pela prestimosa colaboração durante todo o curso.

À minha mãe Izis Silva da Costa pelo especial encorajamento, apoio e incentivo durante todo o desenvolver deste trabalho. Espero que ela possa ver neste estudo, um pouco abstrato, o resultado de seu afeto e de sua devoção.

Agradeço especialmente à minha esposa Clarice e a minha filha Carolina, por toda inspiração e ânimo que o amor, a paciência e a alegria que vieram delas promoveram durante todos esses anos.

Resumo

LOTOS é uma linguagem formal para especificação de sistemas padronizada pela ISO desde 1988. Do ponto de vista teórico, LOTOS é bem fundamentada e apoia-se no paradigma de álgebras de processos para especificação da parte comportamental de um sistema. Esta linguagem oferece uma abordagem nova com técnicas e métodos formais para verificação dos sistemas descritos por ela. LOTOS tem sido usada com sucesso para especificação de sistemas, em particular de sistemas distribuídos.

Contudo, o padrão LOTOS possui uma limitação grave: não possui qualquer elemento semântico que permita especificar de forma explícita sistemas onde o tempo intervém. Tais, sistemas chamados também de sistemas dependentes do tempo, incluem uma grande classe de sistemas importantes como por exemplo: os sistemas tempo-real e as aplicações multimídia. Sistemas dependentes do tempo têm características bastante diferentes dos demais, e neles, até o conceito de correção é diferente e depende também da dimensão do tempo. Sendo assim, uma extensão de LOTOS com as características temporais que lhe faltam a tornaria apta para especificar sistemas dependentes do tempo, e se constituiria numa abordagem interessante para especificação e verificação deste tipo de sistemas.

Nesta tese é apresentada uma extensão da linguagem LOTOS com capacidade para representar e tratar restrições de tempo associadas à ações (tais ações são referenciadas como ações temporizadas). Estas restrições, especificadas como intervalos de tempo, têm uma influência direta no modo que as ações podem ser oferecidas e sincronizadas. Uma característica importante do modelo proposto é o caráter "imperativo" que foi dado às restrições de tempo, daí o nome RT-LOTOS (Real Time LOTOS). Ações específicas são introduzidas no modelo para expressar violações temporais que são a conseqüência das ações não realizadas. Assim, o modelo apresentado torna-se particularmente atraente para especificação formal de sistemas dependentes do tempo. Um novo operador, o de preempção temporal, é definido na linguagem para permitir ao usuário expressar o comportamento do sistema na presença de falhas do sistema devidas a violações temporais. A definição formal de RT-LOTOS inclui: a definição de uma semântica operacional estrutural *à la Plotkin*, a definição de bissimulações no contexto temporizado e o estabelecimento de uma definição semântica alternativa em termos de Grafos Temporizados. Esta definição da semântica de RT-LOTOS em Grafos Temporizados é particularmente interessante pois permite propor técnicas e construir ferramentas para verificação de sistemas dependentes do tempo bastante eficazes. Alguns exemplos das áreas de sistemas tempo-real, protocolos de comunicação e aplicações multimídia são propostos para ilustrar os principais conceitos deste formalismo.

Palavras-chave: LOTOS, álgebra de processos, especificação formal, verificação, grafos temporizados, sistemas distribuídos, sistemas tempo-real, aplicações multimídia.

Abstract

LOTOS is a formal language for system specification standardized by ISO since 1988. From a theoretical point of view LOTOS is a well-defined language based in the process algebra paradigm. Its main application is in the specification of the behavioral part of a system. LOTOS provides a new approach with formal techniques and methods for verification of the described systems. This language has been successfully used in the specification of classical systems such as distributed systems.

Nevertheless, LOTOS has a serious limitation: it doesn't have any semantic element that allows the explicit specification of systems with time dependencies. Such system includes a wide range of important systems like real-time systems and multimedia applications. Time dependent systems have peculiar characteristics, specially in the sense that correctness concept also includes time dimension (timeliness). A temporal extension of LOTOS, with all necessary temporal characteristics, would enable the specification of time dependent systems and would be an interesting approach to specify and verify this class of systems.

In this thesis we extend the LOTOS formal description technique to express and handle temporal constraints associated with actions (such actions are referred to as timed actions). These constraints, specified as time intervals, have a direct influence on the way the actions are offered and synchronized. An important feature of the proposed model is that temporal constraints are imperative, hence the name of RT-LOTOS (Real Time LOTOS). Specific actions are introduced in the model to express temporal violations that are consequence of certain non-realized actions. Thus, the proposed model becomes particularly attractive for the formal specification of time dependent systems and applications. A new operator, temporal disrupt, is defined in the language to enable the user to express the system behavior in presence of system fails due to temporal violations. The formal definition of the RT-LOTOS includes: a structural operational semantic *à la Plotkin* definition, time context bisimulation' definition and an alternative semantic definition of RT-LOTOS in Timed Graphs. This timed graphs semantic of RT-LOTOS is particularly important since it permits the proposal of news techniques and the construction of news tools for effective time dependent systems verification. Some examples from the real-time systems, communication protocols and multimedia areas are proposed to illustrate the basic concepts of the formalism.

Keywords: LOTOS, process algebra, formal specification, verification, timed graphs, distributed systems, real-time systems, multimedia applications.

Índice

Lista de figuras

Lista de tabelas

Introdução

Correção, confiabilidade e bom desempenho são requisitos exigidos normalmente no desenvolvimento de sistemas computacionais em geral, podendo ter no caso dos dois últimos requisitos citados um grau maior ou menor de importância dependendo da aplicação. Para sistemas computacionais dependente do tempo e particularmente para aplicações altamente críticas um cuidado especial com esses requisitos se faz necessário. O próprio conceito de correção muda neste tipo de sistemas pois ela depende não só dos resultados lógicos computados, mas também do tempo no qual esses resultados são produzidos. Atualmente, os sistemas dependentes do tempo incluem três grandes classes de aplicações computacionais: sistemas tempo-real, protocolos de comunicação e sistemas multimídia. Algumas das aplicações de sistemas tempo-real, tais como, sistemas de controle de tráfego aéreo, plantas de energia nuclear, sistemas de comando de aviões, sistemas automotivos e aplicações de robótica, apresentam características altamente críticas; isto é, a ocorrência de falhas em tais sistemas pode originar danos sérios. Não é difícil imaginar as catástrofes que poderiam ser geradas por comportamentos incorretos de sistemas como os anteriormente citados.

Os sistemas dependentes do tempo são, em geral, difíceis de projetar e entender devido à sua alta complexidade. Desta maneira, a utilização de técnicas formais para auxílio no desenvolvimento destes sistemas, nas diversas fases do ciclo de vida (especificação, projeto e implementação), apresenta-se atualmente como uma solução necessária para o desenvolvimento deste tipo de sistemas pois a partir da utilização de técnicas formais é possível:

- descrever de maneira precisa, sem ambigüidades, as funções do sistema;
- analisar e validar estas descrições, podendo assim detectar erros de concepção e
- utilizar ferramentas de software baseadas no formalismo matemático escolhido para suportar as atividades do ciclo de vida.

Muitos modelos formais permitem dar suporte ao ciclo de vida de sistemas computacionais complexos. Os exemplos mais comuns são: Máquinas de Estados Finitas, Lógicas Temporais, Redes de Petri e Álgebras de Processos, entre outros.

Neste trabalho o interesse é centrado no desenvolvimento de uma linguagem de especificação, baseada em álgebras de processos, para sistemas onde o tempo intervém. Ao formalismo desenvolvido aqui, deu-se o nome de RT-LOTOS (para Real-Time LOTOS), visto que é uma extensão da linguagem de especificação LOTOS [ISO 88].

Esta dissertação está organizada em quatro capítulos que estão organizados da seguinte maneira:

- o capítulo I caracteriza o contexto geral no qual o trabalho desenvolvido nesta tese se insere, apresentando os conceitos básicos relacionados à representação formal do tempo, as principais características e requisitos gerais de sistemas dependentes do tempo, uma visão geral das principais abordagens formais existentes para especificação de sistemas dependentes do tempo, e uma rápida sinopse e comparação das principais extensões temporais de LOTOS existentes na literatura;

- o capítulo II é dedicado à apresentação do formalismo básico proposto nesta tese. Ele começa apresentando a visão intuitiva que levou à RT-LOTOS. Apresenta-se a seguir a sua sintaxe e semântica informal. A seguir, a álgebra de processos RT-LOTOS Básica é inteiramente definida, apresentando-se uma semântica operacional *à la Plotkin*, um conjunto de propriedades satisfeitas pelo formalismo, e noções de bissimulações no contexto temporal. Neste mesmo capítulo, para certificar-se da expressividade de RT-LOTOS, apresenta-se a seguir um conjunto de exemplos de especificações de mecanismos e situações comuns em sistemas dependentes do tempo. Finalmente, discute-se o interesse de uma representação de RT-LOTOS no modelo de Grafos Temporizados e apresenta-se uma definição semântica de RT-LOTOS neste modelo;

- no capítulo III, estende-se o formalismo básico RT-LOTOS para incorporar uma nova funcionalidade que permite registrar o tempo e discute-se o problema do acoplamento da representação de dados ACT ONE a RT-LOTOS para obter-se a linguagem RT-LOTOS Completa. Apresenta-se a sintaxe da linguagem RT-LOTOS Completa e discute-se a respeito da definição formal da sua semântica. Encerrando este capítulo, apresenta-se um conjunto bastante variado de exemplos e de aplicações com objetivo de mostrar o poder de expressão da linguagem proposta;

- o capítulo IV encerra esta dissertação e apresenta-se nele: as principais conclusões sobre a proposta feita neste trabalho e algumas das suas limitações, uma breve comparação de RT-LOTOS com as outras extensões temporais de LOTOS apresentadas no capítulo I, e perspectivas futuras para a continuação deste trabalho.

Capítulo I

Quadro Geral do Trabalho

Neste capítulo, é apresentada uma visão geral do contexto no qual o presente trabalho se insere. Inicialmente, a problemática geral, alguns conceitos básicos e uma visão geral sobre sistemas tempo-real e sistemas multimídia são apresentados. A seguir, são discutidas as diversas abordagens formais para o desenvolvimento de sistemas dependentes do tempo; incluindo-se aí algumas das mais importantes: redes de Petri com tempo, lógicas temporais tempo-real e álgebras de processos temporizadas. Em seguida, são expostas e discutidas as escolhas feitas para a abordagem apresentada neste trabalho. Finalmente, são apresentados um conjunto de características e propriedades de álgebras de processos temporizadas e uma sinopse comparativa das principais extensões temporais de LOTOS.

I.1 Apresentação da problemática dos sistemas dependentes do tempo e conceitos básicos

I.1.1 O tempo e suas interpretações

Nesta seção apresenta-se a noção de tempo como é entendida na informática com algumas das suas principais interpretações. Esta seção se apresenta como uma compilação de vários trabalhos [Raynal 91, Motus 92, Farines 93].

Tempo na execução e na programação: A partir do conceito abstrato do tempo, pode ser feita uma primeira diferenciação deste conceito do ponto de vista da informática entre:

- o tempo como *suporte de execução*, onde este é considerado como um recurso a ser consumido pelo sistema em execução tal qual outros recursos físicos e lógicos; neste caso a intervenção do tempo é implícita e
- o tempo como *objeto de programação*, onde este pode ser visto como uma grandeza a ser manipulada pelo sistema do mesmo modo que outras variáveis (inteiros, booleanos, etc); neste caso a intervenção do tempo pode ser implícita ou explícita.

A segunda abordagem é de maior interesse neste trabalho e, assim, passa-se a uma apresentação de diferentes maneiras de qualificar o tempo nesse contexto.

O tempo como objeto de programação: visto como objeto de programação o tempo tem como função principal prover uma base para a especificação das restrições temporais impostas aos sistemas. E deste ponto de vista, devem ser distinguidos os conceitos de tempo apresentados a seguir:

- Tempo físico versos tempo lógico:
  - *Tempo físico:* este é um tempo métrico usado para representar a distância entre eventos.
  - *Tempo lógico:* nesta interpretação do tempo, o interesse reside unicamente na ordem dos eventos. Este tempo, chamado de *tempo causal*, é interno ao programa e serve para definir uma relação de ordem total sobre o conjunto dos eventos.

    Existe uma segunda interpretação de tempo lógico que é definido externamente ao sistema. Este tempo lógico produz uma seqüência global de pulsações que regula a evolução do sistema de acordo com um modelo computacional síncrono[1].
- Tempo global versos tempo local:
  - *Tempo global:* tempo global é uma noção abstrata (como se fosse vista por um observador externo) que deve ser oferecida a todo usuário de um sistema distribuído para lhe permitir ter acesso a um tempo físico de referência único.
  - *Tempo local:* ainda com referência a um sistema distribuído, um tempo local é um tempo (lógico ou físico) usado num nodo do sistema.
- Tempo absoluto versos tempo relativo:
  - *Tempo absoluto:* no tempo absoluto a referência é sempre estabelecida em relação a um evento global que serve como origem do tempo (por exemplo o instante em que o sistema foi ligado). O tempo absoluto é sempre global.
  - *Tempo relativo:* no tempo relativo uma referência é estabelecia em relação a um dado evento local no sistema. Geralmente, pode-se ter vários tempos relativos simultaneamente num sistema. O tempo relativo é sempre local.

Do ponto de vista da representação do tempo, este pode ser ainda classificado como denso ou esparso:

- *Tempo Denso:* diz-se que o tempo é denso se ele é representado pelos números reais positivos, sobre os números racionais positivos, ou mesmo sobre outro conjunto denso qualquer.
- *Tempo Esparso:* se o tempo é representado pelo conjunto dos números naturais positivos ou por um outro que seja isomorfo a este, então ele é dito esparso.

[1] No contexto da *execução computacional*, um processo pode seguir um dos dois seguintes tipos de computação [Scholefield 90]:

Síncrono: um processo em computação síncrona evolui em compasso com os *"ticks"* de um relógio.

Assíncrono: um processo em computação assíncrona pode evoluir em qualquer velocidade; seu progresso não é ditado por um relógio, ou por qualquer outro processo (exceto por uma comunicação voluntária). Isto é, processos assíncronos executam suas ações de forma independente uns dos outros.

No capítulo seguinte os conceitos de tempo denso e tempo esparso serão definidos formalmente.

I.1.2 Características e requisitos de sistemas dependentes do tempo

Esta seção é dedicada a uma rápida apresentação das principais características e requisitos de três grandes classes de sistemas informáticos dependentes do tempo: sistemas tempo-real, protocolos de comunicação e aplicações multimídia.

I.1.2.1 Sistemas Tempo-Real

I.1.2.1.1 Definição de Sistemas Tempo-Real

São muitas as definições encontradas na literatura para os Sistemas Tempo-Real e a maior parte delas está estreitamente relacionada com os problemas específicos tratados pelos seus autores [Dodhiawala 89]. Neste trabalho, objetivando a maior abstração possível de aplicações específicas, adotaremos a definição de Sistema Tempo-Real apresentada em [Audsley 90].

Definição 1 Um *Sistema Tempo-Real* é um sistema que deve reagir a estímulos do ambiente (incluindo a passagem do tempo físico) dentro de intervalos de tempo impostos pelo comportamento do seu ambiente.

Na classe de sistemas tempo-real em que se encontram os sistemas embutidos, faz-se a distinção entre o sistema (ou objeto) a controlar, o sistema computacional de controle e o operador. O sistema a controlar e o operador são considerados como ambiente do sistema computacional. A interação entre eles se faz através das interfaces de instrumentação (compostas de sensores e atuadores) e da interface de operador. A figura I.1 mostra o esquema geral de um sistema tempo-real.

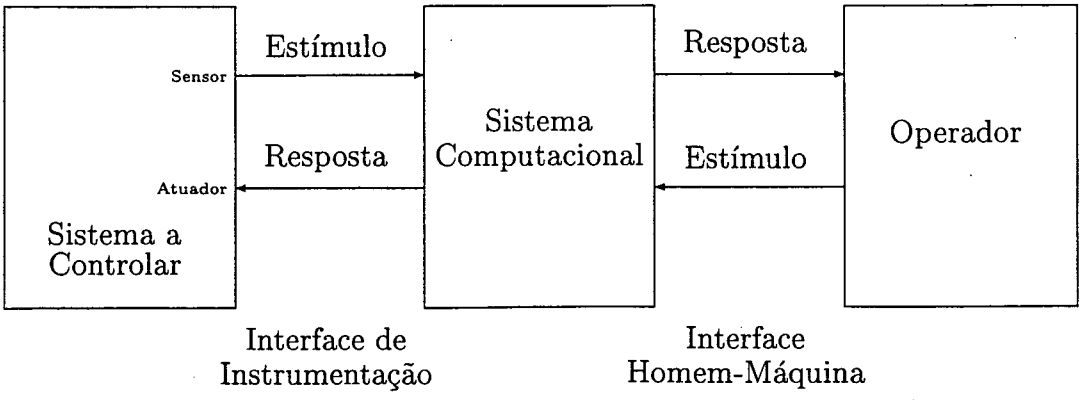


Figura I.1: Sistema Tempo-Real

I.1.2.1.2 Requisitos Gerais de Sistemas Tempo-Real

A seguir serão apresentados os requisitos gerais exigidos para os Sistemas Tempo-Real que permitem diferenciá-los dos outros sistemas e classificá-los.

Esses requisitos são normalmente determinados pelo ambiente. Todos os sistemas tempo-real, por definição, devem satisfazer o requisito de *Correção Temporal* (*"Timeliness"*), isto é, devem reagir a estímulos do ambiente dentro de intervalos de tempo especificados. Existem ainda outros requisitos que permitem caracterizar os sistemas tempo-real e até classificá-los [Ausdley 90]:

- *Relação de ordem ("Orderliness").* As relações de ordem entre as entradas do sistema devem ser mantidas nas saídas correspondentes do sistema.
- *Atualidade ("Freshness").* O sistema deve usar, ou fornecer, o dado que seja o mais válido possível, do ponto de vista do comportamento temporal.
- *Previsibilidade* (*"Predictability"*). "O comportamento funcional e temporal de um sistema tempo-real deve ser tão determinista quanto necessário para satisfazer as especificações do sistema", diz Stankovic em [Stankovic 88]. Do ponto de vista temporal, um sistema tempo real pode ser: imprevisível, estocasticamente previsível ou determinista.
- *Coordenação* (*"Coordination"*). As ações de um sistema no seu ambiente devem ser coordenadas de modo que um não prejudique o outro.

I.1.2.1.3 Sistemas Tempo-Real Críticos

Definição 2 Um sistema tempo-real é dito *crítico* se existir pelo menos uma falha que pode ser julgada catastrófica. No caso de todas as falhas serem benignas, o sistema é dito *não-crítico.*

Considera-se como falha benigna uma falha cujo custo é da mesma ordem de grandeza que os benefícios do sistema em operação normal. Ao contrário, quando as conseqüências de uma única falha no sistema podem exceder aos benefícios em várias ordens de magnitude, então a falha é dita catastrófica.

Definição 3 Se em um sistema tempo-real crítico conseqüências catastróficas podem ser consideradas como resultantes de falhas no aspecto tempo, então o sistema é chamado de *"hard Real-Time Systems"* (sistema tempo-real "Hard").

Para os sistemas Tempo-Real Hard, podem se distinguir dois tipos de situações assim caracterizadas:

- *Seguro em caso de Falha* (*"Fail Safe"*): quando um ou vários estados seguros para o comportamento do ambiente podem ser alcançados em caso de falha do sistema (ex.: sistema de sinalização ferroviário).
- *Operacional em caso de Falha* (*"Fail Operational"*): quando não existe nenhum estado seguro identificável e que o sistema computacional pode fornecer um serviço mínimo em caso de falhas, evitando uma catástrofe (ex.: sistema embarcado de controle de vôo).

I.1.2.1.4 Modos de demanda e de resposta

Em [Ausdley 90], são apresentados dois outros requisitos que permitem diferenciar os sistemas tempo-real críticos: modos de demanda e modos de resposta.

Modos de Demanda: A *demanda de carga* que um sistema deve atender tem características bem variadas. A carga pode ser sempre uniforme, ou pode variar de um nível mínimo até um valor de pico, chamado de valor de carga máxima viável. Por sua vez, as variações na carga de um sistema podem estar relacionadas à sua freqüência, e a sua continuidade ou não. Em situações de *pico de carga* a operação do sistema é exigida urgentemente para evitar uma catástrofe. Sistemas com estas características são chamados de *sistemas tempo-real críticos a picos de carga* ("peak-load critical real-time systems").

Modos de Resposta: Outra maneira de se classificar os sistemas tempo-real é segundo o grau de funcionalidade requerido para evitar uma catástrofe. Existem dois modos operacionais extremos aceitáveis para prevenir uma catástrofe: *funcionalidade total*, e *silêncio*. Um modo aceitável para o funcionamento normal é ter uma funcionalidade total; vários modos intermediários fornecem os comportamentos degradados ou alternativos, em caso de falhas.

I.1.2.1.5 Restrições de Tempo em sistemas tempo-real e mecanismos de representação

Entendendo um sistema a partir do paradigma estímulo/resposta, uma restrição de tempo estabelece exigências sobre o tempo de ocorrência de estímulos e respostas em um sistema. A necessidade de ser capaz de representar estas restrições de tempo e de garanti-las, nos leva a estudá-las mais detalhadamente.

Classificação de Restrições de Tempo Três tipos de restrições temporais apresentam interesse para os sistemas tempo-real:

1. *Tempo Máximo:* no máximo, uma quantidade t de tempo pode passar entre a ocorrência de dois eventos;
2. *Tempo Mínimo:* no mínimo, uma quantidade t de tempo deve passar entre a ocorrência de dois eventos e
3. *Duração:* um evento ou uma seqüência de eventos deve ocorrer por t unidades de tempo.

O termo *"evento"* é utilizado aqui para representar os estímulos gerados pelo ambiente, bem como as respostas, observadas externamente, que o sistema produz no seu ambiente.

Cabe aqui a observação de que os três tipos de restrições acima não são mutuamente exclusivas. Por exemplo, ambos os limites de tempo máximo e mínimo podem ser associados à ocorrência de um mesmo evento, e os próprios eventos podem ter durações específicas.

A seguir, serão discutidas as possibilidades existentes de restrições de tempo para as três classes possíveis: máximo, mínimo e duração. Estas três classes de restrições de tempo podem ser analisados segundo o paradigma de estímulo/resposta para sistemas tempo-real. Os estímulos e as respostas são associados à eventos na reta de tempo da figura I.2.

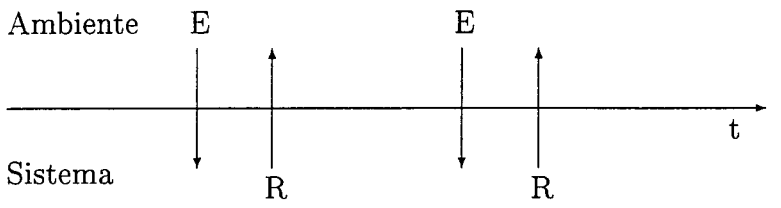


Figura I.2: Reta de tempo, dois estímulos E e duas respostas R

Máximo:

a) um tempo máximo é permitido entre a ocorrência de dois estímulos (E-E);

b) um tempo máximo é permitido entre a ocorrência de um estímulo e a resposta do sistema (E-R);

c) um tempo máximo é permitido entre a ocorrência de uma resposta e o próximo estímulo do ambiente (R-E);

d) um tempo máximo é permitido entre a ocorrência de duas respostas do sistema (R-R).

Os casos a) e c) modelam requisitos a nível de ambiente. Estas restrições podem ser descritas por um construtor do tipo *"timer"* (isto é, uma sinalização da expiração de uma quantidade de tempo especificada) [Lee 85]. Os casos b) e d) são restrições de tempo que caracterizam requisitos de tempo a nível de sistema. Estas restrições têm características que poderiam ser representadas por um construtor do tipo *"latency"* [Lee 85] que, do ponto de vista do ambiente, significa que o sistema está "ocupado" e não está disponível a ele.

Mínimo:

a) um tempo mínimo é necessário entre a ocorrência de dois estímulos (E-E);

b) um tempo mínimo é necessário entre a ocorrência de um estímulo e a resposta do sistema (E-R);

c) um tempo mínimo é necessário entre a ocorrência de uma resposta e a ocorrência do próximo estímulo do ambiente (R-E);

d) um tempo mínimo é necessário entre a ocorrência de duas respostas do sistema (R-R).

Os casos a) e c) poderiam, como no caso anterior, ser tratados com construtores do tipo *"timer"*. Os casos b) e d) poderiam ser tratados com construtores do tipo *"delay"* (ou seja, um retardo de uma quantidade de tempo) [Lee 85].

Duração:

a) aplicar um estímulo continuamente por um intervalo de tempo fixado e

b) observar uma resposta contínua por um intervalo de tempo fixado.

Observe-se que apresentou-se nesta seção alguns dos mecanismos temporais mais importantes a serem representados em sistemas tempo-real como: "timer", "delay", "latency", duração, tempo mínimo e tempo máximo.

I.1.2.2 Protocolos de Comunicação

Algumas das características gerais dos protocolos de comunicação são as seguintes [Blair 93]:

- *Concorrência:* protocolos de redes de computadores são sistemas essencialmente paralelos, pois os nodos de comunicação os executam em paralelo de maneira autônoma.
- *Interação:* os protocolos de comunicação expressam uma interação precisa entre os nodos. O termo interação denota todas as formas de comunicação entre nodos, por exemplo comunicação síncrona, comunicação assíncrona, comunicação por difusão e comunicação por memória compartilhada.
- *Imprevisibilidade ("unpredictability"):* a interação entre os sistemas nos nodos é incerta e imprecisa, isto quer dizer que mensagens podem ser perdidas e tempos de transferência de mensagens não podem ser garantidos.

Além das características citadas acima, deve-se considerar também os mecanismos temporais necessários para representação de situações bastante freqüentes em protocolos de comunicação como por exemplo a detecção e tratamento de erros: "timeout"; e também tempos mínimos e máximos, e duração de atividades de comunicação. Estes conceitos temporais usam o mesmo tipo de representação que no caso dos sistemas tempo-real.

I.1.2.3 Sistemas Multimídia

Nesta seção são apresentadas as principais características dos sistemas multimídia. A abordagem usada aqui prioriza os aspectos temporais destes sistemas, visto que a compreensão destes aspectos são o nosso objetivo maior.

Um Sistema Multimídia é entendido como um sistema que gera, processa, armazena, restitui e apresenta informações de vários tipos e origens, chamadas de mídia[2]. De maneira mais concreta, pode-se sintetizar o conceito de sistema multimídia através da definição seguinte.

[2]Um pouco diferente do significado comum de mídia utilizado em outras áreas, esta palavra neste contexto é empregada como sinônimo de "meio", ou um "conjunto de meios", ou ainda para referir-se ao coletivo de "*meios* ou métodos para fornecer uma informação", mas também pode ser entendido como "suportes que fornecem ou (e) transportam informações" ou como "as próprias informações".

Definição 4 Um *Sistema Multimídia* é um sistema que permite ao usuário final armazenar, transmitir, intercambiar, apresentar, representar e perceber uma variedade de formas de informação de uma maneira integrada.

Os sistemas multimídia combinam várias fontes de informações, tais como, voz, gráficos, áudio de alta-fidelidade e vídeo, e suas aplicações se estendem em áreas como automação de escritórios, sistemas de tele-conferência e aprendizado à distância, ambientes de cooperação científica e controle de tráfego aéreo.

Combinando recursos da multimídia com o suporte para sistemas distribuídos, tem-se os sistemas multimídia distribuídos.

Tipos de Mídia: Os vários tipo de mídia podem ser classificadas em dois grupos: estáticas ou contínuas. Usa-se o termo *mídia estática* para referir-se a mídia que não tem uma dimensão temporal. Exemplos de mídia estática são: arquivos binários e ASCII, dados numéricos de tamanho fixo (reais, inteiros, etc), texto, gráficos de vetores, imagens bit map, etc. Por sua vez, *mídia contínua* refere-se a mídia com uma forte dependência temporal, ou seja, na qual as informações são apresentadas em uma taxa especificada e por um tempo especificado [Blair 93]. Exemplos de mídia contínua são: voz e áudio de alta qualidade digitalizados, informações de vídeo digitalizadas, vídeo compactado, dados interativos, etc.

Voz digitalizada:	armazenamento para voz digitalizada (bandwidth 4KHz, tam. amostra 8 bits, taxa amostragem 8 KHz) $\approx$ 8 Kbytes/seg
Áudio (hi-fi) digitalizado:	armazenamento para áudio de alta qualidade digitalizado (bandwidth 22KHz, tam. amostra 16 bits, taxa amostragem 44 KHz) $\approx$ 108 Kbytes/seg
Vídeo digitalizado:	armazenamento para vídeo digitalizado (padrão PAL, 25 quadros/seg) $\approx$ 30 Mbytes/seg

Tabela I.1: Um sumário dos requisitos de importantes tipos de mídia contínua

A título de ilustração, apresenta-se na tabela I.1 um sumário do requisitos dos principais tipos de mídia [Blair 93]. Os valores apresentados nesta tabela foram calculados de modo a prover uma dada qualidade de serviço.

A coexistência, interação, integração e conversão de vários tipos de mídia requerem no caso de mídia contínua ou mista (estática e contínua) coordenação e sincronização; conseqüentemente devem ser levadas em conta as restrições temporais, geralmente definidas para cada aplicação a partir das exigências feitas para garantir a qualidade dos serviços oferecidos.

Na seqüência serão analisados os dois principais aspectos dos sistemas multimídia no qual o tempo intervém: a qualidade do serviço e a sincronização.

I.1.2.3.1 Qualidade de Serviço

A noção de *qualidade de serviço* é fundamental para os sistemas multimídia. Nestes sistemas é importante especificar o nível de qualidade com o qual este serviço deve ser garantido

e este nível difere segundo o tipo de aplicação e dos tipos de mídia envolvidos. A qualidade dos serviços multimídia é garantida [Farines 93] a partir da definição e do controle de um parâmetro pelo usuário ou pela aplicação, o QoS ("*Quality of Service*"); isto equivale a estabelecer e acompanhar um contrato entre o cliente (usuário) e o servidor (sistema distribuído) que especifica direitos, deveres, benefícios e condições para estes benefícios, e penalidades. O QoS especifica então estas garantias em termos de atraso, vazão, confiabilidade e outros requisitos que serão apresentados a seguir.

A garantia de QoS pode ser caracterizada de dois pontos de vista diferentes: o determinista e o estatístico. No caso determinista a garantia consiste em estabelecer limites para o desempenho dos objetos multimídia numa sessão (ex.: atraso fim-a-fim máximo). Por outro lado, no caso estatístico a garantia é expressa em percentual de objetos multimídia com desempenho satisfazendo o valor estabelecido (ex.: percentual de pacotes com atraso menor que o atraso máximo). Ainda no caso estatístico, deve-se fazer a distinção entre garantia *ad infinitum* (ou em estado permanente) e a garantia definida para um intervalo de tempo finito.

Parâmetros característicos do QoS: São muitas vezes as questões relacionadas à qualidade do serviço que impõem aos sistemas multimídia a satisfação de importantes requisitos temporais. E para melhor visualizar estes requisitos temporais, apresenta-se alguns parâmetros que caracterizam o QoS [Ferrari 90, Farines 93]:

Atraso: O *atraso fim-a-fim* corresponde ao atraso entre a geração de uma amostra (ou de um objeto multimídia) na fonte e sua apresentação no receptor (usuário). Os principais componentes de um atraso fim-a-fim são: o atraso de geração-emissão na fonte (geração da amostra, codificação, empacotamento), o atraso de transmissão na rede, e o atraso de recepção-apresentação no receptor (bufferização, desempacotamento, decodificação, apresentação).

O termo *jitter* é usado para referir-se a variação instantânea no atraso fim-a-fim de amostras (ou objetos multimídia). Ele é ocasionado principalmente por atrasos em filas na fonte e no receptor, por erros na rede e por diferenças existentes entre os relógios da fonte e do receptor. O jitter é um parâmetro importante no QoS pois a partir do seu conhecimento e do tamanho dos pacotes, pode ser dimensionado um buffer no receptor que permite filtrar estas variações de atraso. Normalmente, a ultrapassagem do valor do jitter é tolerada mas leva a uma degradação da qualidade do serviço, e os pacotes que chegam com um jitter além do valor estabelecido devem receber o mesmo tratamento que um pacote perdido.

Requisitos de atraso: O valor máximo do atraso (D_{max}) que nenhum atraso de pacote recebido poderá ultrapassar e do seu jitter (J_{max}) em relação a um atraso ideal (D) são definidos pelo usuário; cada pacote deverá respeitar limites superior ($D + J_{max}$) e inferior ($D - J_{max}$) para seu atraso.

Vazão: A *vazão* indica a taxa de transferência de pacotes entre a fonte e o receptor. Normalmente, a vazão é expressa em kilobits por segundo ou em número de pacotes entregues por segundo.

Requisitos de vazão: O limite superior da vazão de transferência de informação é sempre fornecido pela taxa de envio na fonte; a possibilidade de perda de pacotes

ou de ocorrência de erros abaixa o valor da vazão. Por outro lado, a vazão máxima é limitada ainda pela carga da rede o que pode levar a uma saturação da vazão se a fonte aumentar a taxa de envio. O usuário define um limite inferior da vazão (V_{min}) que deve ser fornecido pelo sistema. Quando a faixa passante alocada para a transferência não é constante, variando dinamicamente segundo a demanda, é interessante utilizar o limite de vazão média (V_{med}).

Confiabilidade: A confiabilidade em um sistema multimídia é medida em termos da *taxa de erro por bit* (BER) e da *taxa de erro por pacote* (PER) que representam o número de erros por unidade de tempo, por bit e por pacote respectivamente.

Requisitos de confiabilidade: Por diversas razões, os pacotes que saem da fonte podem chegar incorretos à recepção ou serem perdidos na transmissão. Assim, é suficiente expressar o limite mínimo de confiabilidade como o limite inferior da probabilidade (C_{min}) da mensagem ser entregue corretamente: $Prob$(mensagem entregue correta) $\geq C_{min}$, o que implica que a probabilidade de perda é limitada por $(1 - C_{min})$.

Outros Requisitos para o QoS: Existe ainda um conjunto de outros requisitos que podem ser propostos pelo usuário (cliente) como desejáveis e, entre estes, pode-se destacar:

- *Seqüenciamento:* No caso de fluxo contínuo ("*stream*"), é importante que a entrega seja em seqüência, mesmo que ela seja incompleta. No caso de ausência de garantia de um seqüenciamento total, deve-se permitir ao menos especificar um limite sobre a probabilidade de entrega de um mensagem fora de seqüência ou considerá-la como perda a ser juntada com as outras perdas de mensagens definidas por C_{min}.
- *Ausência de duplicações:* Pode ser necessário estabelecer um limite quando as duplicações involuntárias (isto é, não impostas por uma alta taxa de confiabilidade) forem freqüentes o que corresponde a uma disfunção no sistema de comunicação.
- *Tratamento de faltas:* Podem ser especificados os tempos de reparo máximo e o tipo de operação a ser realizada em caso de faltas.
- *Tempo de "*Setup*" de serviço:* Um limite máximo de espera para um pedido de "*setup*" pode ainda ser definido entre as garantias de um serviço.

I.1.2.3.2 Sincronização

Uma das características mais marcantes dos sistemas multimídia é o forte relacionamento temporal entre os objetos básicos ou fluxos contínuos ("*streams*") que entram na composição de um objeto multimídia, e este relacionamento temporal é a essência do conceito de sincronização neste tipo de aplicação. É comum distinguir dois tipos de sincronização que correspondem a dois tipos de relações temporais entre os objetos: a sincronização contínua e a sincronização discreta (baseada em eventos).

- A *sincronização contínua* trata da entrega sincronizada de um ou vários fluxos contínuos de informação ("*streams*") como deve ser o caso, por exemplo, em aplicações de telefonia visual onde deve-se sincronizar a recepção da voz e do vídeo gerados separadamente, ou em situações de recepção de imagens de filmes (problema conhecido como "sincronização de lábios" ou "*lip-synch*").

- A *sincronização discreta* (ou sincronização baseada em eventos) trata da entrega sincronizada de unidades de informação (por exemplo, previamente armazenadas) ou da realização de alguma ação em algum ponto de sincronização de algum ponto de sincronização de um fluxo contínuo de uma mídia. A apresentação de slide onde se precisa sincronizar a imagem e a voz ou as anotações faladas de textos feitas em pontos determinados deste (pontos de sincronização) são alguns exemplos deste tipo de sincronização.

O tratamento da sincronização contínua deve levar em conta as restrições temporais por ser mais crítica do ponto de vista do tempo-real. Por sua vez, o tratamento de sincronizações discretas restringe-se geralmente a utilizar noções de tempo lógico e de ordenamento causal de eventos [Farines 93].

Em resumo, a questão temporal está presente nos sistemas multimídia tanto nos requisitos do QoS quanto no problema da sincronização/coordenação. Nestes dois aspectos encontram-se embutidas a maioria das restrições de tempo tratadas no início deste capítulo, e que se traduzem na utilização de mecanismos clássicos para seu tratamento, tais como: timers, delays, latency etc.

I.2 Métodos formais para desenvolvimento de sistemas dependentes do tempo: as diversas abordagens

Nesta seção é apresentado um resumo dos principais métodos e técnicas formais para tratamento de sistemas dependentes do tempo destacando três dos mais significativos deles: as redes de Petri com tempo, as lógicas para tempo-real e as álgebras de processos temporizadas. O conteúdo desta seção é inspirado nos trabalhos de [Scholefield 90, Ostroff 92, Blair 93, e Farines 93].

I.2.1 As principais abordagens para o tempo-real

As principais abordagens formais para suporte do desenvolvimento de sistemas tempo-real podem ser classificadas como:

- Redes de Petri com tempo
- Lógicas temporais tempo-real
- Álgebras de processos temporizadas

Cada uma destas abordagens será vista com mais detalhes mais adiante. Porém, outros métodos e linguagens têm sido também usados para o desenvolvimento de sistemas dependentes do tempo, apesar de serem menos expressivos; poderiam ser citados entre outros [Ostroff 92]:

- VDM e Z que são linguagens de especificação baseadas em teoria dos conjuntos e lógica de predicados. Estes métodos têm sido úteis na especificação de sistemas comerciais

de grande porte, mas não possuem grandes habilidades para tratar a concorrência e o tempo-real.

- Técnicas baseadas em métodos álgebricos e teoria de autômatos/linguagens como por exemplo o formalismo apresentado em [Ramadge 87]. Estes métodos são particularmente úteis para síntese. Mas o tratamento do tempo-real e a incorporação de dados ainda não são suficientemente bem desenvolvidos nestes métodos.

Além das abordagens acima existem ainda outros métodos baseados em lógica não temporal. Tais métodos incluem:

- os chamados métodos assertivos (*assertional*) para tratamento de sistemas tempo-real que são baseados nas triplas de Hoare clássicas $\{q\}P\{r\}$ estendidas com uma assertiva C para representar características da computação tempo-real reativa [Hooman 89], e
- os métodos baseados na lógica RTL [Jahanian 86] que é uma lógica tempo-real de primeira-ordem baseada em quatro conceitos básicos: ações, estados, eventos e restrições temporais que provêem asserções sobre as temporizações dos eventos.

Na seqüência a atenção é centrada nos formalismos para o tempo-real que são dotados de bases ricas para o tratamento de sistemas tempo-real concorrentes. Alguns desses métodos são complementares [Ostroff 92]. Por exemplo, as lógicas temporais são eficientes para descrever propriedades que são próprias do sistema completo como segurança (*safety*), justiça (*fairness*), vivacidade (*liveness*), correção temporal (*timeliness*), e propriedades temporais, mas as especificações em lógicas temporais são complexas e desprovidas de estruturas. Enquanto, por outro lado, as álgebras de processos oferecem noções de processos altamente estruturadas e levam a especificações relativamente simples, mas apresentam dificuldades para especificar propriedades globais como justiça (*fairness*). A combinação destas duas abordagens parece apontar para um caminho bastante promissor para o tratamento de sistemas tempo-real e um exemplo disso pode ser visto em [Hansson 91]. Outros exemplos de complementaridade se encontram na literatura.

I.2.2 Redes de Petri com tempo

A teoria de *redes de Petri* [Brams 83] (RdP) foi um dos primeiros formalismos introduzidos para tratar concorrência, não-determinismo e conexões causais entre eventos. Duas características das redes de Petri são a legibilidade dos modelos gerados e a capacidade de expressar, de maneira natural, as diversas relações de causalidade: paralelismo, seqüenciamento, não determinismo e sincronização. No apêndice A encontra-se a definição formal de redes de Petri.

Redes de Petri têm sido criticadas por não serem aptas a tratar com justiça (*fairness*) e com estruturas de dados, mesmo que um número de fichas em um determinado lugar possam ser vistas como uma variável local de programa [Ostroff 92]. Outrossim, mecanismos de estruturação tais como operadores de composição não são nativos na teoria de redes de Petri apesar de poder ser introduzidos se necessário.

A teoria de redes de Petri foi também um dos primeiros formalismos para concorrência a tratar com o tempo-real, associando o tempo aos lugares, aos arcos, ou as transições, de

diversas formas. Existe uma equivalência mostrada na literatura entre a associação do tempo com lugares e arcos e a associação do tempo com transições. Dessa maneira, optou-se por apresentar apenas algumas extensões temporais de redes de Petri que associam o tempo às transições.

I.2.2.1 Alguns modelos de redes de Petri com tempo:

O modelo de Ramchandani e Zuberek No modelo de Ramchandani e Zuberek [Rudin 85] a inclusão do tempo é feita pela associação de uma quantidade fixa de tempo τ_{const} a cada transição. A regra de disparo de transições é a seguinte: tão logo existam fichas suficientes para habilitar uma transição, ela dispara imediatamente, consumindo efetivamente as fichas pela quantidade de tempo τ_{const}; após este tempo elas são liberadas e depositadas nos lugares de saída. Ou seja, no disparo de uma transição t_r, as fichas envolvidas na transição permanecem em t_r, de forma indisponível, durante o tempo τ_{const} associado à transição.

O modelo de Razouk No modelo de Razouk [Rudin 85] uma transição deve permanecer sensibilizada durante um tempo t_{min} para ser disparada. Se a transição dispara, então o seu disparo consome um intervalo de tempo t_{const} e desta maneira as fichas só ficam disponíveis nos lugares de saída após este intervalo.

Redes de Petri Temporais - o modelo de Merlin O modelo redes de Petri Temporais de Merlin [Merlin 76] estende o modelo básico de redes de Petri associando a cada transição um intervalo de números, (t_{min}, t_{max}), podendo ser definido sobre os números racionais não negativos. Neste modelo o disparo de transições não consome tempo (instantânea). Para que o disparo de uma transição possa ocorrer ela deve estar sensibilizada por um tempo mínimo t_{min}, quando se torna possível o seu disparo; ela pode ficar sensibilizada até um tempo máximo t_{max}, quando então ela *deve* necessariamente ser disparada, se ainda não o fez. Uma transição sensibilizada pode ser desabilitada pela chegada de uma ficha que sensibilize outra transição que possua um tempo máximo de disparo inferior ao da primeira transição. Esta característica do Modelo de Merlin torna-o expressivo o suficiente para representar situações de *timeouts*. Além disso, este formalismo possui expressividade suficiente para representar maquinas de Turing de dois contadores [Bolognesi 89].

No apêndice A encontra-se uma descrição mais detalhada do modelo redes de Petri Temporais [Merlin 76].

Resumo das características temporais das redes de Petri A tabela I.2 resume as características temporais dos modelos apresentados. O significado de t, t_1, t_2, t_3, t_4, t_5 e τ é dado junto à tabela.

$t \equiv$ tempo no qual a transição está sensibilizada
$t_1 \equiv$ menor tempo no qual as fichas podem ser removidas dos lugares de entrada
$t_2 \equiv$ maior tempo no qual as fichas podem ser removidas dos lugares de entrada
$\tau \equiv$ período de tempo requerido para disparo de uma transição
$t_3 \equiv$ menor tempo no qual as fichas podem ser colocadas nos lugares de saída
$t_4 \equiv$ maior tempo no qual as fichas podem ser colocadas nos lugares de saída
$t_5 \equiv$ tempo do início do disparo da transição

Autor	t_1	t_2	τ	t_3	t_4	t_5
Ramchandani	0	0	τ_{const}	τ_{const}	τ_{const}	0
Razouk	t_{min}	t_{min}	t_{const}	$t_{min} + t_{const}$	$t_{min} + t_{const}$	t_{min}
Merlin	t_{min}	t_{max}	0	t_{min}	t_{max}	$t_{min} \leq t \leq t_{max}$

Tabela I.2: Resumo de alguns modelos de RdP que incluem tempo

Ainda existem várias extensões de Redes de Petri com probabilidades que permitem estudos de desempenho dos sistemas representados.

Em resumo, a teoria de redes de Petri tem sido utilizada, com bons resultados, como técnica de suporte para desenvolvimento de sistemas dependentes do tempo. Seu uso em áreas como protocolos de comunicação e sistemas tempo-real tem sido freqüentes. Porém, as limitações do formalismo básico (falta de estruturação e fraca representação de dados) e os problemas de complexidade associados à sua análise, entre outras características, tornam as redes de Petri um formalismo incompleto para tratar as os principais problemas ligados ao desenvolvimento de sistemas tempo-real mais complexos.

I.2.3 Lógicas temporais tempo-real

Lógica Temporal Lógica temporal é uma classe de designação para as lógicas modais em que os operadores modais são interpretados de uma maneira temporal [Alur 92]; por exemplo, o operador $\Box$ é interpretado como "doravante" e seu dual $\Diamond$ significa "eventualmente". A primeira aplicação de lógica temporal em informática foi realizada por A. Pnueli em 1977, e deste então ela tem sido intensamente estudada.

Lógicas temporais são compostas de operadores lógicos proposicionais que combinados com um conjunto de operadores temporais permitem que as seguintes propriedades sejam satisfeitas:

1. *Invariâncias*: propriedades que são sempre verdadeiras em um programa
2. *Eventualidade*: propriedades que devem tornar-se verdadeiras em algum tempo futuro no sistema
3. *Precedências*: asserções que estabelecem que um evento deve ocorrer antes de um outro.

Exemplos de operadores de lógicas temporais são:

$\bigcirc P$	significando	*em seguida P*
$\Box P$	significando	*doravante P*
$\Diamond P$	significando	*eventualmente P*
$P\,\mathcal{U}\,Q$	significando	*P até Q*

A combinação dos operadores ordinários com os temporais permitem expressar propriedades interessantes de sistemas concorrentes. Alguns exemplos são dados na seqüência.

1. A especificação da exclusão mútua entre dois processos P_1 e P_2 que compartilham um recurso pode ser expressa em [Blair 93] pela fórmula:

$$\Box\neg(crit\_region\_P_1 \wedge crit\_region\_P_2)$$

2. A propriedade de resposta típica de sistemas reativos "*todo estímulo do ambiente p deve ser seguido por uma resposta do sistema q*" pode ser expresso como em [Alur 92] pela fórmula:

$$\Box(p \rightarrow \Diamond q)$$

3. A propriedade "*todo estímulo p é possivelmente seguido por uma resposta q*" pode ser expresso como em [Alur 92] pela fórmula:

$$\forall\Box(p \rightarrow \exists\Diamond q)$$

Como visto anteriormente, lógicas temporais proporcionam uma maneira natural e sucinta de expressar *qualitativamente* as características de causalidade desejadas de um sistema não dependente do tempo. Porém os operadores temporais tradicionais não se referem a uma métrica de tempo explícito e, assim, são insuficientes para especificar requisitos temporais *quantitativos* [Alur 92].

Lógicas temporais tempo-real Lógicas temporais tempo-real estendem as lógicas temporais clássicas incorporando possibilidades de representação e tratamento de propriedades *quantitativas* tais como periodicidade, deadlines, e delays. Existem basicamente duas abordagens para as lógicas temporais tempo-real: lógicas com relógio explícito e com relógio implícito. Existem ainda muitas possibilidades de se definir a semântica temporal de uma lógica temporal tempo-real. Entre os modelos semânticos mais utilizados [Ostroff 92] encontram-se:

- *Semântica de intervalos.* A semântica é representada sobre intervalos de tempo nos quais são representados porções finitas do comportamento do sistema. Uma característica deste tipo de lógica temporal tempo-real é que um intervalo pode ser dividido em dois sub-intervalos contíguos através de operadores de corte (*chop*).

- *Semântica de pontos.* Na semântica de pontos as fórmulas temporais são interpretadas como requerendo algum comportamento do sistema com respeito a alguma referência pontual no tempo. Dois tipos de operadores estão presentes neste caso: operadores de *passado* que se referem ao tempo antes do ponto de referência, e operadores de *futuro* que se referem ao tempo após o ponto de referência.

  A semântica de pontos pode ser subdividida em classes principais[3]:

[3]Recentemente, um novo tipo de semântica, a *semântica de ordem parcial*, tem sido bastante estudada [Ostroff 92].

- *Semântica linear.* Na semântica linear, cada momento tem apenas um futuro possível que corresponde à história do desenvolvimento do sistema. Em outras palavras, cada seqüência de estados representa uma seqüência de execução do sistema.
- *Semântica arborescente.* Nesta semântica o tempo tem uma estrutura ramificada no qual, em cada instante de tempo, o tempo pode ser dividido em caminhos alternativos que representam as várias escolhas feitas pelo sistema durante a sua execução.

Algumas lógicas temporais tempo-real A seguir são apresentadas algumas das lógicas temporais tempo-real mais conhecidas na literatura.

A *Metric Temporal Logic* (MTL) [Koymans 90] é uma lógica temporal de semântica de tempo linear para tratamento de sistemas dependentes do tempo. Interpretações para MTL são estruturas de ponto baseados em um domínio de tempo denso e uma função de distância para representar certas restrições temporais. Os operadores de MTL são interpretados sobre estruturas de pontos. Por exemplo [Ostroff 92], a fórmula $A \rightarrow \Diamond_{\leq 5} B$ significa que se A ocorre, então eventualmente dentro de 5 unidades de tempo B deve ocorrer.

A *Real Time Temporal Logic* (RTTL) [Ostroff 89] é uma lógica temporal para o desenvolvimento de sistemas a eventos discretos. RTTL possui semântica de tempo linear definida sobre uma estrutura de pontos os quais são definidos em um domínio de tempo discreto. RTTL permite definir asserções sobre transições em máquinas de estado estendidas.

A *Temporal Computation Tree Logic* (TCTL) [Alur 90] é uma lógica temporal com semântica de tempo arborescente e semântica temporal de pontos definidos sobre um domínio de tempo denso. Em TCTL os operadores temporais podem ser subscritos com intervalos de tempo para representação de restrições temporais. Por exemplo, a propriedade "P é válido ao menos uma vez durante o intervalo de tempo (a, b) ao longo de algum caminho computacional" pode ser representada em TCTL como $\exists \Diamond_{(a,b)} P$.

Em [Ostroff 90] e [Alur 92] pode-se encontrar um estudo detalhado das principais lógicas temporais tempo-real, bem como uma análise comparativa de importantes aspectos práticos ligados a estas lógicas tais como: decidibilidade, verificação de modelos (*model checking*[4]) e procedimentos semi-automatizados.

I.2.4 Álgebras de processos temporizadas

Álgebras de processos A abordagem de álgebra de processos tem sido desenvolvida dentro do campo da teoria da concorrência e teve seu início a partir dos trabalhos de Hoare, com o seu Communicating Sequential Processes - CSP [Hoare 85], e de Milner, com o seu Calculus of Communicating Systems - CCS [Milner 80].

Além de CCS e CSP, que são os marcos de referência para as álgebras de processos, duas outras álgebras de processos, CIRCAL [Milne 85] e ACP [Bergstra 86], se tornaram importantes mais recentemente. A significância dos métodos baseados em álgebra de processos

[4] O problema de verificação de modelos (*model checking*) é definido [Ostroff 92] como segue: verificar se uma fórmula φ satisfaz todas as computações de um programa ou sistema de estados finitos.

é devida principalmente à elegância pela qual, com poucos construtores, pode-se obter uma linguagem capaz de expressar toda a complexidade de sistemas concorrentes ou de sistemas distribuídos.

Os principais operadores existentes nas álgebras de processos são:

- *Comunicação síncrona:* Um modelo de comunicação síncrona [5] é utilizado como operação primitiva.
- *Escolha:* Um construtor de escolha é usado para se definir processos com comportamentos alternativos.
- *Não determinismo:* O não determinismo é usualmente expresso como um caso especial da escolha.
- *Concorrência:* Existem operadores para expressar as várias formas da composição paralela (entrelaçamento, paralelismo verdadeiro, sincronização).
- *Restrição:* A ocultação de informações é definida formalmente em álgebras de processos usando operadores de restrição. Normalmente, uma ação especial é usada para atividades ocultadas (internas).
- *Renomeação:* Utiliza-se operadores de renomeação para evitar o conflito de nomes.

A grande generalidade das álgebras de processos possuem operadores para expressar este conjunto de operações. Porém, muita diferença existe entre as várias álgebras na semântica destas operações, na sua utilização e, também, na introdução ou não de outros operadores.

As álgebras de processos se constituem hoje como uma das abordagens mais ricas e promissoras para especificação, verificação e testes de sistemas distribuídos. Tanto isso é verdade que desde 1988 que a linguagem LOTOS [ISO 88] é um padrão ISO para especificação de sistemas distribuídos. LOTOS é uma linguagem baseada na álgebra de processos CCS, e enriquecida com o princípio de comunicação síncrona de *rendezvous* múltiplo do CSP e a representação de dados algébricos Act One. Entretanto, o modelo básico de álgebras de processos não é suficientemente adequado para modelar comportamentos tempo-real. Em particular dois problemas são identificados em todas as abordagens básicas de álgebras de processos [Blair 93]:

1. Não é possível especificar quantitativamente o tempo preciso que um evento deve ocorrer.
2. Não é possível forçar a ocorrência de certos eventos.

[5] Classicamente a comunicação pode ser classificada em síncrona e assíncrona:

- Em comunicação síncrona o emissor e receptor se encontram em algum instante do tempo, e neste instante eles trocam dados.
- Em comunicação assíncrona o emissor coloca dados em um *buffer*; e em algum instante no futuro o receptor pode obter estes dados no buffer.

Contudo, nos últimos anos algumas extensões de álgebras de processos têm sido bastante usadas também para especificação de sistemas dependentes do tempo em geral, e mais particularmente de sistemas tempo-real. Por exemplo, as álgebras de processos apresentadas em [Azcorra 90, Baeten 89, Bolognesi 90, Cardelli 82, Hansson 91, Hennessy 90, Moller 89, Nicollin 90, Yi 91], entre outras, tem características que objetivam o tratamento desse tipo de sistemas.

Neste trabalho o interesse central reside em uma extensão temporal da álgebra de processos LOTOS. E procurando tornar a leitura desta tese mais linear, deixou-se para o apêndice B a descrição de CCS [Milner 89] e das suas principais extensões temporais. Também com este mesmo objetivo foi deixada a apresentação da álgebra de processos LOTOS Básico [Bolognesi 87] para o apêndice C. Porém, neste capítulo as extensões temporais de LOTOS serão ainda estudadas em detalhe.

I.3 As extensões temporais de LOTOS

Esta seção tem como objetivo o estudo e apresentação das principais características das extensões temporais de LOTOS. Inicialmente, discute-se a utilização de LOTOS como modelo básico para a incorporação do tempo. Depois, são discutidas as principais caracterísricas e propriedades pelas quais se pode classificar e comparar as extensões temporais de LOTOS. Depois disso, é apresentada uma sinopse das principais propostas de LOTOS temporizadas e um resumo comparativos destas.

I.3.1 Por quê escolher a Álgebra de Processos LOTOS?

Antes mesmo de se discutir a respeito da escolha de LOTOS como formalismo de base para especificação de sistemas dependentes do tempo, cabe uma pergunta ainda anterior: por quê uma álgebra de processos ao invés de uma lógica temporal ou uma rede de Petri? Não existe uma resposta única para esta pergunta. Advogando-se em favor das álgebras de processos, pode-se destacar entre outras qualidades que elas são geralmente bem dotadas de meios para se modelar concorrência, comunicação e composição de processos; e também são altamente estruturadas e permitem obter especificações simples.

Por outro lado, pode-se apresentar argumentos desfavoráveis às lógicas temporais e as redes de Petri. As lógicas temporais se constituem em um excelente instrumento para se descrever propriedades de um sistema; porém, as especificações em lógicas temporais são desprovidas de estruturas e são altamente complexas. As redes de Petri, apesar de serem um poderoso formalismo para especificação de sistemas concorrentes, não possuem (pelo menos os modelos básicos) operadores de composição de processos.

Feita a opção por uma álgebra de processos, não é difícil justificar a escolha de LOTOS como o formalismo de base neste trabalho. LOTOS é a mais completa álgebra de processos definida na atualidade [Blair 93], mesmo sendo incompleta em alguns sentidos, como por exemplo:

- incapacidade de representar corretamente processos dependentes do tempo,

- não representação do paralelismo verdadeiro, sendo que a semântica atribuída ao operador de composição paralela é de entrelaçamento de ações, e
- a não possibilidade de representação de processos móveis e da reconfiguração dinâmica de sistemas distribuídos.

LOTOS se destaca pelas características seguintes:

- é uma linguagem assíncrona (como o mundo que se pretende representar);
- é uma linguagem simples e bem estruturada;
- permite o tratamento da concorrência, paralelismo e interação entre processos;
- possibilita a representação e tratamento dos dados de um sistema.

Além disso, LOTOS é a única álgebra de processos que conseguiu sair com sucesso da comunidade acadêmica e fazer incursões no mundo das aplicações industriais. LOTOS foi padronizada pela ISO [ISO 88], e tem sido usada extensivamente na especificação de protocolos de comunicação e sistemas distribuídos, e mais recentemente, começa também a ser utilizado em aplicações multimídia [Blair 93].

I.3.2 Como avaliar uma extensão temporal de LOTOS?

Apresenta-se a seguir um conjunto de características e propriedades relacionadas às álgebras de processos temporizadas - em particular as extensões de LOTOS. A apresentação destas características e propriedades têm por objetivo traçar um perfil dos formalismos a serem analisados, possibilitar a classificação deles e estabelecer um padrão para comparação de extensões temporais de LOTOS. As características e propriedades aqui apresentadas serão divididas em dois grupos: as gerais e as de expressividade.

I.3.2.1 Propriedades e características gerais

As principais características e propriedades gerais que permitirão diferenciar as álgebras de processos temporizadas são apresentadas a seguir:

Formalismo síncrono ou assíncrono Do ponto de vista da realização das ações um formalismo pode seguir uma abordagem síncrona ou assíncrona. Um processo que realiza as suas ações de maneira *assíncrona* pode evoluir em qualquer velocidade; seu progresso não é ditado por um relógio, ou por qualquer outro processo (exceto por uma comunicação voluntária). Em outras palavras, processos assíncronos realizam as suas ações de maneira independente uns dos outros. Geralmente, os processos dependentes do tempo, sobretudo os de aplicações tempo-real possuem, natureza fortemente assíncrona.

Por outro lado, um processo realiza as ações de maneira *síncrona* quando estas seguem em compasso com os "*ticks*" de um relógio

Propriedades temporais gerais: As seguintes propriedades temporais abaixo a serem definidas formalmente na seção II.2.5 devem ser analisadas.

- **Determinismo temporal:** Uma álgebra de processos tem a propriedade de determinismo temporal se para todo processo E, o comportamento resultante é completamente determinado por E e por d, sempre que E não realizar nenhuma ação durante um tempo d.
- **Aditivivade temporal:** Diz-se que uma álgebra de processos tem a propriedade de aditividade temporal se, e somente se, para todo processo E, sempre que E puder ser retardado por $d + d'$, E também poderá ser retardado por d e depois por d' unidades de tempo, e o resultado em ambos os casos deverá ser o mesmo.
- **Implementabilidade:** Uma álgebra de processos possui a propriedade de ser implementável se ela não permitir que processos possam realizar um número infinito de ações em uma quantidade finita de tempo.

Aspectos semânticos:

- **Semântica da sincronização:** A semântica atribuída ao operador de sincronização é um aspecto a ser analisado com cuidado em uma álgebra de processos temporizada; ela pode entre outras características:
  - permitir, ou não, que ações a sincronizar tenham restrições temporais;
  - exigir, ou não, que as ações a sincronizar sejam ocultadas, etc.
- **Tipos de definições da semântica:** A definição da semântica de uma álgebra de processos pode ser realizada utilizando-se de diversas abordagens; por exemplo: operacional *à la Plotkin*, denotacional, axiomática, grafos temporizados, etc. Geralmente, define-se a semântica de uma álgebra de processos através de uma semântica operacional *à la Plotkin* sobre sistemas de transições. Porém, em muitas aplicações pode ser interessante dispor da definição do formalismo através de uma outra abordagem formal. Como será visto mais tarde, grafos temporizados podem, por exemplo, ser utilizados com muitas vantagens para possibilitar a verificação de modelos de lógicas temporais tempo-real.
- **Consistência da semântica operacional:** Visto que a definição da semântica formal de uma álgebra de processos é normalmente a abordagem operacional, então é fundamental que se prove que esta definição semântica seja pelo menos consistente.

Existência de um teorema de expansão: Nem todas as álgebras de processos temporizadas possuem um teorema de expansão [Godskesen 92]. Algumas vezes, a existência de um tal teorema só é garantida pela introdução de operadores de muito alto nível como é o caso em [Yi 91] com o operador de registro de tempo.

I.3.2.2 Propriedades e características de expressividade

Do ponto de vista da expressividade, as principais características a analisar para diferenciar as álgebras de processos temporizadas são:

Ser extensão estrita (ou conservativa) de LOTOS Uma extensão temporal de LOTOS deve garantir que processos não temporizados escritos em LOTOS continuem semanticamente válidos e satisfazendo as mesmas propriedades na extensão temporal como no formalismo básico.

Características e propriedades temporais de expressividade:

- **Tempo denso ou esparso:** Em álgebras de processos temporizadas o tempo pode ser representado de maneira densa (isomorfo aos números racionais, por exemplo) ou de maneira discreta (isomorfo aos números naturais, por exemplo), ou por ambas.

- **Instantaneidade das ações:** As ações em uma extensão temporal de LOTOS podem ser instantâneas ou podem ter durações não nulas associadas a elas. A associação de durações às ações faz com que se perca completamente a possibilidade de se ter um teorema de expansão. Porém, não é garantido obter teoremas de expansão para formalismos temporizados com ações atômicas [Baeten 92]. Para se representar ações com durações é necessário adotar uma semântica de *paralelismo verdadeiro* (como por exemplo a semântica de ordem causal em [Coelho da Costa 92]) que permite representar bem tal situação mas que pode levar a uma semântica operacional complexa.

- **Operadores especiais para mecanismos temporais:** Muitas álgebras de processos temporizadas incorporam operadores especiais para representação de restrições de tempo e de mecanismos temporais básicos como delays, timeouts e watchdogs (um exemplo deste tipo de formalismo é o descrito em [Leduc 92]). Outras álgebras de processos temporizadas não requerem operadores especiais para representação destas características temporais, se satisfazendo com a utilização da semântica de tempo atribuída ao formalismo mais a extensão dos operadores do formalismo básico para incorporação do tempo (um exemplo deste tipo de formalismo é o descrito em [Bolognesi 91]).

- **Tratamento de exceções temporais:** Uma característica interessante e bastante necessária à representação de sistemas tempo-real é a especificação e o tratamento de exceções temporais quando existe falha temporal.

- **Bloqueio do tempo:** Em algums formalismos existem processos especiais de terminação (a exemplo do *stop*) que bloqueiam a progressão do tempo. Tais processos, apesar de um pouco artificiais, parecem ser interessantes do ponto de vista de uma análise de uma especificação pois permitiriam colocar em evidência a situação anormal que teria provocado o bloqueio [Nicollin 92].

- **Urgência de ação:** Existem situações onde é necessário representar uma ação que deve (*must*) ocorrer num instante preciso do tempo. O formalismo que permite esta representação é considerado como tendo urgência nas ações. Em alguns formalismos a urgência é atribuída apenas às ações internas e, neste caso, esta propriedade é conhecida como de atraso mínimo ou progressão máxima[6] das ações internas, ou seja, estas ações ocorrem prioritariamente em relação às ações de progressão do tempo. Em outros formalismos pode-se atribuir urgência apenas em algumas situações escolhidas a priori

[6] Diz-se que um formalismo possui semântica de progresso máximo ou atraso mínimo para uma dada ação se ela ocorre tão logo esteja pronta para tal [Hooman 89, Nicollin 92, Regan 93].

para compatibilizar alguns operadores de LOTOS básico a uma semântica temporal como no caso do operador de ocultação no modelo apresentado em [Leduc 93].

- **Persistência:** Esta propriedade existente em alguns modelos, expressa que a progressão do tempo não pode deixar de oferecer uma ação potencialmente executável. Em outros modelos esta propriedade não existe ou é enfraquecida pela definição da persistência sobre um intervalo de tempo no qual a ação continua sendo oferecida.

No próximo capítulo são apresentadas as definições formais das propriedades apresentadas anteriormente.

I.3.3 Sinopse e comparação de algumas extensões temporizadas de LOTOS

Usando as propriedades e características já definidas nesta seção, passa-se à apresentação de uma breve revisão bibliográfica e comparação das principais extensões temporais de LOTOS. As extensões temporais de LOTOS que serão revistas a seguir, foram selecionadas como as mais significativas apresentadas nos congressos FORTE, PSTV e CFIP nos últimos anos, visto que estes foram os principais foros de debate deste tema. Também, serão considerados apenas as últimas versões de cada abordagem escolhida. Assim, a versão escolhida para o modelo T-LOTOS desenvolvido por Bolognesi, Lucidi e Trigila, no CNUCE/C.N.R. será a contida em [Bolognesi 93] e não as contidas em [Bolognesi 89, Bolognesi 90, Bolognesi 91 ou Bolognesi 92]. Além do modelo T-LOTOS, os outros modelos a serem revistos são: Temporal LOTOS [Regan 93], ET-LOTOS [Leduc 94] e TIC-LOTOS [Azcorra 90].

Temporal LOTOS: A extensão temporal de LOTOS nominada Temporal LOTOS [Regan 93] é uma transposição para LOTOS da álgebra de processos temporizada *Temporal CCS* descrita em [Hennessi 90] (o apêndice B inclui um resumo de Temporal CCS.) Temporal LOTOS é um formalismo síncrono, definido num domínio de tempo discreto e ações instantâneas e não temporizadas. O tempo em Temporal LOTOS é introduzido de forma mínima, isto é, apenas a dimensão do tempo e uma ação σ que representa um *tick* de relógio foram introduzidas nele. Assim, pode-se prefixar um processo com essa ação para se representar um atraso unitário imposto a um processo. Nenhuma estrutura sintática que permita especificar o tempo de forma explícita e quantitativamente é disponível nesse formalismo.

As propriedades de determinismo e aditividade temporal são evidentes para Temporal LOTOS. Porém, como na grande maioria das extensões temporais de LOTOS, esta também não satisfaz a propriedade de implementabilidade.

Do ponto de vista da definição semântica, Temporal LOTOS é muito bem fundamentada; existindo para este formalismo definições semânticas no estilo operacional, denotacional e axiomático. A existência de um teorema de expansão para Temporal LOTOS não é provada em [Regan 93]; contudo, não é difícil de se observar pela simplicidade do modelo de tempo adotado que um tal teorema seja bastante natural no formalismo em questão.

Temporal LOTOS é definida como uma extensão estrita de LOTOS, e apenas um novo operador é definido: o operador "then". Este operador é escrito como $\lfloor P \rfloor (Q)$, para dois processos P e Q, e se comporta da maneira seguinte: inicialmente, as ações de P são oferecidas;

se uma delas é realizada, digamos a, ocorrendo uma transição $P \xrightarrow{a} P'$, então o comportamento $\lfloor P \rfloor(Q)$ evoluirá para P', isto é, ocorrerá a transição $\lfloor P \rfloor(Q) \xrightarrow{a} P'$ desabilitando o operador de composição. Porém, se nenhuma das ações de P ocorrerem em uma unidade de tempo, então o processo P é abandonado e o processo Q é executado. Temporal LOTOS não possui operadores de bloqueio de tempo.

Observa-se que em Temporal LOTOS pode-se atribuir urgência a uma dada ação através da sua ocultação pelo operador *hide*. O seu operador "then" pode ser visto como um tipo operador de tratamento de violações temporais bastante primitivo, mas é este mesmo operador que faz com que a propriedade de persistência não seja satisfeita neste formalismo.

Finalmente, a semântica dada ao operador de sincronização não adota o conceito de semântica de progresso máximo para as ações à sincronizar, mas a adota para as ações à serem ocultadas pelo operador *hide*.

TIC-LOTOS: TIC-LOTOS [Azcorra 90] é uma abreviação para TImed-Calculus for LOTOS e é uma evolução de trabalhos de diversos autores da Universidade de Madri [Quemada 87 e Quemada 89]. Quemada e Fernandez foram os primeiros autores a publicar uma proposta para uma extensão temporal de LOTOS [Quemada 87].

TIC-LOTOS é uma extensão síncrona de LOTOS no qual foi adicionado um modelo de tempo visando a representação de processos temporizados. A extensão foi obtida pela introdução quantitativa do tempo no cálculo algébrico de LOTOS através de restrições sobre a ocorrência de eventos.

O tempo em TIC-LOTOS é discreto, mapeado sobre os inteiros positivos, e global. Ou seja, as expressões de comportamento geram um sistema de transição rotulado, onde os rótulos são eventos temporizados. Os valores associados com um rótulo são contadores de tempo relativos em relação às transições anteriores. Isto eqüivale a existência de um relógio global implícito no sistema de transição. Os sistemas de transições podem ser representados como árvores, e o tempo relacionado com um dado estado pode ser calculado adicionando todos os atributos de tempo do caminho que inicia na raiz da árvore (estado inicial) e termina no estado.

O relógio global é atualizado através de uma função de envelhecimento (passagem de tempo) que atualiza o tempo em cada parte sintática da descrição do comportamento no qual o tempo deve ter passado. Esta função é chamada "*Old*" e tem dois parâmetros, um contador de tempo e um comportamento. $Old(t, B)$ é uma versão do comportamento B que é t unidades de tempo mais "velha". Esta função é necessária quando duas partes de um mesmo comportamento evoluem de modo independente, e quando elas são entrelaçadas.

Algumas das principais características que foram acrescentadas a LOTOS nesta extensão são:

1. uma etiqueta de tempo é atribuída a cada ação para indicar o tempo exato em que ocorreu;

2. as ações são autônomas, isto é ocorrem na estampilha de tempo *timestamp* que lhes são especificadas e

3. um construtor de escolha temporal ("Time Choice") é introduzido como extensão do

operador de prefixação para representar a possibilidade de realização de uma ação dentro de um intervalo de tempo.

No cálculo de TIC-LOTOS, cada evento ocorre em um dado instante do tempo, o qual lhe é justaposto como um valor numérico. Por exemplo, $a3$ representa o evento a que ocorrerá no tempo 3, se nenhum outro ocorreu anteriormente. No cálculo o operador de prefixação ";" é utilizado para prefixar no tempo uma ação a um comportamento. Desta maneira, $a2; b3$ representa que a ocorre duas unidades de tempo após o instante inicial e que b ocorrerá 3 unidades de tempo após o evento a.

Em TIC-LOTOS, existem dois tipos ações: ações normais e a ação invisível i. Todos as ações são temporizadas, tal como o exemplo: $(a2; stop \;|||\; b3; stop)$. Esta composição tem o seguinte significado: o evento a, que *deve ("must")* ocorrer 2 unidades de tempo após o instante inicial, entrelaça com o evento b que *deve* ocorrer 3 unidades de tempo após o instante inicial. Desta discussão observa-se que a expressão acima é equivalente à expressão $(a2; b1)$ (onde $b1$ é o evento $b3$ envelhecido de 2 unidades de tempo).

A semântica do cálculo é definida operacionalmente à la Plotkin e não existem operadores especiais para tratamento de processos temporizados; contudo, o operador de prefixação é estendido para a forma $aT; B$, onde T é um intervalo de tempo, de maneira a possibilitar a ocorrência de uma ação, no caso a, em qualquer instante deste intervalo. Com isto TIC-LOTOS obtém expressividade suficiente para representar algumas situações de *timeouts*. Porém, cabe ressaltar que TIC-LOTOS não pode expressar nem urgência de ações nem de interações.

O formalismo TIC-LOTOS não é uma extensão estrita de LOTOS, visto que o acoplamento das estampilhas de tempo (*timestamps*) às ações lhe rouba esta propriedade. Porém, este formalismo satisfaz as propriedades de determinismo temporal, aditividade temporal, persistência e implementabilidade. Noções de equivalências forte e fraca são definidas e é estabelecido um teorema de expansão para TIC-LOTOS. E por fim, cabe salientar que TIC-LOTOS não possui operadores de bloqueio do tempo nem de tratamento de exceções temporais.

T-LOTOS: A linguagem T-LOTOS (para *Timed LOTOS*) apresentada em [Bolognesi 93] é a evolução dos modelos apresentados em [Bolognesi 90, Bolognesi 91 e Bolognesi 92]. Desenvolvida por Bolognesi, Lucidi e Trigila no CNUCE/C.N.R de Pisa, Itália, T-LOTOS pretende ser uma abordagem unificadora para extensões temporais de LOTOS.

T-LOTOS é um formalismo assíncrono, baseado num domínio de tempo denso e que possui as características seguintes:

- Nenhuma ação é urgente em T-LOTOS: nem as ações ordinárias, nem a ação de término com sucesso e nem a ação interna i são urgentes (ou autônomas). Isto equivale a dizer que as ações devidas à progressão do tempo têm um caráter preemptivo sobre as demais.

- Adota-se uma semântica de árvores com ortogonalidade de tempo/ação e as transições devidas às ações não decorrentes da passagem do tempo em T-LOTOS tem a forma

$$B \xrightarrow{a^t} B'$$

que se lê: o comportamento B realiza agora a ação a de idade (*age*) t e se transforma no comportamento B'.

A idade (tempo) do oferecimento de uma ação é incrementada por transições de passagem de tempo. Assim, em a^t, o t registra exatamente o tempo em que a foi oferecida. (Este tempo é usada para controlar as interações urgentes através do operador "time".)

- Acrescenta a LOTOS dois operadores temporais primitivos: "time" e "time-once".
  - O operador "time $a(t_1, t_2)$ in B" dá a uma ação a em um processo B a possibilidade de ocorrer em um intervalo de tempo (t_1, t_2) com uma semântica equivalente às transições de uma rede de Petri de Temporal [Merlin 76]. Ou seja, a ação a deve ocorrer no intervalo (t_1, t_2) a menos que seja desabilitada pela ocorrência de outra ação.
  - O operador "time – once $a(t_1, t_2)$ in B" possui uma semântica quase idêntica ao operador "time", a única diferença é que ele perde seu efeito após uma primeira aplicação. O operador "time-once" permite definir um conjunto grande de operadores derivados que permitem expressar funcionalidades bastante interessantes. Por este caminho são definidos em T-LOTOS alguns operadores interessantes como:
    * "$a(t_1, t_2); B$" definido como "time – once $a(t_1, t_2)$ in B" e significando que a ação a ocorrerá no intervalo (t_1, t_2) se não for desabilitada;
    * "$a(t); B$" definido como "$a(t, t); B$" e significando que a ação a ocorrerá após um retardo de t unidades de tempo e
    * "urge a in B" definido como "time $a(t_1, t_2)$ in B" e significando que a ação a deve ocorrer em B assim que ela seja possível.

O T-LOTOS separa bem os conceitos de urgência e necessidade de realização de ações. As ações quando especificadas são, normalmente, não urgentes; porém, elas podem tornar-se urgentes se forem submetidas a um operador que imponha este tipo de comportamento (os operadores descritos anteriormente, por exemplo). De maneira análoga, as interações também não são urgentes por natureza, mas podem se tornar urgentes desde que as ações sincronizáveis sejam submetidas a uma operação de "time".

T-LOTOS não possui um Teorema de Expansão, não satisfaz a propriedade de implementabilidade, nada diz a respeito da consistência da sua semântica operacional e não possui operadores de tratamento de violações temporais. Contudo, a linguagem T-LOTOS satisfaz algumas propriedades interessantes; entre elas: ser uma extensão estrita de LOTOS, determinismo temporal, aditividade temporal e persistência.

ET-LOTOS: Em [Leduc 94], Leduc e Léonard da Universidade de Liège, Bélgica, apresentam a linguagem ET-LOTOS (para *Enhanced Timed LOTOS*). ET-LOTOS é uma versão melhorada de três outras apresentadas anteriormente em [Leduc 91, Leduc 92, Leduc 93 e Leduc 93a].

ET-LOTOS é um formalismo assíncrono que incorpora um modelo de tempo denso. Algumas características principais de ET-LOTOS são:

- As ações observáveis e a ação de término com sucesso não são urgentes.

- O modelo semântico subjacente a ET-LOTOS é o de sistemas de transições temporizadas onde o conjunto de ações considerado agrupa as ações ordinárias do formalismo e as ações devidas a progressão do tempo. Resultando assim, em um único tipo de transição que ora representa as ocorrências das ações e ora reflete a progressão do tempo no sistema.

- Em ET-LOTOS existem dois operadores básicos que foram adicionados aos operadores normais de LOTOS: a prefixação temporal e o atraso.

  - O operador de prefixação (incluído em ET-LOTOS na versão apresentada em [Leduc 93]) possui a seguinte forma geral:

    $$a@t\{d\}; E$$

    onde $@t$ e $\{d\}$ são opcionais.

    * O parâmetro $\{d\}$ representa o conceito de *redutor de vida* que quando operado com a ação a (no caso) atribui à prefixação a seguinte semântica: a ação a é oferecida durante um período de tempo d durante o qual ela pode se realizar (ser aceita pelo ambiente), após este período, se a não se realizou, então o processo $a@t\{d\}; E$ se transforma em *stop* silenciosamente. Ou seja, a transição $a@t\{d\}; E \xrightarrow{i} stop$ é realizada.
    * O parâmetro $@t$ representa o registro do tempo de espera até a realização da ação que está associada a ele. Assim, o construtor $a@t; E$ (introduzido inicialmente em [Yi 91]) indica que o tempo relativo no qual a ação a começou a ser oferecida é registrado na váriável t que será utilizada por E.

  - O operador de atraso é denotado por $\Delta^{[d_1,d_2]}$ e permite a imposição de um atraso a um processo; este atraso é um tempo não determinista dentro do intervalo $[d_1, d_2]$. A expiração do retardo é sinalizada na semântica de ET-LOTOS pela ocorrência de uma ação especial θ.

Algumas outras características de interesse de ET-LOTOS são: possuir um operador de bloqueio de tempo, não possuir operadores para tratamento de exceções temporais e assumir a instantaneidade da realização das ações.

Algumas propriedades satisfeitas por ET-LOTOS são: consistência da semântica operacional, determinismo temporal, persistência reversa. Entretanto, não são satisfeitas as propriedades de aditividade temporal, persistência e implementabilidade. ET-LOTOS também possui um teorema de expansão e é uma extensão estrita de LOTOS.

Outrossim, não pode-se deixar de constatar as limitações da semântica dada às sincronizações de ações em ET-LOTOS. Neste formalismo, a sincronização da ação a no processo $a; E|[a]|a; F$ pode ser postergada indefinidamente, e não existe maneira de impor uma urgência à esta sincronização como no caso de T-LOTOS.

I.3.3.1 Comparação dos formalismos

A tabela I.3 apresenta uma comparação entre as extensões temporais de LOTOS vistas anteriormente.

Modelos	Temporal LOTOS	TIC-LOTOS	T-LOTOS	ET-LOTOS
Referência	[Regan 93]	[Azcorra 90]	[Bolognesi 93]	[Leduc 94]
Domínio do tempo	Esparso	Esparso	Denso	Denso
Modelo Computacional Síncrono / Assíncrono	Síncrono	Síncrono	Assíncrono	Assíncrono
Existência de um Teorema de Expansão	Sim	Sim	Não	Sim
Determinismo Temporal	Sim	Sim	Sim	Sim
Aditividade Temporal	Sim	Sim	Sim	Não
Implementabilidade	Não	Sim	Não	Não
Persistência	Não	Sim	Sim	Não (só persitência reversa)
Extensão Estrita de LOTOS	Sim	Não	Sim	Sim
Ações Instantâneas	Sim	Sim	Sim	Sim
Urgência -Ação -Sincronização -Política	 Internas Nas ações ocultadas por "hide" Ação interna urgente	 Todas Todas Ações são autônomas, exceto quando relaxadas por "$aT; E$"	 Nas ações com "time" Nas sincronizações com "time" Nenhuma ação é urgente	 Internas Nas ações ocultadas por "hide" Ação interna urgente
Ações Especiais	σ (retardo unitário)	não tem	não tem	θ (atraso não determinista)
Operadores Especiais para o Tempo	$\lfloor E \rfloor(F)$	$aT; E$	time $a(t_1, t_2)$ in B, time−once $a(t_1, t_2)$ in B	$\Delta^{[d_1,d_2]}$, $a@t\{d\}$
Tratamento de Exceções Temporais	sim, mas bastante primitivo	Não	Não	Não
Bloqueio do tempo	Não	Não	Sim	Sim
Atraso	$\sigma; E$	$it; E$	$i(t_1, t_2); E$	$\Delta^{[d_1,d_2]}E$
Timeout	$\lfloor \sigma; E \rfloor(F)$	$it; E[]F$	$i(t); E[]F$	$\Delta^{[d_1,d_2]}i; E$
Watchdog	$F[> \sigma; E$	$F[> it; E$	$F[> i(t); E$	$F[> \Delta^{[d_1,d_2]}i; E$

Tabela I.3: Comparação das principais características das extensões temporais de LOTOS

Resumindo esta seção mostra-se na figura I.3 uma visão global comparativa das principais extensões de LOTOS do ponto de vista da expressividade versus a complexidade semântica. Esta figura foi construída a partir de uma avaliação qualitativa do autor.

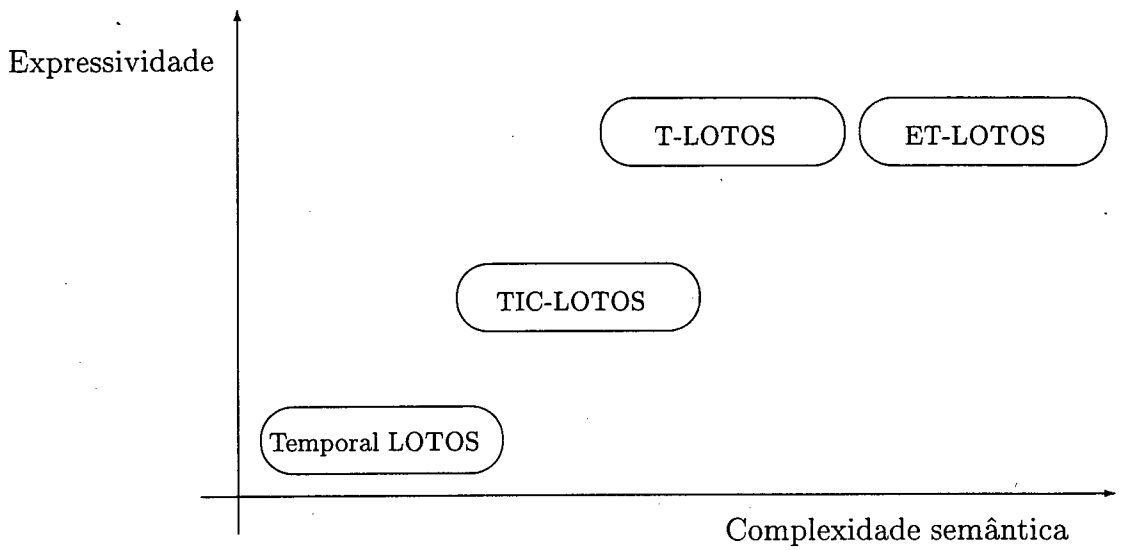


Figura I.3: Expressividade versus complexidade semântica para LOTOS temporais

Uma comparação da proposta apresentada neste trabalho com aquelas discutidas aqui será apresentada no último capítulo.

I.4 Conclusão

Neste capítulo foram apresentados alguns conceitos básicos referentes à especificação formal de sistemas dependentes do tempo. Inicialmente, apresentou-se conceitos básicos associados ao tempo e a sua especificação formal. Depois, tratou-se de apresentar as características básicas e os requisitos das aplicações consideradas neste trabalho, quais sejam: sistemas tempo-real, protocolos de comunicação e sistemas multimídia. Em seguida, visitou-se as principais abordagens existentes para especificação de sistemas dependentes do tempo. Neste ponto, justificou-se a opção feita pela linguagem LOTOS e apresentou-se as principais características das extensões temporais deste formalismo. Finalmente, revisou-se as principais propostas existentes na literatura que abordam o mesmo problema tratado neste trabalho e apresentou-se um resumo comparativo das características destas propostas.

O próximo capítulo é dedicado à apresentação da álgebra de processos RT-LOTOS para especificação de sistemas dependentes do tempo. Lá, será apresentada a intuição que deu origem ao formalismo básico e sua sintaxe e semântica operacional *à la Plotkin*. Apresenta-se também uma semântica formal alternativa em grafos temporizados de RT-LOTOS, bem como inúmeros exemplos de representação de processos temporizados em RT-LOTOS.

Capítulo II

RT-LOTOS: O Formalismo Básico

Neste capítulo é apresentada a álgebra de processos RT-LOTOS para especificação de sistemas dependentes do tempo. O trabalho apresentado nas próximas seções é uma evolução natural de outras versões de extensões temporais de LOTOS propostas pelo autor em trabalhos anteriores [Camargo 90, Camargo 91, Camargo 92, Courtiat 93, Courtiat 93b] e pode ser também visto como uma visão unificada das propostas descritas em [Camargo 93, Camargo 94].

Este capítulo está organizado da seguinte maneira. Primeiramente, na seção II.1 apresenta-se a linguagem RT-LOTOS básica através de uma visão intuitiva e da descrição da sua sintaxe e da descrição informal de seus operadores. Depois, na seção II.2 a álgebra de processos RT-LOTOS é definida formalmente; apresenta-se: uma semântica operacional *à la Plotkin*, um conjunto de propriedades que o formalismo satisfaz, e define-se algumas bissimulações e estabelecem-se propriedades satisfeitas por estas. Na seção II.3 são apresentados alguns exemplos de especificações em RT-LOTOS. Na seção II.4, apresenta-se uma definição semântica de RT-LOTOS a partir do formalismo grafos temporizados. E finalmente, a seção II.5 apresenta algumas conclusões pertinentes a este capítulo.

II.1 Apresentação do Formalismo

II.1.1 A visão intuitiva de RT-LOTOS

Nesta seção é apresentada a visão intuitiva que serviu de base para propor uma nova extensão temporal para a linguagem LOTOS [ISO 88] que atende aos requisitos apontados no capítulo anterior. A extensão aqui proposta é chamada de RT-LOTOS, abreviação para Real-Time LOTOS, tendo em vista as características que ela incorpora para especificação de sistemas tempo-real e outros sistemas dependentes do tempo. Para manter-se em conformidade com a estrutura básica definida para Basic LOTOS [Bolognesi 87], chamaremos *Act* o conjunto das ações observáveis e i a ação interna.

Uma das características chave de RT-LOTOS é a atribuição de intervalos de tempo às ações como forma de estipular quando as ações podem ser realizadas, ou seja, impõem-se restrições temporais às ocorrências das ações. Para compreender bem essa característica faz-se uma breve descrição da intuição básica que guiou essa forma de definir as restrições

de tempo em RT-LOTOS.

Originalmente, a linguagem de especificação LOTOS foi concebida de maneira a se abstrair de qualquer aspecto temporal. Assim, uma prefixação de uma ação a a um processo E, escrita como

$$a; E$$

significa que quando o ambiente eventualmente estiver pronto para realizar a ação a, então ela ocorrerá. Porém, se uma dimensão temporal também é observada, pode-se dizer que isto é semanticamente equivalente a dizer que a ação a ocorrerá num instante de tempo qualquer dentro do intervalo temporal $[0, \infty]$ (mais uma hipótese de *fairness* que asseguraria que a ação eventualmente ocorreria). Entretanto, existem situações nas quais a ação a não poderá ser realizada no intervalo $[0, \infty]$ mais num intervalo $[t_{\min}, t_{\max}]$ definindo-se assim o conceito de *intervalo de possibilidade de realização de uma ação.* Em RT_LOTOS este conceito é implementado pela atribuição de um intervalo de tempo a uma ação para *limitar* a possibilidade de sua ocorrência a este intervalo. Por exemplo, em RT_LOTOS o comportamento

$$[t_{min}, t_{max}]a; E$$

representa uma prefixação em que a ação a só pode ocorrer no intervalo temporal $[t_{min}, t_{max}]$. Em outras palavras, estes intervalos determinam em que instante as diferentes ações podem ser oferecidas ao seu ambiente. No caso de nenhum intervalo ser associado explicitamente à uma dada ação observável, faz-se a hipótese de que por *default* o intervalo $[0, \infty]$ está associado a esta ação.

Entretanto, o fato de associar um tempo máximo t_{max} à uma ação não implica que se deseja forçar a realização de uma tal ação temporizada neste instante, pois procurou-se manter o paradigma das álgebras de processos no qual a aceitação de uma ação observável depende de seu ambiente. Em RT-LOTOS, objetiva-se indicar que se a ação $[t_{min},t_{max}]a$ não pôde se realizar durante o intervalo especificado, então ela não poderá o ser fora deste intervalo. Para poder caracterizar esta situação, define-se um conjunto de ações especiais, chamado Act^*. Estas ações especiais, chamadas violações temporais, têm por objetivo notificar, quando da execução de uma especificação, que ações temporizadas não puderam se realizar dentro dos intervalos que lhes foram associados. Assim, chama-se a^* a ação especial caracterizando a violação temporal associada à não realização de uma certa ação temporizada $[t_{min}, t_{max}]a$. O interesse de poder dispor destas ações especiais reside na possibilidade de representar em uma especificação tratamentos de exceção à serem realizados quando da ocorrência de tais violações temporais. Para realizar isso, introduz-se um novo operador, chamado de operador de preempção temporal, destinado a exprimir os tratamentos de exceção desejados.

Como será mostrado nas próximas seções, a introdução dos intervalos de tempo em RT-LOTOS permitiu representar as principais restrições de tempo e mecanismos temporais necessários à especificação de sistemas tempo-real, bem como possibilita a definição de um operador de tratamento de exceções temporais bastante útil.

Em resumo, pode-se afirmar que a definição de RT-LOTOS baseia-se nas seguintes modificações feitas a LOTOS:

- Introduzir intervalos temporais nas prefixações de ações e
- Adicionar um operador especial para tratamento de exceções temporais.

Então, a prefixação de uma ação a para um processo E, quando intervalo de possibilidade de ocorrência é limitado ao intervalo de tempo $[t_{min}, t_{max}]$ se escreve $[t_{\min}, t_{\max}]a; E$ e sua semântica informal é:

- Espera até $t_{\min}$. Apenas a passagem do tempo ocorre durante este tempo.

$$[t_{\min}, t_{\max}]a; E \not\rightarrow$$

- Entre $[t_{\min}, t_{\max}]$ a ação a *pode* ocorrer.

$$[t_{\min}, t_{\max}]a; E \xrightarrow{a} E$$

- Se em $t_{\max}$ a ação a ainda não ocorreu, então neste instante ocorrerá de maneira urgente e incontrolável uma ação interna, chamada de a^*, que notifica internamente ao sistema a impossibilidade de ocorrência da ação a, ou seja, a violação da restrição temporal imposta à ocorrência da ação a.

$$[t_{\min}, t_{\max}]a; E \xrightarrow{a^*} stop$$

Outrossim, as ações de violação temporal podem ser tratadas através do operador de tratamento de violações temporais cuja sintaxe é dada por

$$E < a_1, \cdots, a_n] \{a_1 : Q_1, \cdots, a_n : Q_n\}$$

e sua semântica informal pode ser entendida como:

- Se no processo E não ocorrem ações de violações temporais a_i^* para algum $a_i \in \{a_1, \cdots, a_n\}$, então o comportamento do processo composto é idêntico ao processo E.

- Por outro lado, se durante a realização de E ocorrer algum ação de violação temporal a_i^* com $a_i \in \{a_1, \cdots, a_n\}$, então no exato instante da realização desta ação a execução do processo E será abortada e o processo Q_i será lançado no seu lugar.

Ainda, como poderá ser observado na descrição formal da semântica de RT-LOTOS, permite-se também a ocultação de ações observáveis temporizadas. Para manter a coerência do formalismo, optou-se por permitir a associação de restrições temporais também à ação interna i sem que seja aumentada a complexidade da semântica. Todavia, devem ser notadas com relação à temporização da ação i duas diferenças fundamentais que caracterizam a urgência da ação interna i:

- na falta de uma temporização explícita da ação interna i ela é considerada como sendo temporizada pelo intervalo temporal $[0, 0]$, isto é, a ação i ocorre imediatamente (e incontrolavelmente) quando do seu oferecimento,

- além disso não é definida violação temporal associada à $[t_{min}, t_{max}]i$, dado que i deve, necessariamente, realizar-se no intervalo $[t_{min}, t_{max}]$.

Finalmente, procurou-se definir RT-LOTOS como uma extensão estrita de Basic LOTOS. Para convencer-se disso é suficiente associar intervalos temporais $[0, \infty]$ às ações observáveis e de não mais considerar, na semântica operacional, as regras que definem a progressão do tempo. Além disso, o fato de considerar intervalos temporais $[0, \infty]$ para todas as ações tornam sem nenhum efeito as regras da semântica operacional que tratam das violações temporais.

II.1.2 Descrição da sintaxe e dos operadores de RT-LOTOS

II.1.2.1 Sintaxe de RT-LOTOS

Os termos de RT-LOTOS são gerados pela sintaxe apresentada a seguir. Observe-se que esta é uma extensão direta da sintaxe de Basic LOTOS.

$E ::=$	$stop$	(* inação *)
|	$exit$	(* terminação com sucesso *)
|	$[t_{min}, t_{max}]a; E$	(* prefixação: ação observável *)
|	$[t_{min}, t_{max}]i; E$	(* prefixação: ação interna *)
|	$E[\,]E'$	(* escolha *)
|	$E\|[L]\|E'$	(* composição paralela *)
|	$hide\ L\ in\ E$	(* ocultação *)
|	$E >> F$	(* composição seqüencial *)
|	$E\ [> F$	(* preempção *)
|	$E <L]\ \{a_1{:}Q_1, \cdots, a_n{:}Q_n\}$	(* preempção temporal *)
|	$P[a_1, \cdots, a_n]$	(* instanciação de processos *)

O símbolo $\mathcal{E}$ será usado para denotar o conjunto dos termos gerados pela sintaxe de RT_LOTOS. Além disso, cometer-se-á, na continuação do texto, o abuso de linguagem clássico confundindo expressões de comportamento e processos para qualificar os termos gerados por esta sintaxe.

II.1.2.2 Descrição informal dos operadores de RT-LOTOS

A descrição informal dos termos de RT-LOTOS é dada a seguir.

"*stop*" representa um processo que não faz nada, exceto deixar o tempo progredir.

"*exit*" representa o processo de terminação com sucesso clássica de LOTOS. A ação δ da semântica de *exit* não é urgente.

"$[t_{min}, t_{max}]a; E$" representa um processo no qual a ação oferecida a só pode ser realizada dentro do intervalo de tempo $[t_{min}, t_{max}]$; antes de t_{min} a ação sofre um atraso deixando o tempo apenas progredir, e depois de t_{max} o oferecimento da ação "caduca", deixando de existir. Se no intervalo de tempo a ação a é realizada, então $[t_{min}, t_{max}]a; E$ passa a se comportar como E. Se a ação a ficou oferecida até o instante t_{max} e não foi realizada, então ocorre a violação temporal a^* neste instante e o processo passa a se comportar como o processo *stop*.

"$E[\,]F$" representa a escolha entre os processos E e F. O processo se comporta como E ou como F, com a escolha sendo realizada no instante da realização de uma primeira ação $a \in Act \cup \{i, \delta\}$; isto significa que a realização de uma ação de violação temporal a^* não decide uma escolha.

"$E|[L]|F$" representa o operador de composição paralela de dois processos. Cada um dos dois processos pode realizar, de modo independente, as ações que não pertençam ao conjunto L, sendo que para as ações no conjunto L os processos devem se sincronizar. As

sincronizações ocorrem desde que elas sejam possíveis, isto é, se ambos os processos oferecem uma ação para se sincronizar, então esta ação é realizada imediatamente. Ou seja, estamos atribuindo uma semântica de progresso máximo às *ações sincronizáveis*. No caso de um dos dois processos não oferecer a ação sincronizável até a expiração do intervalo de tempo da ação oferecida ocorrerá uma ação de violação temporal associada a ação cujo intervalo temporal expirou; o comportamento subseqüente continuará sendo a composição paralela do processo que segue a violação temporal com o processo que não ofereceu a ação à sincronizar.

"hide L in E" representa um processo que transforma as ações observáveis em L em invisíveis (e urgentes).

"$E >> F$" representa a composição seqüencial de dois processos E e F. Sua interpretação informal é que se o processo E termina com sucesso, e não por causa de um deadlock prematuro, então a execução do processo F é possível.

"$E [> F$" representa um processo que se comporta como E, mas que pode ser interrompido a qualquer instante pela realização de uma ação de F. Se E termina sem que o processo F comece, então "$E [> F$" também termina. Mas, se uma ação de F ocorre antes do término de E, então a execução do processo E é abandonada e o controle passa para o processo F.

"$E <L] \{a_1 : Q_1, \cdots, a_n : Q_n\}$" representa o operador de preempção temporal. Este operador possui aridade $n+1$ onde $n < \infty$ é a cardinalidade do conjunto L, i.e. $|L| = n$. Os "$a_i : Q_i$" têm o seguinte propósito: os "a_i" são rótulos identificadores e "Q_i" são processos; "a_i" indica que é o processo "Q_i" que tratará uma eventual violação temporal "a_i^*". Para simplificar a explanação considere-se o processo $P = E <L] \{a_1 : Q_1, \cdots, a_n : Q_n\}$. Assim, o processo P começa pela execução de E. Se E termina sem ocorrências de violações temporais a^* tal que $a \in L$, então o processo P termina. Mas, se durante a execução de E ocorrer uma violação temporal a_i^* com $a_i \in L$, então a execução do processo E é abandonada e o processo Q_i é iniciado no seu lugar.

$P[a_1/a_1', \cdots, a_n/a_n']$ representa uma instanciação do processo $P[a_1', \cdots, a_n']$ em que as ações a_i' têm seu nome trocado para a_i, para $i = 1, \cdots, n$.

II.2 A álgebra de processos RT-LOTOS

Esta seção é dedicada à definição formal da álgebra de processos RT-LOTOS.

Uma álgebra de processos é definida segundo [Nicollin 92] por uma quádrupla $AP = (OP, \mathcal{L}, R_{\mathcal{L}}^{OP}, \sim)$ onde

- OP é um conjunto de operadores definindo a linguagem da álgebra de processos
- $\mathcal{L}$ é um conjunto de nomes de transições
- $R_{\mathcal{L}}^{OP}$ é um conjunto de regras da semântica operacional estrutural *à la Plotkin* [Plotkin 81] associando os termos da álgebra de processos com sistemas de transições rotuladas em $\mathcal{L}$ (modelos)
- $\sim \subseteq AP \times AP$ é uma equivalência comportamental sobre modelos.

No caso de uma álgebra de processos temporizada aumenta-se o conjunto $\mathcal{L}$ unindo-se a ele os elementos de um domínio de tempo D, obtendo-se um conjunto de ações estendido $\mathcal{L} \cup D$. Definem-se, para cada operador da álgebra, regras de inferência rotuladas por elementos deste domínio de acordo com a semântica que se quer atribuir à álgebra temporizada.

Os elementos que compõem a álgebra de processos RT-LOTOS são definidos nas próximas seções na seguinte ordem. Inicialmente, define-se o conceito de domínio temporal e as hipóteses adotadas em RT-LOTOS para este domínio. Depois, são definidos os conceitos básicos para definição da semântica operacional: sistemas de transições rotuladas temporizadas e o conjunto de ações de RT-LOTOS. A seguir são apresentadas as regras da semântica operacional estrutural de RT-LOTOS, algumas propriedades satisfeitas, e finalmente algumas bissimulações desenvolvidas para RT-LOTOS, dentre elas uma congruência.

II.2.1 O Domínio de Tempo Considerado

II.2.1.1 Definições

Um domínio de tempo é um monóide comutativo[1] $(D, +, 0)$ que satisfaz as condições seguintes:

1. $d + d' = d \Leftrightarrow d' = 0$
2. a pré-ordem "$\leq$" definida por $d \leq d' \Leftrightarrow \exists d'' \in D$ tal que $d + d'' = d'$ é uma ordem parcial bem definida.

As propriedades seguintes são válidas para um domínio de tempo D:

- 0 é o menor elemento de D.
- $\forall d, d' \in D$, se $d \leq d'$, então o elemento $d'' \in D$ tal que $d + d'' = d'$ é único, e é denotado por $d' - d$.

D é chamado *denso* se $\forall d, d' \in D : d < d' \Rightarrow \exists d'' \in D : d < d'' < d'$.

D é chamado *esparso* se $\forall d \in D \quad \exists d' \in D : (d < d') \wedge (\forall d'' \in D : d < d'' \Rightarrow d' \leq d'')$. Como a pré-ordem "$\leq$" é total, o elemento d' é único e é chamado o *sucessor* de d, denotado por $succ(d)$. Uma importante propriedade que os domínios esparsos satisfazem é: $\forall d,\ succ(d) = d + succ(0)$; o que quer dizer que qualquer elemento de D pode ser obtido de 0 adicionando $succ(0)$ o quanto for necessário.

Denota-se por D^ω o conjunto $D \cup \{\omega\}$, onde ω é um elemento não definido em D tal que $\forall d \in D$, $d < \omega$ e $\omega + d = \omega$.

As definições acima foram resumidas de [Nicollin 92]

[1]Monóide: conjunto munido de uma lei de composição interna associativa e com um elemento neutro. Além disso, se $(A, \bullet, 1)$ é um monóide e $\forall a, b \in A$ satisfazer $a \bullet b = b \bullet a$, então diz-se que $(A, \bullet, 1)$ é um monóide comutativo. Por exemplo, o conjunto dos números naturais munido da adição é um monóide comutativo.

II.2.1.2 O domínio de tempo em RT-LOTOS

Em RT-LOTOS, nem a sintaxe, nem a semântica dependem de um domínio de tempo particular. A única hipótese feita aqui é que o domínio de tempo seja enumerável o que garante que o modelo semântico subjacente seja um Sistema de Transições Rotuladas. Procedendo desta maneira, pode-se considerar tanto domínios de tempo *esparsos* (números inteiros naturais) quanto *densos* (números racionais).

As hipóteses feitas sobre a densidade ou não do domínio de tempo adotado para a álgebra de processos temporizada são importantes e refletem-se diretamente em propriedades e características do formalismo resultante.

Para compreender melhor a diferença semântica resultante da escolha de um domínio de tempo particular, considere-se o processo:

$$a; (b; [0]c \ [\,] \ [1]i; R) \ [\,] \ [1]i; R$$

Para os domínios de tempo dos números naturais, $\mathbf{N}$ e dos racionais, $\mathbf{Q}$, as evoluções possíveis deste processo são (observe-se que $\mathbf{N} \subset \mathbf{Q}$):

- Com o domínio de tempo $\mathbf{N}$, a pode ocorrer somente antes do tempo 1, isto é, no tempo 0; então b pode ocorrer também somente antes do tempo 1, e finalmente c é realizada imediatamente após b. Assim, c pode somente ocorrer antes do tempo 1 (ou seja, no tempo 0).

- Com o domínio de tempo $\mathbf{Q}$, a pode ocorrer antes do tempo 1, e isto inclui quaisquer valores racionais inferiores a 1. Também, b pode ser realizada até o tempo inferior a 1 após a realização de a; assim, b pode potencialmente ser realizada num tempo $0 \leq t_b < 2$. Ou seja, c pode ser realizada em qualquer tempo inferior a 2 desde que a ocorra num tempo inferior a 1.

O exemplo acima ilustra a situação em que tem-se dois domínios de tempo D e D' tal que $D \subset D'$ e que o comportamento temporal de um processo P em D, em símbolos $(P)_D$, é diferente de P em D', $(P)_D \neq (P)_{D'}$. Esta anomalia é uma situação geral para álgebras de processos temporizadas que admitem tempo denso. (Nicollin e Sifakis [Nicollin 92] consideram a igualdade da expressão anterior como uma importante propriedade a ser satisfeita por álgebras de processos temporizadas, pois garante que o formalismo seja imune às diferentes hipóteses sobre o domínio de tempo.)

II.2.2 Sistemas de Transições Rotuladas - o caso temporizado

Uma das maneiras mais usuais de descrever a semântica formal de uma álgebra de processos é através da atribuição de uma semântica operacional estrutural (*à la Plotkin*) [Plotkin 81]. Em resumo, uma definição de uma semântica operacional estrutural consiste na construção de um *sistema de transições rotuladas* que descreve o comportamento dos operadores da álgebra de processos a ser definida através de regras de transições baseadas na estrutura sintática dos operadores. Desta forma, o formalismo básico subjacente às álgebras de processos definidas operacionalmente, é o de sistemas de transições.

Um sistema de transições rotuladas por um alfabeto $\mathcal{L}$ é definido como em [Bolognesi 87] por uma quadrupla $\langle \mathcal{E}, \mathcal{L}, E_0, \rightarrow \rangle$, onde

$\mathcal{E}$ é um conjunto enumerável de *estados*;

E_0 é o *estado inicial*;

$\mathcal{L}$ é um conjunto enumerável de nomes de *ações* e

$\rightarrow \subseteq \mathcal{E} \times \mathcal{L} \times \mathcal{E}$ é uma relação chamada de *relação de transição.*

Quando o estado inicial de um STR for evidente, este será omitido, e usa-se também a notação $E \xrightarrow{a} E'$, com $a \in \mathcal{L}$ e $E, E' \in \mathcal{E}$, para denotar $(E, a, E') \in \rightarrow$. Um STR é dito *finito*, se ambos $\mathcal{E}$ e $\mathcal{L}$ forem finitos.

Em geral, para se definir uma semântica operacional estrutural de uma álgebra de processos temporizada procede-se da maneira que segue. Adiciona-se ao conjunto $\mathcal{L}$ elementos de um domínio de tempo D que deseja-se associar à álgebra, obtendo-se um conjunto de ações estendido $\mathcal{L} \cup D$. Em seguida, definem-se, para cada operador da álgebra, regras de inferência rotuladas por elementos deste domínio de acordo com a semântica que se quer atribuir à álgebra temporizada. Feito desta maneira, é necessário assumir como hipótese básica que o domínio de tempo considerado seja *enumerável* para que o sistema de transições rotuladas subjacente à esta álgebra de processos seja bem definido.

II.2.3 As Ações de RT-LOTOS

As ações de RT-LOTOS podem ser agrupadas em dois grandes conjuntos:

- as ações clássicas de LOTOS, a saber as *ações observáveis* pertencentes ao conjunto Act, a *ação interna* i e a *ação de término com sucesso* δ., Os conjuntos de ações são definidos da seguinte maneira: $Act^i = Act \cup \{i\}$, $Act^\delta = Act \cup \{\delta\}$ e $Act^{i,\delta} = Act \cup \{i\} \cup \{\delta\}$.
- as ações específicas de RT-LOTOS que são *violações temporais* a^* que pertencem ao conjunto Act^*, sabendo que existe uma bijeção entre Act e Act^*.

RT-LOTOS permite temporizar as ações observáveis e também a ação interna. A ação de terminação com sucesso δ e as violações temporais a^*, sendo devidas a ocorrências de operações especificadas na linguagem mas não sendo diretamente manipuláveis nesta, não podem obviamente ser temporizadas.

Seja a uma ação pertencendo ao conjunto Act^i. RT-LOTOS permite temporizar uma tal ação escrevendo $[t_{min}, t_{max}]a$ com t_{min} e t_{max} $(t_{min} \leq t_{max})$ pertencendo ao domínio de tempo D^ω considerado.

t_{min}, é o limite inferior do intervalo de tempo, e caracteriza o retardo mínimo durante o qual a ação a deve ser apresentada (ou sensibilizada) antes de ser realmente oferecida a seu ambiente. A partir do cumprimento deste retardo, a ação pode se realizar durante um intervalo de tempo igual a $[0, t_{max} - t_{min}]$. No final deste intervalo de tempo o comportamento será diferente dependendo do tipo da ação (se observável a ou interna i). Se a é uma ação observável, a não ocorrência de a dentro do intervalo estabelecido provocará a realização de uma ação especial, a ação a^* caracterizando a violação do intervalo de tempo associado a a. De outra maneira, se a é a ação interna i, então a progressão do tempo é suspensa até a

realização de i ou de uma outra ação oferecida no mesmo instante e que esteja em conflito com i como é no caso do operador de escolha.

No que diz respeito a semântica das ações de RT-LOTOS, constata-se que as ações observáveis só podem realizar-se durante o intervalo de tempo que lhes é associado, enquanto a violação temporal é autônoma e urgente por definição. Além disso, a ação i torna-se "*urgente*" quando da expiração de seu intervalo de tempo, enquanto a ação δ continua não urgente por definição.

Além disso, e em conformidade com a semântica padrão de LOTOS, todas as ocorrências de ações são consideradas atômicas e instantâneas.

II.2.4 Semântica Operacional de RT-LOTOS

A semântica operacional é apresentada no estilo SOS (*Structured Operational Semantics*) de Plotkin [Plotkin 81] e compreende:

- um conjunto de regras de inferência para as ações clássicas de RT-LOTOS
- um conjunto de regras de inferência para as violações temporais
- um conjunto de regras de inferência para a progressão do tempo

Antes de detalhar a semântica operacional de RT-LOTOS, serão introduzidas uma função para auxílio das definições das regras semânticas e algumas notações.

II.2.4.1 Uma Função Auxiliar

Definimos a seguir a função "$\hookrightarrow$" que estabelece formalmente em que circunstâncias sintáticas um processo está habilitado para evoluir pela realização de uma dada ação.

Definição 5 A função $\hookrightarrow$: $\mathcal{E} \times Act^{i,\delta} \cup Act^* \mapsto \{tt, ff\}$ (onde tt e ff representam os valores lógicos verdadeiro e falso, respectivamente) representa um predicado que denota a possibilidade de um processo evoluir pela realização de uma dada ação $a \in Act^{i,\delta} \cup Act^*$. $\hookrightarrow$ é definido por:

$\hookrightarrow (stop, a) = ff$

$\hookrightarrow (exit, a) = \begin{cases} tt, & \text{se } \delta = a \\ ff, & \text{senão} \end{cases}$

$\hookrightarrow ([t_{min}, t_{max}]b; E, a) = \begin{cases} tt, & \text{se } b = a \text{ e } t_{min} = 0 \\ ff, & \text{senão} \end{cases}$

$\hookrightarrow (E[\,]F, a) = \hookrightarrow (E, a) \vee \hookrightarrow (F, a)$

$\hookrightarrow (E|[L]|F, a) = \begin{cases} \hookrightarrow (E, a) \wedge \hookrightarrow (F, a), & \text{se } a \in L \\ \hookrightarrow (E, a) \vee \hookrightarrow (F, a), & \text{senão} \end{cases}$

$\hookrightarrow (E|||F, a) = \hookrightarrow (E, a) \vee \hookrightarrow (F, a)$

$\hookrightarrow (E||F, a) = \hookrightarrow (E, a) \wedge \hookrightarrow (F, a)$

$\hookrightarrow (\text{hide } L \text{ in } E, a) = \begin{cases} \hookrightarrow (E, a), & \text{se } a \notin L \\ ff, & \text{senão} \end{cases}$

$\hookrightarrow (E >> F, a) = \hookrightarrow (E, a)$

$\hookrightarrow (E <L] \{a_1 : Q_1, a_2 : Q_2, \cdots, a_n : Q_n\}, a) = \hookrightarrow (E, a)$

$\hookrightarrow (E\ [> F, a) = \hookrightarrow (E, a) \vee \hookrightarrow (F, a)$

$\hookrightarrow (P, a) = \hookrightarrow (B_p, a)$ se $P := B_p$ é uma definição de processo. □

II.2.4.2 Notação

Doravante, para simplificar o texto, utilizaremos:

- $E \xrightarrow{a}$ significando $\hookrightarrow (E, a) = tt$, ou seja, que existe em E uma ação a pronta a se realizar.
- $E \not\xrightarrow{a}$ significando $\hookrightarrow (E, a) = ff$, ou seja, que não existe em E uma ação a que esteja pronta a se realizar.
- $[t]$ significando $[t, t]$ para $t \in D^\omega$
- i significando $[0, 0]i$
- $a; E$ significando $[0, \omega]a; E$
- $E|||F$ significando $E|[\emptyset]|F$ (entrelaçamento total das ações)
- $E||F$ significando $E|[Act]|F$ (sincronização total das ações)
- $E < a]F$ denotando o caso particular de $E < a]\{a : F\}$

II.2.4.3 Regras Semânticas dos Operadores

Operador de Inação O operador *stop* não bloqueia a progressão do tempo, o que é traduzido pela regra seguinte:

$$\frac{-}{stop \xrightarrow{t} stop} \quad (t \in D^\omega)$$

Operador de Terminação com Sucesso

1. Comportamento

$$\frac{-}{exit \xrightarrow{\delta} stop}$$

2. Progressão do tempo

$$\frac{-}{exit \xrightarrow{t} exit} \quad (t \in D^{\omega})$$

O axioma 2 expressa a não urgência da ação δ.

Operador de Prefixação

1. Comportamento

(a) ações clássicas

$$\frac{-}{[0,t]a; E \xrightarrow{a} E} \quad (t \in D^{\omega}) \quad (a \in Act^{i})$$

(b) violações temporais

$$\frac{-}{[0,0]a; E \xrightarrow{a^*} stop} \quad (a \in Act)$$

A regra (a) significa que a ação a pode ser realizada durante toda a duração do intervalo temporal $[0,t]$; A regra (b) tem por objetivo exprimir uma violação temporal, denotada a^*, indicando que a ação a não pôde ser realizada dentro das restrições temporais impostas. O processo resultante torna-se *stop*, e no caso da necessidade de fazer sentir este fato dentro da especificação, essa violação temporal pode ser tratada por um mecanismo de exceção *ad hoc* através do operador $E < L]\,\{a_1 : Q_1, a_2 : Q_2, \cdots, a_n : Q_n\}$, onde $a \in L = \{a_1, \cdots a_n\}$ que será tratado a seguir.

2. Progressão do tempo

(a)

$$\frac{-}{[0,t+s]a; E \xrightarrow{s} [0,t]a; E} \quad (t,s \in D^{\omega} \text{ e } s > 0) \quad (a \in Act^{i})$$

(b)

$$\frac{-}{[t_1+s, t_2+s]a; E \xrightarrow{s} [t_1,t_2]a; E} \quad (s,t_1,t_2 \in D^{\omega} \text{ e } s > 0) \quad (a \in Act^{i})$$

Observe que a ação i pode também ser temporizada. Tão logo o intervalo de tempo associado com ela torna-se $[0,0]$, ela torna-se uma ação urgente; como conseqüência, não existe violação temporal associada com a ação i.

Operador de Escolha

1. Comportamento

(a) ações clássicas

$$\frac{E \xrightarrow{a} E'}{E[\,]F \xrightarrow{a} E' \quad F[\,]E \xrightarrow{a} E'} \quad (a \in Act^{i,\delta})$$

(b) violações temporais

$$\frac{E \xrightarrow{a^*} E'}{E[\,]F \xrightarrow{a^*} E'[\,]F \quad F[\,]E \xrightarrow{a^*} F[\,]E'} \quad (a \in Act)$$

A regra (b) estipula que se uma violação temporal afeta o processo E (i.e. E oferece a ação a^*), então esta violação temporal *não* resolve a escolha.

2. Progressão do tempo

$$\frac{E \xrightarrow{t} E', \quad F \xrightarrow{t} F'}{E[\,]F \xrightarrow{t} E'[\,]F'} \quad (t \in D^{\omega})$$

Operador de Composição Paralela

1. Comportamento

As regras seguintes descrevem o comportamento do operador de composição paralela dependendo se a sincronização pode ou não ocorrer, e dependendo da ocorrência da violação temporal.

(a) presença de sincronização (ocorrência de $a \in L$)

i. ações clássicas

$$\frac{E \xrightarrow{a} E', \; F \xrightarrow{a} F'}{E|[L]|F \xrightarrow{a} E'|[L]|F'} \quad (a \in L \cup \{\delta\})$$

ii. violações temporais

A.

$$\frac{E \xrightarrow{a^*} E', \; F \xrightarrow{a^*} F'}{E|[L]|F \xrightarrow{a^*} E'|[L]|F'} \quad (a \in L)$$

B.

$$\frac{(E \xrightarrow{a^*} E') \wedge (F \not\xrightarrow{\delta} \vee F \not\xrightarrow{a^*})}{E|[L]|F \xrightarrow{a^*} E'|[L]|F \quad F|[L]|E \xrightarrow{a^*} F|[L]|E'} \quad (a \in L)$$

(b) ausência de sincronização (ocorrência de $a \notin L$)

i. ações clássicas

$$\frac{E \xrightarrow{a} E'}{E|[L]|F \xrightarrow{a} E'|[L]|F \quad F|[L]|E \xrightarrow{a} F|[L]|E'} \quad (a \notin L \cup \{\delta\})$$

ii. violações temporais

$$\frac{E \xrightarrow{a^*} E'}{E|[L]|F \xrightarrow{a^*} E'|[L]|F \quad F|[L]|E \xrightarrow{a^*} F|[L]|E'} \quad (a \notin L)$$

2. Progressão do tempo

$$\frac{(E \xrightarrow{t} E' \wedge F \xrightarrow{t} F'),\ (\forall a \in L\ E \not\xrightarrow{a} \wedge F \not\xrightarrow{a})}{E|[L]|F \xrightarrow{t} E'|[L]|F'} \quad (t \in D^\omega)$$

Operador de Ocultação

1. Comportamento

(a) ações clássicas

i.

$$\frac{E \xrightarrow{a} E'}{\text{hide } L \text{ in } E \xrightarrow{a} \text{hide } L \text{ in } E'} \quad (a \notin L)$$

ii.

$$\frac{E \xrightarrow{a} E'}{\text{hide } L \text{ in } E \xrightarrow{i} \text{hide } L \text{ in } E'} \quad (a \in L)$$

(b) violações temporais

i.

$$\frac{E \xrightarrow{a^*} E'}{\text{hide } L \text{ in } E \xrightarrow{a^*} \text{hide } L \text{ in } E'} \quad (a \notin L)$$

ii.

$$\frac{E \xrightarrow{a^*} E'}{\text{hide } L \text{ in } E \xrightarrow{i} \text{hide } L \text{ in } E'} \quad (a \in L)$$

2. Progressão do tempo

$$\frac{E \xrightarrow{t} E',\ (\forall a \in L\ E \not\xrightarrow{a})}{\text{hide } L \text{ in } E \xrightarrow{t} \text{hide } L \text{ in } E'} \quad (t \in D^\omega)$$

Comentário: Mesmo que a regra acima imponha urgência na ocorrência das ações que são oferecidas, existe um não-determinismo entre a ocorrência de um a ou de um a^* quando da especificação de uma expressão como $[t,t]a;E$. É para tratar situações como esta que as regras 1.(b).i e 1.(b).ii foram estabelecidas. Observe que a única possibilidade de ocorrência de um a^* acontece quando atribuímos um intervalo pontual, do tipo $[t,t]$, numa prefixação da ação a. Assim, as regras do item 1.(b) tratam exclusivamente este caso. Em todas as outras situações a urgência imposta pela regra no item 2 é mandatória, e as ações ocorrem tão logo elas sejam possíveis.

Operador de Composição Seqüencial

1. Comportamento

(a) ações clássicas

i.

$$\frac{E \xrightarrow{a} E'}{E \gg F \xrightarrow{a} E' \gg F} \quad (a \in Act^i)$$

ii.

$$\frac{E \xrightarrow{\delta} E'}{E \gg F \xrightarrow{i} F}$$

(b) violações temporais

$$\frac{E \xrightarrow{a^*} E'}{E \gg F \xrightarrow{a^*} E' \gg F} \quad (a \in Act)$$

2. Progressão do tempo

$$\frac{E \not\xrightarrow{\delta}, \; E \xrightarrow{t} E'}{E \gg F \xrightarrow{t} E' \gg F} \quad (t \in D^\omega)$$

Comentário: A regra de progressão do tempo acima estabelece o único caso de urgência da ação δ. Ou seja, se a ação δ pode ocorrer num processo E, então ela ocorrerá imediatamente no processo $E \gg F$. (O tempo só pode progredir em $E \gg F$, se E não pode realizar δ.)

Operador de Preempção

1. Comportamento

(a) ações clássicas

i.

$$\frac{E \xrightarrow{a} E'}{E \,[\!> F \xrightarrow{a} E' \,[\!> F} \quad (a \in Act^i)$$

ii.

$$\frac{F \xrightarrow{a} F'}{E \,[\!> F \xrightarrow{a} F'} \quad (a \in Act^{i,\delta})$$

iii.

$$\frac{E \xrightarrow{\delta} E'}{E \,[\!> F \xrightarrow{\delta} E'}$$

(b) violações temporais

i.

$$\frac{F \xrightarrow{a^*} F'}{E \,[\!> F \xrightarrow{a^*} E \,[\!> F'} \quad (a \in Act)$$

ii.

$$\frac{E \xrightarrow{a^*} E'}{E\,[>\,F \xrightarrow{a^*} E'\,[>\,F} \quad (a \in Act)$$

2. Progressão do tempo

$$\frac{E \xrightarrow{t} E',\quad F \xrightarrow{t} F'}{E\,[>\,F \xrightarrow{t} E'\,[>\,F'} \quad (t \in D^{\delta})$$

Operador de tratamento de violações temporais

1. Comportamento

(a) ações clássicas

i.

$$\frac{E \xrightarrow{a} E'}{E < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\} \xrightarrow{a} E' < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\}} \quad (a \in Act^i)$$

ii.

$$\frac{E \xrightarrow{\delta} E'}{E < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\} \xrightarrow{\delta} E'}$$

(b) violações temporais

i.

$$\frac{E \xrightarrow{a_j^*} E'}{E < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\} \xrightarrow{i} Q_j} \quad (a_j \in L)$$

Comentário: A ocorrência de uma violação temporal a_j^* no processo "E", com $a_j \in L$, faz com que "$E < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\}$" torne-se, silenciosamente, o processo "Q_j" que estava associado a ação a_j.

ii.

$$\frac{E \xrightarrow{b^*} E'}{E < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\} \xrightarrow{b^*} E' < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\}} \quad (b \in Act - L)$$

Comentário: Se uma ação $b \notin L$, então violações temporais b^* no processo "E" não provocam nenhuma transformação no processo "$E < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\}$".

2. Progressão do tempo

$$\frac{E \xrightarrow{t} E'}{E < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots, a_n{:}Q_n\} \xrightarrow{t} E' < L]\,\{a_1{:}Q_1, a_2{:}Q_2, \cdots a_n{:}Q_n\}} \quad (t \in D^{\omega})$$

Operador de instanciação de processos

1. Comportamento

(a) ações clássicas

$$\frac{E[a_1/a'_1,\cdots,a_n/a'_n] \xrightarrow{a} E' \ , \ \ P[a'_1,\cdots,a'_n] := E}{P[a_1,\cdots,a_n] \xrightarrow{a} E'} \quad (a \in Act^{i,\delta})$$

(b) violações temporais

$$\frac{E[a_1/a'_1,\cdots,a_n/a'_n] \xrightarrow{a^*} E' \ , \ \ P[a'_1,\cdots,a'_n] := E}{P[a_1,\cdots,a_n] \xrightarrow{a^*} E'} \quad (a \in Act)$$

2. Progressão do tempo

$$\frac{E[a_1/a'_1,\cdots,a_n/a'_n] \xrightarrow{t} E' \ , \ \ P[a'_1,\cdots,a'_n] := E}{P[a_1,\cdots,a_n] \xrightarrow{t} E'} \quad (t \in D^{\omega})$$

II.2.5 Propriedades de RT-LOTOS

Nesta seção são apresentadas algumas importantes propriedades satisfeitas pelo formalismo RT-LOTOS.

Muitas das propriedades apresentadas nesta seção são propriedades que dizem respeito às transições: $E \xrightarrow{\alpha} E'$. Desta maneira, para simplificar a trabalho das demonstrações (e quando for o caso), as provas de tais propriedades utilizarão a técnica de *indução transicional* [2] [Milner 89].

II.2.5.1 Consistência da Semântica Operacional Transicional

O primeiro requisito que deve ser satisfeito pelo formalismo proposto é que o conjunto de regras que definem operacionalmente a sua semântica seja *consistente* [3].

A existência de regras com *premissas negativas* (regras operacionais que incluem premissas do tipo $E \not\xrightarrow{a}$) é o principal responsável pela introdução de inconsistências em semânticas operacionais. Em [Groote 90] é estabelecido um método para se verificar em que condições pode-se ter uma semântica operacional consistente mesmo com regras com premissas negativas.

No caso da definição da semântica operacional de RT-LOTOS constata-se que as "premissas negativas" existentes são na realidade funções de predicados lógicos avaliados sobre

[2] A idéia da técnica de provas por indução transicional é a seguinte: toda transição é derivada da aplicação de um número finito de regras de inferência. Assim, a árvore de inferência é finita com sua raiz sendo a própria transição (exemplos de árvores de inferência são dados em [Milner 89], página 47). Então, pode-se provar uma propriedade de $E \xrightarrow{\alpha} E'$ por indução na estrutura da árvore de inferência para inferir $E \xrightarrow{\alpha} E'$.

[3] Um conjunto de regras operacionais é dito inconsistente, se através dele puder ser derivado que só é possível realizar uma dada ação se ela não puder ser realizada. Um conjunto de regras inconsistente não define uma semântica operacional transicional

a estrutura sintática das expressões de comportamentos (função "$\hookrightarrow$"), e segundo [Nicollin 92] estas estão *bem definidas*. E assim, de acordo com Groote [Groote 90], a relação de transição $\rightarrow$ está *de acordo com* o sistema de transições definido pelas regras semânticas da seção anterior, ou seja a relação de transição também está bem definida.

Por outro lado, considera-se ainda as funções de predicado sintático que definem situações de impossibilidade de realização de ações (como por exemplo: $E \stackrel{a}{\not\rightarrow}$) como premissas negativas. Pode-se provar que essas premissas negativas existentes em algumas regras operacionais da semântica de RT-LOTOS não geram inconsistências, e assim, que a semântica operacional transicional definida para RT-LOTOS é consistente. A prova deste fato baseia-se na verificação da satisfação das hipóteses do teorema 2.4.2 de [Groote 90] que estabelece uma condição suficiente para que uma definição semântica com premissas negativa seja consistente. Desta maneira, pode-se estabelecer para RT-LOTOS a seguinte proposição:

Proposição 1 A semântica operacional de especificações em RT-LOTOS é consistente.

Prova: A especificação do sistema de transições dado na seção anterior, o qual pretende-se definir a semântica operacional de RT-LOTOS, é *estratificável*[4] de acordo com a definição 2.3.1 de [Groote 90]. Para certificar-se deste fato, pode-se provar (analogamente ao que é que feito em [Leduc 94]) que a função S é uma estratificação: $S(E \stackrel{a}{\rightarrow} E) := rk(a)$, onde $rk(a) = 0$ se $a \in Act$ e $rk(a) = 1$ se $a \in D$. Assim, aplicando-se o teorema 2.4.2 de [Groote 90], tem-se que a semântica é consistente. □

II.2.5.2 Determinismo Temporal

Quando um processo E não realiza nenhuma ação durante um tempo d, e o comportamento resultante é completamente determinado por E e d, então diz-se que a progressão do tempo é determinista, ou que tem-se a propriedade de determinismo temporal. Esta propriedade pode ser caracterizada formalmente como:

Para quaisquer expressões de comportamento E, E', E'' e $\forall d \in D$, se $E \stackrel{d}{\rightarrow} E' \wedge E \stackrel{d}{\rightarrow} E''$, então $E' = E''$, onde "=" é a igualdade sintática.

Proposição 2 As transições temporais da semântica operacional de RT-LOTOS são deterministas.

Intuitivamente esta propriedade pode ser verificada pela simples inspeção das regras de progressão do tempo na semântica operacional de RT-LOTOS. Observe-se que a semântica operacional não possui nenhuma regra que a partir de uma mesma transição temporal como premissa permita duas ou mais transições diferentes como conseqüência. Formalmente, a prova desta proposição é dada a seguir.

Prova: Assuma que $E \stackrel{d}{\rightarrow} E'$, $E \stackrel{d}{\rightarrow} E''$, e $d \in D$. Deseja-se mostrar que $E' \equiv E''$ por indução transicional sobre $E \stackrel{d}{\rightarrow} E'$ e $E \stackrel{d}{\rightarrow} E''$.

[4] A definição deste termo é feita através do estabelecimento de uma forma normal que uma definição semântica deve satisfazer. Por esta ser demasiadamente longa, e um pouco fora dos objetivos centrais deste trabalho, ela será omitida e unicamente referenciada [Groote 90].

1. *stop*: $E \equiv stop$. Trivial.
2. *exit*: $E \equiv exit$. A única regra de transição temporal definida para *exit* leva o processo *exit* a ele mesmo: $\forall d \in D$, $exit \xrightarrow{d} exit$. O que é o resultado desejado.
3. Prefixação: $E \equiv [t_1, t_2]a; F$, com $a \in Act^i$. Suponha-se que $d \leq t_2$. Se $t_2 = 0$ nada tem-se a mostrar pois não existe nenhuma transição temporal possível. No outro caso, tem-se que a única evolução temporal possível é $[t_1, t_2]a; F \xrightarrow{d} [t_1 - d, t_2]a; F$.
4. Escolha: $E \equiv F[]G$. Neste caso, E' e E'' devem ser da forma $F'[]G'$ e $F''[]G''$. Por inferência simples tem-se que $F \xrightarrow{d} F'$ e $F \xrightarrow{d} F''$. Por indução, obtem-se $F' = F''$ e similarmente, $G' = G''$. Assim, $F'[]G' = F''[]G''$, como desejado.
5. Composição paralela: $E \equiv F|[L]|G$. Similar ao caso da escolha.
6. Os casos da Ocultação, Composição seqüencial, Preempção, Preempção temporal, e Instanciação de processos são provados também diretamente por indução transicional. □

II.2.5.3 Aditividade Temporal

Em um formalismo temporizado é mister assegurar a solidez ("*soundness*") da noção de tempo em que o formalismo é fundado. Para este fim, é definida a *aditividade temporal* [Nicollin 92] (ou *continuidade temporal* em [Yi 91]). Esta propriedade garante que

- um processo que pode ser retardado por $d + d'$ unidades de tempo, pode ser retardado por d e depois por d' unidades de tempo, e vice-versa
- em ambos os casos o resultado deve ser o mesmo

Formalmente, a aditividade temporal pode ser estabelecida como

$$\forall E, E' \in \mathcal{E}, \forall d, d' \in D : (\exists E'' : E \xrightarrow{d} E'' \wedge E'' \xrightarrow{d'} E') \Leftrightarrow E \xrightarrow{d+d'} E'$$

Proposição 3 RT-LOTOS satisfaz a propriedade de aditividade temporal.

A satisfação desta propriedade decorre principalmente da definição de domínio temporal e, desde que sua demonstração não apresenta nenhuma dificuldade, podendo ser realizada através de indução transicional, ela será omitida.

II.2.5.4 Persistência

A propriedade de persistência em uma álgebra de processos temporizada assegura que a realização de transições temporais (progressão do tempo) não suprime a possibilidade de realização de uma ação.

Proposição 4 RT-LOTOS satisfaz a propriedade de persistência, isto é: $\forall E, E' \in \mathcal{E}$, $t \in D$, $a \in Act^{i,\delta} \cup Act^*$ se $E \xrightarrow{a}$ e $E \xrightarrow{t} E'$, então $E' \xrightarrow{a}$

Prova: A prova desta propriedade pode ser facilmente realizada por indução transicional, e aqui é apresentado apenas um esboço de sua realização. Por hipótese, tem-se que $E \xrightarrow{a}$ e que uma transição temporal é possível de ser realizada em E. Para todos os casos de E deve-se verificar todas as transições temporais $E \xrightarrow{t} E'$ e certificar-se que $E' \xrightarrow{a}$. Por exemplo, para o operador de prefixação, E deve ser da forma $[t_1, t_2]a; F$, com $t_2 > 0$ (para satisfazer a hipótese de que $E \xrightarrow{t} E'$); assim, tem-se duas possibilidades de realização de transições temporais:

(a) caso $t_1 = 0$. Aplica-se o axioma 2.(a) do operador de prefixação para se obter $[0, t_2]a; F \xrightarrow{t} [0, t_2 - t]a; F$

(b) caso $t_1 \neq 0$. Aplica-se o axioma 2.(b) do operador de prefixação para se obter $[t_1, t_2]a; F \xrightarrow{t} [t_1 - t, t_2]a; F$

E verifica-se em ambos os casos que as expressões de comportamento resultantes $E' = [0, t_2 - t]a; F$ e $E' = [t_1 - t, t_2]a; F$ oferecem ainda a ação a, isto é $E' \xrightarrow{a}$.

Para os outros operadores o tratamento é análogo e simples. □

II.2.5.5 Urgência

Em álgebras de processos temporizadas em geral o conceito de urgência é associado diretamente à realização de ações que não podem ser retardadas e devendo assim, serem realizadas imediatamente e de forma *autônoma*. Formalmente, uma ação urgente (autônoma) satisfaz a seguinte propriedade:

$$\exists E, a, E' : \; E \xrightarrow{a} E' \; \wedge \; \forall d : \; E \overset{d}{\not\rightarrow}$$

O conceito de ação urgente não está em conformidade com o paradigma das álgebras de processos. Por este paradigma uma ação observável oferecida por um processo realiza-se quando o seu ambiente também está pronto para isto; ou seja, a realização de uma ação depende também do ambiente em que ela está inserida. Entretanto, a autonomia das ações urgentes viola este paradigma forçando a sua realização em um instante determinado. A única hipótese de urgência aceita no paradigma de álgebras de processos é a semântica de progresso máximo atribuída às ações internas de uma dada álgebra de processos (a urgência da ação τ está fortemente relacionada ao mecanismo de comunicação de CCS).

Porém, a hipótese de urgência nas ações dota os formalismos de uma expressividade muito elevada; por exemplo, Bolognesi prova que seu modelo proposto em [Bolognesi 92] e que define urgência em todas as ações possui poder de expressão equivalente ao das Máquinas de Turing. Muitas são as álgebras de processos temporizadas que adotam ações imediatas (urgentes ou autônomas), como por exemplo os modelos apresentados em [Moller 89, Nicollin 92, Bolognesi 92, etc]. Outras álgebras de processos não permitem nenhuma urgência nas ações observáveis como por exemplo os modelos em [Hennessy 90 e Leduc 93]. Na álgebra de processos RT-LOTOS optou-se por uma posição intermediária entre a possibilidade e impossibilidade de se especificar ações urgentes. As características de urgência de RT-LOTOS são descritas a seguir.

Em RT-LOTOS, exceto o caso da ação i especificada diretamente (isto é, sem o uso do operador *hide*), não é possível especificar outras ações que devam ocorrer em instantes

determinados. Porém, existem alguns mecanismos que podem fazer com que a realização de ações ordinárias tornem-se urgentes, ou provoquem a realização urgente de uma ação. Por exemplo, a expiração de um intervalo temporal atribuído a uma dada ação a faz com que a ação a^* (notificando a violação temporal) ocorra imediatamente.

Outros exemplos de indução de urgência são encontrados nos operadores de ocultação (*hide L in E*), composição seqüencial ($E >> F$), e nas sincronizações de ações do operador de composição paralela ($E|[L]|F$). Uma ação ocultada pelo operador *hide* torna-se urgente na medida em que ela se realiza desde que seja possível fazê-la; isto a torna compatível com a observação externa do processo que contém o *hide*, pois lá o que é observado é a realização de uma ação i que é urgente por natureza. No caso do operador de composição seqüencial ($E >> F$) a urgência é imposta à realização da ação δ; isto implica que a passagem de mão de um processo *exit* para um processo que esteja composto seqüencialmente com ele tem lugar tão logo o processo *exit* seja possível. Isto evita a geração de um retardo não determinista não desejado em tal situação. No caso do operador de composição paralela ($E|[L]|F$), definiu-se que as ações sincronizáveis ocorrem tão logo sejam oferecidas. A introdução deste tipo de sincronização urgente foi devida a uma decisão de projeto que visou atingir dois principais objetivos: dotar o mecanismo de sincronização do princípio de comunicação urgente de CCS; e simplificar o problema complexo que é a sincronização múltipla de ações em que cada uma das participantes pode estar sujeita a uma restrição temporal diferente. Este último problema não é ainda bem resolvido na literatura, e as melhores soluções propostas para ele [Bolognesi 92, Leduc 93] são mais restritivas que a encontrada neste trabalho. Em [Bolognesi 92] as ações a sincronizar não são temporizadas e ocorrem tão logo sejam possíveis, isto é são urgentes desde que possam ocorrer. Já em [Leduc 93] a solução proposta é a de ocultar por um operador *hide* as operações de sincronização. Ambas as soluções anteriores parecem um pouco restritivas quando comparadas àquela adotada neste trabalho. Em RT-LOTOS, a sincronização múltipla é urgente, isto é as ações a sincronizar podem sofrer restrições temporais e a sincronização ocorre no primeiro instante em que é possível a realização de todas as ações que dela participam. No caso da ocorrência de uma violação temporal em uma ação a sincronizar, então esta violação temporal provocará a realização de uma violação temporal no processo composto, o que pode ser visto como uma falha de sincronização.

Em resumo, existem quatro casos onde é imposta urgência para a realização das ações em RT-LOTOS: em ações ocultadas pelo operador de ocultação; no operador de terminação com sucesso; nas operações de composição paralela com sincronização de ações; e quando expira o intervalo de tempo associado a uma ação, caso em que uma ação de violação temporal ocorre.

Comportamentos urgentes: Como já visto, não se pode especificar ações urgentes em RT-LOTOS. Porém, a urgência das realizações das violações temporais juntamente com o operador de tratamento de violações temporais permite a obtenção de comportamentos urgentes, ou seja, o processo que realizou a violação temporal pode ser imediatamente abandonado e um outro processo pode ser instantaneamente lançado no seu lugar. Comportamentos urgentes são uma forma menos rígida de urgência, mas em muitas situações práticas pode ser tão expressivo quanto as ações urgentes.

Finalmente, note-se que que as regras de progressão do tempo de todas os operadores indutores de urgência discutidos aqui possuem premissa que as fazem verificar a propriedade de urgência estabelecida no início desta seção.

II.2.5.6 Variabilidade Finita (Non-Zenoness)

Um processo tem a propriedade de *variabilidade finita* [Nicollin 92] (também chamada de propriedade *non-Zenoness* em [Hansson 91], em alusão à paradoxos de Zeno de Elea, notadamente o de Aquiles e a tartaruga) se ele pode realizar apenas um número finito de ações em um intervalo de tempo finito. Ao contrário, processos que podem realizar um número infinito de ações em um tempo finito são chamados de *processos Zeno*. Existem essencialmente duas maneiras de se obter esta propriedade em um processo: impondo um retardo antes ou após a realização de cada ação, ou atribuíndo-se uma duração não nula a cada uma das ações. Porém, nesta última possibilidade esbarra-se com dificuldades de ordem teórica e destrói-se o caráter abstrato do tempo. Por outro lado, Hansson [Hansson 91], e Nicollin e Sifakis [Nicollin 92] argumentam que além de processos Zeno não serem intuitivos, eles também não são *implementáveis* [5].

Nem todos os processos descritos em RT-LOTOS possuem a propriedade de variabilidade finita (esta afirmação também é válida para todas as álgebras de processos relacionadas neste trabalho). Ou seja, em RT-LOTOS pode-se utilizar os intervalos das ações para se especificar tanto processos Zeno (não implementáveis), quanto processos non-Zeno (implementáveis).

II.2.5.7 RT-LOTOS como extensão estrita de LOTOS Básico

Procurou-se definir o formalismo RT-LOTOS como uma extensão estrita de LOTOS Básico [ISO 88], e é fácil ver que RT-LOTOS mantém todas as funcionalidades do seu modelo de base. Contudo, é necessário que sejam estabelecidas as bases formais do que significa uma extensão estrita de um formalismo e verificar que RT-LOTOS as satisfazem com respeito a LOTOS Básico.

Considere-se a álgebra de processos LOTOS Básico $= (OP, A, R_A^{OP}, \sim)$, onde OP é o conjunto de operadores, A é o alfabeto de ações, R_A^{OP} é o conjunto de regras da semântica operacional, e $\sim$ é a equivalência observacional forte. Considere-se a álgebra de processos RT-LOTOS $= (OP', A', R_{A'}^{OP'}, \sim_t)$ onde OP' é o conjunto de operadores onde $OP' \subset OP$, $A' = A \cup D^\omega$ é o alfabeto de ações, $R_{A'}^{OP'}$ é o novo conjunto de regras da semântica operacional, e $\sim_t$ é a equivalência observacional forte definida para RT-LOTOS ($\sim_t$ denota a relação de equivalência a ser definida na seção II.2.6.1).

A definição a seguir estabelece o relacionamento teórico que duas álgebras devem satisfazer para que uma seja considerada uma extensão estrita da outra.

Considera-se que uma dada álgebra de processos é uma extensão estrita (ou extensão conservativa) de uma outra, se os dois requisitos abaixo forem satisfeitos para o par de álgebras:

- Conservação da semântica: $\forall r \in R_A^{OP}$, r é válida em $R_{A'}^{OP'}$ se for aplicada sobre termos de LOTOS Básico. As regras R_A^{OP} continuam válidas em RT-LOTOS desde que sejam aplicadas a termos de LOTOS Básico.

[5] Segundo Nicollin e Sifakis [Nicollin 92] um processo é implementável se ele pode ser executado em um processador onde a medida do tempo é provida por um relógio discreto, isto é, para qualquer duração d existe um limite n para o número de ações realizadas. Este requisito é chamado de *variabilidade limitada*. Variabilidade limitada implica em non-Zenoness.

- Isomorfismo: $\forall E, F \in LOTOS$, $E \sim F \Leftrightarrow E \sim_t F$. A teoria de processos em LOTOS Básico é isomorfa à restrição de RT-LOTOS aos construtores de LOTOS Básico.

Esta definição é devida a Nicollin e Sifakis [Nicollin 92] e utilizada também em [Hansson 91] e [Leduc 94].

A conservação semântica de RT-LOTOS com respeito a LOTOS Básico pode ser verificada diretamente das regras da semântica operacional; pois as regras foram estabelecidas objetivando a satisfação desse requisito.

Analogamente ao que é demonstrado em [Leduc 94] o isomorfismo das teorias de (RT_LOTOS, $\sim_t$) e (LOTOS, $\sim$) é verdadeiro somente para especificações guardadas, ou seja, especificações que não levem a comportamentos divergentes. A verificação deste resultado segue diretamente dos dois seguintes lemas:

Lema 1 $\forall E, F \in$ LOTOS e $\forall a \in A$, então $E \xrightarrow{a} F$ de acordo com R_A^{OP} se, e somente se $E \xrightarrow{a} F$ de acordo com $R_{A'}^{OP'}$

A prova do lema 1 segue diretamente da verificação de que a aplicação dos axiomas e regras de inferência de R_A^{OP} e de $R_{A'}^{OP'}$ produzem exatamente o mesmo resultado.

Lema 2 De acordo com $R_{A'}^{OP'}$, $\forall E \in$ LOTOS

- se $E \xrightarrow{i}$, então $\forall d \in D$, $E \not\xrightarrow{d}$
- se $E \not\xrightarrow{i}$, então $\forall d \in D$, $E \xrightarrow{d} E$ para expressões guardadas.

A prova do lema 2 pode ser realizada por indução transicional. Na primeira parte o caso base é $E \equiv i; F$, que, em RT-LOTOS equivale a $E \equiv [0,0]i; F$. Para a segunda parte os casos bases são $E \equiv stop$ e $E \equiv a; F$ com $a \neq i$ (lembrando que em RT-LOTOS $a; F$ equivale a $[0,\omega]a; F$, e que por definição de ω tem-se que $\forall d \in D$, $\omega - d = \omega$, implicando assim, que $[0,\omega]a; F \xrightarrow{d} [0,\omega - d]a; F \equiv [0,\omega]a; F$).

Considerando-se o exposto acima tem-se que RT-LOTOS é uma extensão estrita de LOTOS Básico.

II.2.6 Bissimulações em RT-LOTOS

Nesta seção são definidas bissimulações temporais dentro do contexto de RT-LOTOS. Tendo em vista a hipótese feita sobre o domínio do tempo (ou seja, que o domínio do tempo é enumerável), o modelo subjacente é o usual Sistema de Transições Rotuladas (STR). Numa janela de observação do tempo, no caso de tempo denso, o STR deverá ser infinito (número infinito de estados e ramificações); mas isto não caracteriza um problema para o enfoque deste trabalho, pois objetiva-se apenas qualificar bissimulações temporais e não propor algoritmos para estas.

II.2.6.1 Bissimulação Temporal Forte

A definição da bissimulação temporal forte corresponde à definição da bissimulação forte clássica, onde a ação interna i, as violações temporais, bem como os arcos temporizados de D^{ω} são consideradas como ações quaisquer.

Definição 6 Seja $\mathcal{L} = Act^{i,\delta} \cup Act^{*} \cup D^{\omega}$ um conjunto de ações. Então, define-se a função $\mathcal{F}$, sobre subconjuntos de $\mathcal{E} \times \mathcal{E}$ (isto é, relações binárias sobre expressões de comportamento), como segue. Se $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$, então $(P,Q) \in \mathcal{F}(\mathcal{R})$ se, e somente se, para toda ação $a \in \mathcal{L}$:

1. sempre que $P \xrightarrow{a} P'$ então, para algum Q', $Q \xrightarrow{a} Q'$ e $(P',Q') \in \mathcal{R}$, e
2. sempre que $Q \xrightarrow{a} Q'$ então, para algum P', $P \xrightarrow{a} P'$ e $(P',Q') \in \mathcal{R}$

$\mathcal{R}$ é chamada uma bissimulação temporal forte se, e somente se, $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$. Se $(P,Q) \in \mathcal{R}$ para alguma bissimulação temporal $\mathcal{R}$, então P e Q são ditos fortemente bissimilares temporais, em símbolos $P \sim_t Q$. Como usual, isto pode ser expresso como: $\sim_t = \bigcup\{\mathcal{R} : \mathcal{R}$ é uma bissimulação temporal forte$\}$. □

Proposição 5 Seja $\mathcal{L} = Act^{i,\delta} \cup Act^{*} \cup D^{\omega}$ e seja $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ uma bissimulação temporal forte. Então $(P,Q) \in \mathcal{R}$ implica que:

1. sempre que $P \xrightarrow{a} P'$ então, para algum Q', $Q \xrightarrow{a} Q'$ e $(P',Q') \in \mathcal{R}$, e
2. sempre que $Q \xrightarrow{a} Q'$ então, para algum P', $P \xrightarrow{a} P'$ e $(P',Q') \in \mathcal{R}$

Prova: Por hipótese, $\mathcal{R}$ é uma bissimulação temporal forte, então $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$ o que implica que $(P,Q) \in \mathcal{F}(\mathcal{R})$, logo, obtem-se os itens 1 e 2 da proposição 5. □

Proposição 6 A Relação $\sim_t$ tem as seguintes propriedades:

1. $\sim_t$ é a maior bissimulação temporal forte
2. $\sim_t$ é uma relação de equivalência

Prova: A demonstração da proposição acima é completamente similar à demonstração de propriedades equivalentes realizadas em [Milner 89, pp. 90-91]. □

Proposição 7 A equivalência temporal forte tem a propriedade de poder ser substituida para todos os operadores RT-LOTOS. Em outras palavras, sejam P_1, P_2, Q, e $Q_1, \cdots, Q_n$ expressões de comportamento e $P_1 \sim_t P_2$. Então

(1) $[t_1,t_2]a;P_1 \sim_t [t_1,t_2]a;P_2$

(2) $P_1[]Q \sim_t P_2[]Q$

(3) $P_1|[L]|Q \sim_t P_2|[L]|Q$

(4) $hideLinP_1 \sim_t hideLinP_2$

(5) $P_1 >> Q \sim_t P_2 >> Q$

(6) $P_1 \rhd Q \sim_t P_2 \rhd Q$

(7) $P_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\} \sim_t P_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}$

Prova: A prova de (7) é dada na seqüência, os outros ítens podem ser provadas de maneira semelhante.

Seja $\mathcal{S} = \{(P_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}, P_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}) : P_1 \sim_t P_2\}$. Serão discutidos os seguintes casos:

1. $P_1 \xrightarrow{a} P'_1$ tal que $a \in D^\omega \cup Act^{i,\delta}$:
   como $P_1 \sim_t P_2$, então existe uma derivação $P_2 \xrightarrow{a} P'_2$ tal que $P'_1 \sim_t P'_2$. Também, as regras de transição implicam que
   $P_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\} \xrightarrow{a} P'_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}$ e
   $P_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\} \xrightarrow{a} P'_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}$, e assim
   $(P'_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}, P'_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}) \in \mathcal{S}$.

2. $P_1 \xrightarrow{a^*} P'_1$, como $P_1 \sim_t P_2$, então existe uma derivação $P_2 \xrightarrow{a^*} P'_2$ tal que $P'_1 \sim_t P'_2$. Existem dois casos:

   (a) $a \notin L$, e então tem-se as derivações:
   $P_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\} \xrightarrow{a^*} P'_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}$ e
   $P_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\} \xrightarrow{a^*} P'_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}$, e tem-se
   $(P'_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}, P'_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\}) \in \mathcal{S}$

   (b) $a \in L$, com $a^* = a_k^*$ para algum $k \in \{1, \cdots, n\}$, e então tem-se as derivações:
   $P_1 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\} \xrightarrow{i} Q_k$ e $P_2 < L]\{a_1{:}Q_1, \cdots, a_n{:}Q_n\} \xrightarrow{i} Q_k$ que satisfaz o requisito.

Por um argumento simétrico, completa-se a prova de que $\mathcal{S}$ é uma bissimulação temporal forte. □

Em resumo, a bissimulação temporal forte ($\sim_t$) leva em conta a ocorrência de todos os tipos de ações: ações observáveis ($a \in Act$), ação interna i, ações de violação temporal ($a^* \in Act^*$) e ações devidas à progressão do tempo ($d \in D$). Como exemplo, considere-se as expressões de comportamento E e F a seguir onde a diferença entre elas é o intervalo temporal atribuído à ação inicial a.

$E = [1, 3]a; (b; stop[\,]c; stop)$

$F = [1, 4]a; (b; stop[\,]c; stop)$

Então $E \not\sim_t F$, pois F pode realizar ações devidas à progressão do tempo que E não está apta a fazer. Observe-se que é plenamente justificável o atributo "forte" para a bissimulação $\sim_t$.

II.2.6.2 Bissimulação Temporal Fraca

A definição de equivalência temporal fraca corresponde à definição de bissimulação fraca clássica, onde as violações temporais, bem como os arcos temporizados de D^ω são considerados como uma ação qualquer.

Definição 7 Define-se a relação de transição $\hookrightarrow \subset \mathcal{E} \times \mathcal{E}$ como:

1. $E \stackrel{a}{\hookrightarrow} F$ se, e somente se, $E(\stackrel{i}{\rightarrow})^* \stackrel{a}{\rightarrow} (\stackrel{i}{\rightarrow})^* F$, onde $(\stackrel{i}{\rightarrow})^*$ representa zero ou mais ocorrências da transição $(\stackrel{i}{\rightarrow})$, e $a \in Act^\delta \cup Act^* \cup D^\omega$.

2. $E \stackrel{d}{\hookrightarrow} F$ se, e somente se, $E(\stackrel{i}{\rightarrow})^* \stackrel{d_1}{\rightarrow} (\stackrel{i}{\rightarrow})^* \cdots (\stackrel{i}{\rightarrow})^* \stackrel{d_n}{\rightarrow} (\stackrel{i}{\rightarrow})^* F$, onde $d_j \in D^\omega$ para $j \leq n$ e $d = \sum_{j \leq n} d_j$. □

Note que, em particular, $E \stackrel{\epsilon}{\hookrightarrow} E'$ se, e somente se, $E(\stackrel{i}{\rightarrow})^* E'$.

Seja $\mathcal{L}^+ = Act^\delta \cup Act^* \cup D^\omega \cup \{\epsilon\}$ e considere a relação de transição $\hookrightarrow$, então obtém-se o sistema de transições rotuladas

$$\langle \mathcal{E}, \mathcal{L}^+, \{\stackrel{a}{\hookrightarrow} : a \in \mathcal{L}^+\} \rangle$$

sobre expressões de comportamento. Desta forma, pode-se estabelecer a noção de equivalência desejada neste sistema.

Definição 8 Seja $\mathcal{L}^+ = Act^\delta \cup Act^* \cup D^\omega \cup \{\epsilon\}$ o conjunto de ações. Então, define-se a função $\mathcal{F}^+$, sobre subconjuntos de $\mathcal{E} \times \mathcal{E}$ (isto é, relações binárias sobre expressões de comportamento), como segue. Se $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$, então $(P, Q) \in \mathcal{F}^+(\mathcal{R})$ se, e somente se, para toda ação $a \in \mathcal{L}^+$:

1. sempre que $P \stackrel{a}{\rightarrow} P'$ então, para algum Q', $Q \stackrel{a}{\hookrightarrow} Q'$ e $(P', Q') \in \mathcal{R}$, e

2. sempre que $Q \stackrel{a}{\rightarrow} Q'$ então, para algum P', $P \stackrel{a}{\hookrightarrow} P'$ e $(P', Q') \in \mathcal{R}$

$\mathcal{R}$ é chamada uma bissimulação temporal fraca se, e somente se, $\mathcal{R} \subseteq \mathcal{F}^+(\mathcal{R})$. Se $(P, Q) \in \mathcal{R}$ para alguma bissimulação temporal fraca $\mathcal{R}$, então P e Q são ditos fracamente bissimilares temporais, o que se escreve simbolicamente $P \approx_t Q$. Como usual, isto pode ser expresso como: $\approx_t = \bigcup\{\mathcal{R} : \mathcal{R} \text{ é uma bissimulação temporal fraca}\}$. □

Proposição 8 Seja $\mathcal{L}^+ = Act^\delta \cup Act^* \cup D^\omega \cup \{\epsilon\}$ e seja $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ uma bissimulação temporal fraca. Então $\langle P, Q \rangle \in \mathcal{R}$ implica que:

1. sempre que $P \stackrel{a}{\rightarrow} P'$ então, para algum Q', $Q \stackrel{a}{\hookrightarrow} Q'$ e $(P', Q') \in \mathcal{R}$, e

2. sempre que $Q \stackrel{a}{\rightarrow} Q'$ então, para algum P', $P \stackrel{a}{\hookrightarrow} P'$ e $(P', Q') \in \mathcal{R}$

Prova: Por hipótese, $\mathcal{R}$ é uma bissimulação temporal fraca, então $\mathcal{R} \subseteq \mathcal{F}^+(\mathcal{R})$ o que implica que $(P, Q) \in \mathcal{F}^+(\mathcal{R})$, logo, tem-se (1) e (2). □

Proposição 9 A relação $\approx_t$ tem as seguintes propriedades:

1. $\approx_t$ é a maior bissimulação temporal fraca

2. $\approx_t$ é uma relação de equivalência

Prova: Similar à de proposição equivalente apresentada em [Milner 89]. □

Resumindo, a bissimulação temporal fraca ($\approx_t$) absorve apenas as ocorrências da ação interna i, e distingue todas as outras ações: ações observáveis ($a \in Act$), ações de violação temporal ($a^* \in Act^*$) e ações devidas à progressão do tempo ($d \in D$).

Como exemplo, considere-se as expressões de comportamento a seguir: $E = a;H$ e $F = i;a;H$. Então, tem-se que $E \approx_t F$. Porém, se $G = [t]i;a;H$, então $F \not\approx_t G$ (G pode realizar transições temporais que F não está apta para realizar). Em geral, se $E' = [t_1,t_2]i;H$ e $E'' = [t_3,t_4]i;H$ com $t_1 \neq t_3$ ou $t_2 \neq t_4$, então $E' \not\approx_t E''$.

II.2.6.3 Bissimulação Temporal Direta

A definição da bissimulação temporal direta corresponde à definição de bissimulação fraca clássica, onde os arcos temporizados de D^ω são considerados como ações quaisquer, e violações temporais são consideradas como ações internas. Utiliza-se o termo bissimulação temporal direta pois esta bissimulação se abstrai das violações temporais, e conseqüentemente não existem mais notificações de que não ocorrem dentro de seus respectivos intervalos de tempo. Com esta noção de equivalência pode-se observar a progressão do tempo e a realização de ações normais que ocorrem satisfazendo suas restrições de tempo. As ações incontroláveis $a^* \in Act^*$ e i não são observáveis. Esta é a razão de chamar $\overset{*}{\approx}$ de uma equivalência temporal direta.

Definição 9 Define-se a relação de transição $\rightsquigarrow \subset \mathcal{E} \times \mathcal{E}$ como:

1. $E \overset{a}{\rightsquigarrow} F$ se, e somente se $E(\overset{\rho}{\rightarrow})^* \overset{a}{\rightarrow} (\overset{\rho}{\rightarrow})^*F$, onde $(\overset{\rho}{\rightarrow})^*$ representa zero ou mais ocorrências da transição $(\overset{\rho}{\rightarrow})$, $a \in Act$ e $\rho \in Act^* \cup \{i\}$.

2. $E \overset{d}{\rightsquigarrow} F$ se, e somente se $E(\overset{\rho}{\rightarrow})^* \overset{d_1}{\rightarrow} (\overset{\rho}{\rightarrow})^*F \cdots (\overset{\rho}{\rightarrow})^* \overset{d_n}{\rightarrow} (\overset{\rho}{\rightarrow})^*$, onde $d_j \in D^\omega$ para $j \leq n$, $d = \sum_{j \leq n} d_j$ e $\rho \in Act^* \cup \{i\}$. □

Seja $\mathcal{L}^\circ = Act^\delta \cup D^\omega \cup \{\epsilon\}$ e considere $\rightsquigarrow$, então obtém-se um sistema de transições rotuladas padrão:

$$\langle \mathcal{E}, \mathcal{L}^\circ, \{\overset{a}{\rightsquigarrow} : a \in \mathcal{L}^\circ\} \rangle.$$

Sob este sistema pode-se estabelecer a noção de equivalência temporal direta.

Definição 10 Seja $\mathcal{L}^\circ = Act^\delta \cup D^\omega \cup \{\epsilon\}$ o conjunto de ações. Então, define-se a função $\mathcal{F}^\circ$, sobre subconjuntos de $\mathcal{E} \times \mathcal{E}$ (isto é, relações binárias sobre expressões de comportamento), como segue. Se $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$, então $(P,Q) \in \mathcal{F}^\circ(\mathcal{R})$ se, e somente se, para toda ação $a \in \mathcal{L}^\circ$:

1. sempre que $P \overset{a}{\rightarrow} P'$ então, para algum Q', $Q \overset{a}{\rightsquigarrow} Q'$ e $(P',Q') \in \mathcal{R}$, e

2. sempre que $Q \overset{a}{\rightarrow} Q'$ então, para algum P', $P \overset{a}{\rightsquigarrow} P'$ e $(P',Q') \in \mathcal{R}$

$\mathcal{R}$ é chamada uma bissimulação temporal direta se, e somente se, $\mathcal{R} \subseteq \mathcal{F}^\circ(\mathcal{R})$. Se $(P,Q) \in \mathcal{R}$ para alguma bissimulação temporal direta $\mathcal{R}$, então P e Q são ditos diretamente bissimilares temporais, em símbolos $P \overset{*}{\approx}_t Q$. Como usual, isto pode ser expresso como: $\overset{*}{\approx}_t = \bigcup\{\mathcal{R} : \mathcal{R} \text{ é uma bissimulação temporal direta}\}$. □

Proposição 10 Seja $\mathcal{L}^\circ = Act^\delta \cup D^\omega \cup \{\epsilon\}$ e seja $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ uma bissimulação temporal direta. Então $\langle P, Q\rangle \in \mathcal{R}$ implica que:

1. sempre que $P \xrightarrow{a} P'$ então, para algum Q', $Q \overset{a}{\rightsquigarrow} Q'$ e $(P', Q') \in \mathcal{R}$, e
2. sempre que $Q \xrightarrow{a} Q'$ então, para algum P', $P \overset{a}{\rightsquigarrow} P'$ e $(P', Q') \in \mathcal{R}$

Prova: Por hipótese, $\mathcal{R}$ é uma bissimulação temporal direta, então $\mathcal{R} \subseteq \mathcal{F}^\circ(\mathcal{R})$ o que implica que $(P, Q) \in \mathcal{F}^\circ(\mathcal{R})$, logo, tem-se (1) e (2). □

A proposição 11 estabelece algumas boas propriedades da bissimulação temporal direta.

Proposição 11 Dadas as relações binárias R, R_1 and R_2, então sejam:

$R^{-1} = \{(Q, P) : (P, Q) \in R\}$

$R_1R_2 = \{(P, S) :$ para algum $Q, (P, Q) \in R_1$ e $(Q, S) \in R_2\}$.

Assuma que cada R_j $(j = 1, 2, \cdots)$ é uma bissimulação temporal direta. Então todas as seguintes relações também são bissimulações temporais diretas.

(1) $Id_\mathcal{E}$ (3) R_1R_2

(2) R_j^{-1} (4) $\bigcup_j R_j$

Prova: (1), (2) e (4) são óbvios; a prova de (3) segue:

Seja $(P, S) \in R_1R_2$, então para algum Q tem-se $(P, Q) \in R_1$ e $(Q, S) \in R_2$.

Agora seja $P \overset{a}{\rightsquigarrow} P'$. Então para algum Q' tem-se que $Q \overset{a}{\rightsquigarrow} Q'$ e $(P', Q') \in R_1$, pois $(P, Q) \in R_1$.

Também como $(Q, S) \in R_2$ tem-se, para algum S', que $S \overset{a}{\rightsquigarrow} S'$.

Assim, $(P', S') \in R_1R_2$. Similarmente, se $S \overset{a}{\rightsquigarrow} S'$, pode-se encontrar P' tal que $P \overset{a}{\rightsquigarrow} P'$ e $(P', S') \in R_1R_2$. □

Proposição 12 A relação $\overset{*}{\approx}_t$ tem as seguintes propriedades:

1. $\overset{*}{\approx}_t$ é a maior bissimulação temporal direta
2. $\overset{*}{\approx}_t$ é uma relação de equivalência

Prova: O item 1 da proposição 12 segue diretamente da definição 10 e da proposição 10, e o item 2 deriva diretamente da proposição 11. □

Resumindo, a bissimulação temporal direta $(\overset{*}{\approx}_t)$ absorve as ações de violação temporal $(a^* \in Act^*)$ e as ocorrências da ação interna i. Porém, ela distingue ações observáveis $(a \in Act)$, e as ações devidas à progressão do tempo $(d \in D)$.

Para ilustrar a bissimulação temporal direta, considere-se o exemplo seguinte:

```
E = [p-d,p+d]a; stop ||| [p,p]i;E
```

```
F = ([p-d,p+d]a; stop <a] {a:(i;stop)} ||| [p,p]i;F
```

Desta forma, tem-se que $E \overset{*}{\approx}_t F$, Porém, $E \not\overset{*}{\approx}_t F$; isto deve-se ao fato de que o processo F transforma a ocorrência da violação temporal a^* em uma ação interna i.

II.2.6.4 Bissimulação Fraca

Nas seções anteriores foram apresentadas diversas definições de bissimulações no contexto de RT-LOTOS. Iniciou-se com a definição da bissimulação temporal forte (onde todas as ações eram consideradas), depois abstraiu-se das ações internas i e definiu-se a bissimulação temporal fraca, e, finalmente, abstraiu-se das ações internas e das ocorrências de violações temporais para definir-se a bissimulação temporal direta. Nesta seção, conclui-se este processo, e abstraindo-se dos arcos provenientes da passagem do tempo e da ação interna i, redefine-se, para o contexto de RT-LOTOS, o conceito de bissimulação fraca.

A definição da bissimulação fraca no contexto de RT-LOTOS tem como maior interesse a utilização desta linguagem formal em situações onde o tempo não afeta a correção do sistema. Visto que RT-LOTOS é uma extensão direta de LOTOS Básico, a bissimulação fraca também pode ser usada para inferir propriedades deste formalismo.

Na definição da bissimulação fraca são consideradas somente ações observáveis. As ações internas i e os arcos temporizados de D^ω não são levados em conta nesta bissimulação.

Definição 11 Define-se a relação de transição $\Rightarrow \subset \mathcal{E} \times \mathcal{E}$ como:

$E \stackrel{a}{\Rightarrow} F$ se, e somente se, $E(\stackrel{d_1}{\hookrightarrow})^* \stackrel{a}{\hookrightarrow} (\stackrel{d_2}{\hookrightarrow})^* F$, onde transições do tipo $(\stackrel{d_j}{\hookrightarrow})$, para $d_1, d_2 \in D^\omega$, representam arcos de passagem de tempo. □

Note-se que, em particular, $E \stackrel{\epsilon}{\hookrightarrow} E'$ se, e somente se, $E(\stackrel{d_1}{\hookrightarrow})(\stackrel{i}{\rightarrow})^*(\stackrel{d_2}{\hookrightarrow})E'$, para $d_1, d_2 \in D^\omega$.

Seja $\mathcal{L}^\Diamond = Act^\delta \cup Act^* \cup \{\epsilon\}$ e considere a relação de transição $\Rightarrow$, então obtém-se o sistema de transições rotuladas

$$\langle \mathcal{E}, \mathcal{L}^+, \{\stackrel{a}{\Rightarrow} : a \in \mathcal{L}^\Diamond\} \rangle$$

sobre expressões de comportamento. Desta forma, pode-se estabelecer a noção de equivalência desejada neste sistema.

Definição 12 Seja $\mathcal{L}^\Diamond = Act^\delta \cup Act^* \cup \{\epsilon\}$ o conjunto de ações. Então, define-se a função $\mathcal{F}^\Diamond$, sobre subconjuntos de $\mathcal{E} \times \mathcal{E}$ (isto é, relações binárias sobre expressões de comportamento), como segue. Se $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$, então $(P, Q) \in \mathcal{F}^\Diamond(\mathcal{R})$ se, e somente se, para toda ação $a \in \mathcal{L}^\Diamond$:

1. sempre que $P \stackrel{a}{\rightarrow} P'$ então, para algum Q', $Q \stackrel{a}{\Rightarrow} Q'$ e $(P', Q') \in \mathcal{R}$, e
2. sempre que $Q \stackrel{a}{\rightarrow} Q'$ então, para algum P', $P \stackrel{a}{\Rightarrow} P'$ e $(P', Q') \in \mathcal{R}$

$\mathcal{R}$ é chamada uma bissimulação fraca se, e somente se, $\mathcal{R} \subseteq \mathcal{F}^\Diamond(\mathcal{R})$. Se $(P, Q) \in \mathcal{R}$ para alguma bissimulação fraca $\mathcal{R}$, então P e Q são ditos fracamente bissimilares, que é representada simbolicamente por $P \approx Q$. Onde: $\approx = \bigcup\{\mathcal{R} : \mathcal{R} \text{ é uma bissimulação fraca}\}$. □

Proposição 13 Seja $\mathcal{L}^+ = Act^\delta \cup Act^* \cup \{\epsilon\}$ e seja $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ uma bissimulação fraca. Então $\langle P, Q \rangle \in \mathcal{R}$ implica que:

1. sempre que $P \stackrel{a}{\rightarrow} P'$ então, para algum Q', $Q \stackrel{a}{\Rightarrow} Q'$ e $(P', Q') \in \mathcal{R}$, e

2. sempre que $Q \xrightarrow{a} Q'$ então, para algum P', $P \stackrel{a}{\Rightarrow} P'$ e $(P', Q') \in \mathcal{R}$

Prova: Por hipótese, $\mathcal{R}$ é uma bissimulação fraca, então $\mathcal{R} \subseteq \mathcal{F}^{+}(\mathcal{R})$ o que implica que $(P, Q) \in \mathcal{F}^{\Diamond}(\mathcal{R})$, logo, tem-se (1) e (2). □

Proposição 14 A relação $\approx$ tem as seguintes propriedades:

1. $\approx$ é a maior bissimulação fraca
2. $\approx$ é uma relação de equivalência

Prova: Similar à de proposição equivalente apresentada em [Milner 89]. □

Resumindo, a bissimulação fraca ($\approx$) absorve as ocorrências da ação interna i e as ações devidas à progressão do tempo ($d \in D$). Neste caso, somente são distinguíveis as ações observáveis ($a \in Act$) e as ações de violação temporal ($a^* \in Act^*$). Esta bissimulação coloca numa mesma classe de equivalências todas as expressões de comportamentos que diferem apenas pelos limites (finitos) de seus intervalos.

Pela bissimulação fraca, os processos $E = a; H$, $F = i; a; H$ e $G = [t_1, t_2]i; a; H$ são equivalentes, isto é, $E \approx F \approx G$.

Com a apresentação das bissimulações acima fica então completamente definida a álgebra de processos temporizada RT-LOTOS, como sendo uma quádrupla composta de um conjunto de operadores, um conjunto de nomes de ações, um conjunto de regras de semântica operacional e uma equivalência comportamental sobre modelos.

II.3 Representações das restrições temporais básicas em RT-LOTOS

Nesta seção, discutiremos as principais características de RT-LOTOS do ponto de vista temporal, destacando a expressividade, facilidade de uso e alto poder intuitivo e simplificador das escolhas feitas e dos operadores subseqüentes introduzidos no formalismo LOTOS. Algumas operações temporais ("delay", "timeout", "watchdogs", periodicidade) exemplificarão estas características. A seguir, apresentaremos alguns exemplos de comportamentos e situações que são de grande interesse em sistemas tempo-real críticos e outros sistemas dependentes do tempo como por exemplo os sistemas multimídia, e mostraremos como RT-LOTOS pode representar tais situações.

II.3.1 Operações básicas

Restrição de Tempo Em RT-LOTOS, a imposição de restrições temporais às ocorrências de ações é feita de maneira clara e direta pela simples atribuição de um intervalo de tempo a uma ação, da forma "$[t_{min}, t_{max}]a$". A ação só poderá ocorrer depois de ser retardada durante t_{min} unidades de tempo e antes de expirar o tempo máximo t_{max} no qual ela poderia se realizar. Entretanto, não há nenhuma exigência para a ação a ter que ocorrer dentro daquele intervalo.

Violação temporal A ocorrência da ação temporizada $[t_{min}, t_{max}]a$ depende sempre do ambiente e este pode não estar apto a realizá-la dentro do intervalo. Neste caso, a ação a não ocorrerá, sendo violada a restrição de tempo imposta a ação. Um evento que sinaliza a ocorrência de tal violação de tempo a^* faz parte da semântica de RT-LOTOS. O tratamento das violações de tempo é facultativa podendo ser realizada pelo operador de tratamento de exceções temporais discutido abaixo.

Tratamento de exceção temporal RT-LOTOS possui um poderoso operador de tratamento de exceções temporais já apresentado, que permite tratar de forma simple e estruturada os eventos relacionados à não ocorrência, dentro dos intervalos estabelecidos, de ações em um processo. O operador de tratamento de exceções temporais (ou preempção temporal) pode ser considerado como um processo supervisor de alto nível que, a cada instante, está apto a detectar a ocorrência de uma violação temporal e tratá-la imediatamente através da desabilitação do processo que a gerou e a inicialização de um processo específico para tratamento desta.

Retardo ("delay") Um retardo de $t \in D^{\omega}$ unidades de tempo imposto a um processo P pode ser representado por:

```
Q:=[t]i;P
```

Processo periódico Sejam Q um processo e $t \in D^{\omega}$ uma quantidade de tempo, então um processo P que representa o lançamento do processo Q a cada período de tempo t pode ser representado por:

```
P:=Q|||[t]i;P
```

Timeout Sejam P e Q dois comportamentos, e $t \in D^{\omega}$. Então, um *timeout* é um mecanismo dependente do tempo que se comporta como P, se uma ação inicial de P ocorre até o instante t, ou como Q após o tempo t. No nosso formalismo, um mecanismo de *timeout* pode ser modelado como:

```
P [] [t]i;Q
```

O formalismo RT-LOTOS também permite representar um mecanismo que trata timeouts diferentes em um conjunto de ações particulares dentro de um processo arbitrário. Como exemplo, seja a uma ação especificada (em possivelmente diferentes pontos) de um processo P com intervalo(s) de tempo do tipo $[t_{min}, t_{max}]$, com $t_{max} < \omega$ e seja Q um outro comportamento (um tratador de exceções, por exemplo). Então, o processo

```
P <a] {a:Q }
```

comporta-se como P se nenhuma ação de violação temporal a^* ocorrer durante a realização neste processo. No caso da ocorrência de uma violação temporal a^* (isto é, a não realização de a dentro do(s) intervalo(s) de tempo associado(s) a ela), então o processo P é descontinuado e o processo Q é iniciado.

Watchdog Em RT-LOTOS um mecanismo de *watchdog* pode ser modelado utilizando o operador de preempção, da maneira que segue:

```
P [> [t]i;Q
```

onde P e Q representam respectivamente o comportamento normal e o de exceção e $t \in D^{\omega}$. O processo resultante se comporta como P até o instante t, após o que P é abortado, e Q é iniciado.

II.3.2 Alguns exemplos simples de aplicação

Recuperação após sinalização de falha temporal: Considere-se um processo P de um sistema tempo-real em que uma dada ação a seja considerada imprescindível e devendo satisfazer restrições de tempo do tipo $[t_{min}, t_{max}]$ em cada uma das suas ocorrências no processo. Deseja-se representar o seguinte comportamento: uma eventual não realização da ação a por violação da restrição de tempo no processo P deve provocar a realização das três etapas seguintes:

1. a instância do processo P, em execução, deve ser abortada,
2. um processo de recuperação Q deve ser lançado, e em seguida, após a recuperação deste
3. uma outra instância do processo P deve ter início.

De forma genérica, a especificação de um tal comportamento em RT-LOTOS pode ser facilmente representada por:

```
P <a] {a:(Q >> P)}
```

Assim, na primeira vez que ocorrer uma violação temporal (ação a^*) no processo P o processo "`(Q >> P)`" será iniciado. No caso da não ocorrência de a^* em P o processo "`(Q >> P)`" nunca terá inicio, e se P terminar, então o processo "`P <a] {a:(Q >> P)}`" terminará também.

Fluxo periódico com "*jitter*": Deseja-se especificar um fluxo periódico infinito com "*jitter*" (neste exemplo não são considerados derivas permanentes oriundas de atrasos), sendo que a ação a representa a unidade de informação do fluxo [Carmo 92].

Um tal fluxo periódico de período p com jitter d ($d < \frac{p}{2}$) pode ser especificado pelo seguinte processo:

```
fluxo := [p-d,p+d]a;stop ||| [p]i;fluxo
```

A recursão define a repetição do processo $fluxo$ a cada p unidades de tempo pois a ação interna i é urgente após o retardo de p unidades de tempo. Por outro lado, a ação a pode ocorrer em qualquer tempo dentro do intervalo $[p-d, p+d]$, que caracteriza o "jitter" associado a ocorrência de uma instância de uma unidade de informação. A não ocorrência da ação a dentro deste intervalo induz a ocorrência da violação temporal a^* no final do intervalo. Como consequência ainda, ocorrências de a^* caracterizam perdas de unidades de informação que podem ser tratadas num nível superior.

Fluxo com relações de dependência entre unidades de informação: Deseja-se especificar um fluxo onde são definidas certas relações de dependência entre as unidades de informação. Uma trama transporta informações que caracterizam as modificações de uma imagem com relação com relação a uma imagem de referência; é interessante neste caso de poder evitar o fornecimento de tramas dependentes das tramas de referência que não puderam ser enviadas.

Deseja-se representar um fluxo-vídeo no qual existe uma trama de referência representada pela ação `m` no instante `d` e duas outras tramas que dependem da trama (*frame*) de referência e que serão representadas pelas ações `a` e `b` respectivamente nos instantes 2d e 3d. Além disso, quando a trama de referência `m` ou uma das outras tramas `a` e `b` não é fornecida, a execução do processo em curso é abandonada por um tempo equivalente ao tempo necessário à chegada da proxima trama de referência (respectivamente 2d, `d` ou 0).

A especificação RT-LOTOS é dada por:

```
video := ([d]m;[d]a;[d]b;video)
        <m,a,b>
          {m:[2d]i;video, a:  [d]i;video, b:video }
```

O exemplo do cruzamento de uma estrada de ferro: A seguir, para poder mostrar a expressividade de RT-LOTOS considerar-se-á o conhecido exemplo do cruzamento de uma estrada de ferro. A versão do exemplo aqui considerada é uma versão mais detalhada daquela apresentada em [Alur 92]; aqui, consideramos alguns eventos relacionados ao mal funcionamento e a eventuais falhas as quais o sistema estaria sujeito. O sistema é composto de três componentes: `Train`, `Gate` e `Controller`.

O conjunto de ações do processo `Train` é {`approach`, `in`, `out`, `leaving`, `stop_the_train`}. O processo `Train` se comunica com o processo `Controller` com duas ações `approach`, que sinaliza a aproximação do trem, e `leaving`, que notifica sua saída do cruzamento. As ações `in` e `out` marcam os eventos de entrada e saída do trem do cruzamento da estrada de ferro. A ação `stop_the_train` representa uma eventual parada de emergência do trem que teria como objetivo evitar um acidente no cruzamento. O trem deve sinalizar sua aproximação para o processo `Controller` com no mínimo 2 e no máximo 5 segundos de antecedência da sua entrada no cruzamento. Sabe-se também que o trem sai do cruzamento em no máximo 3 segundos após ter entrado, e que, por razões de segurança, a sinalização de sua saída é feita no mínimo 1 segundo após esse evento. Além disso, o processo `Train` está sujeito a receber o sinal `stop_the_train` do `Controller` a qualquer instante, e quando isto ocorrer, o trem deverá parar imediatamente (para simplificar o exemplo é assumido que o Trem para instantaneamente ao receber o sinal correspondente do Controller).

```
Process Train :=
     TrainMotion  [>  Emergency

Where
     Process TrainMotion :=
        approach;
        [2,5]in;
        [0,3]out;
        [1,w]leaving;
        TrainMotion
     Endproc

     Process Emergency :=
        stop_the_train;
        stop
     Endproc
Endproc
```

O conjunto de ações do processo `Gate` é {`lower`, `down`, `raise`, `up`, `siren`}. O processo `Gate` comunica-se com o processo `Controller` por meio das ações `lower` (abaixar o portão) e `raise` (levantar o portão). Os eventos `up` e `down` denotam a abertura e fechamento do portão do cruzamento. O processo `Gate` responde à ação `lower` fechando o portão em, no máximo, 1 segundo, e responde à ação `raise` num instante entre 1 e 2 segundos. O portão está sujeito a falhas e pode não fechar no tempo requisitado, e este evento deve fazer soar uma sirene de alarme e comunicar tal situação de emergência ao processo `Controller`, tais eventos são representados pela ação `siren`. Na descrição do processo `Gate` foi utilizado o operador de preempção temporal que descreve com simplicidade a falta temporal da ação `down` e o seu tratamento.

```
Process Gate :=
     GateControl < down ] {down: Alarm}

Where
     Process GateControl :=
        lower;
        [0,1]down;
        GateControl
     []
        raise;
        [1,2]up;
        GateControl
     Endproc

     Process Alarm :=
        [0]siren;
        stop
     Endproc
Endproc
```

As ações do processo `Controller` são {`approach`, `leaving`, `lower`, `raise`, `siren`, `stop_the_train`}. Sempre que ocorre uma ação `approach`, o processo `Controller` responde com uma ação `lower`. O tempo de resposta é 1 segundo. Sempre que ocorre uma ação `leaving`, o processo `Controller` responde com uma ação `raise` em no máximo 1 segundo. O processo `Controller` também está sempre pronto para tratar exceções de emergência provocadas por pane ou mal funcionamento do processo `Gate`, e fica sempre pronto para realizar uma ação `siren` provocada pelo processo `Gate` e responder de imediato com uma ação `stop_the_train` que deverá ser tratada pelo processo `Train`.

```
Process Controller :=
     Control [> Exception

Where
     Process Control :=
        approach;
        [1]lower;
        Control
     []
        leaving;
        [0,1]raise;
        Control
     Endproc

     Process Exception :=
        siren;
        [0]stop_the_train;
        stop
     Endproc
Endproc
```

O processo composto "RailRoadCrossing" é mostrado abaixo ao lado do processo Controller:

```
Process RailRoadCrossing :=
       (Train
     |[approach, leaving, stop_the_train]|
        Controller)
     |[lower, raise, siren]|
        Gate
Endproc
```

II.4 A definição formal de RT-LOTOS em Grafos Temporizados

Geralmente, sistemas de transições rotuladas são os formalismos básicos mais utilizados para representação da semântica formal de álgebras de processos temporizadas (ex.: [Quemada 89, Bolognesi 90, Hansson 90, Moller 90, Nicollin 92, Leduc 92 etc]). Porém, as técnicas de verificação existentes para esta abordagem (ex.: verificação de bissimulações) são baseadas na enumeração de todos os estados do sistema de transições que representa o sistema especificado, e no caso temporizado isto exige que se leve em conta todos os estados alcançados inclusive por transições devidas à progressão do tempo. Desta maneira, o número de estados a serem tratados pelos sistemas de transições originados de álgebras de processos temporizadas é muito elevado, se comparado a uma álgebra de processos sem tempo. Além do mais, este número de estados cresce mais drasticamente ainda quando o domínio de tempo

considerado na álgebra de processos temporizada é denso; isto implicaria, por exemplo no domínio temporal dos racionais, que se teria um número infinito de estados originados por transições de progressão do tempo para qualquer intervalo finito de tempo. Por sua vez, uma das características mais importantes dos *Grafos Temporizados* quando comparados aos *Sistemas de Transições Rotuladas* é que o primeiro formalismo absorve nas condições de sensibilização todas as transições de progressão do tempo que existem no último formalismo. Desta maneira, a utilização de grafos temporizados como formalismo básico subjacente a uma álgebra de processos temporizada possibilita a "eliminação" dos estados intermediários alcançados por transições de progressão do tempo e a conseqüente redução dos grafos. Por esta razão, a alternativa de definir uma semântica formal de RT-LOTOS permite o tratamento de sistemas dependentes do tempo especificados em RT-LOTOS.

Nesta seção é apresentada a definição formal da semântica de RT-LOTOS no formalismo Grafo Temporizado. Inicialmente, os grafos temporizados são definidos formalmente. Em seguida, define-se a semântica operacional deste formalismo. Depois, a semântica de RT-LOTOS é definida em grafos temporizados. A seguir, apresenta-se um pequeno exemplo para ilustrar a nova representação subjacente à RT-LOTOS. Finalmente, discute-se o interesse e as vantagens, do ponto de vista de verificação, de ter uma representação em grafos temporizados de RT-LOTOS.

II.4.1 Grafos Temporizados

O formalismo básico *Grafo Temporizado* [Alur 90] constitui-se atualmente num importante método alternativo para descrição de sistemas com restrições temporais. Neste formalismo, as mudanças de estado são realizadas por ações instantâneas e não existem transições relativas ao progresso do tempo.

Para modelar um sistema usando grafos temporizados, atribui-se a ele um conjunto finito de *nodos* e um conjunto finito de *relógios* com valores tomados num dado domínio de tempo D arbitrário. As transições que um sistema pode realizar dependem dos valores dos relógios. Simultaneamente à realização de uma transição, alguns dos relógios podem ser reinicializados. Em qualquer instante, a leitura de um relógio é igual ao tempo passado desde a última vez que ele foi reinicializado. Restrições de tempo podem ser expressas associando condições de habilitação às transições.

II.4.1.1 Definições

Definição 13 Seja A um vocabulário de ações no qual a denota um elemento de A. Uma tupla $T = (t_1, \cdots, t_n)$ de "*relógios*" é um conjunto de variáveis tomando seus valores em D. Representa-se por $\mathcal{B}(T)$ o conjunto dos predicados b sobre T, isto é, o conjunto das aplicações de D^n em $\{tt, ff\}$. Além disso, $\mathcal{F}(T)$ é o conjunto das aplicações f de D^n em D^n.

Um *Grafo Temporizado* é uma estrutura $G = (N, T, n_0, \mathtt{urg}, \rightarrow)$, onde

- N é um conjunto finito de *nodos*
- T é uma tupla de *relógios* (*timers*)
- $n_0 \in N$ é o *nodo inicial*

- urg é uma aplicação de N em $\{tt, ff\}$, chamada de *função de urgência*
- $\rightarrow \subseteq N \times A \times \mathcal{B}(T) \times \mathcal{F}(T) \times N$ é a relação de transição

  Escreve-se $n \xrightarrow{a,b,f} n'$ ao invés de $(n, a, b, f, n') \in \rightarrow$. □

Os relógios de um grafo temporizado são variáveis que podem ser incrementadas "continuamente" e na mesma velocidade. Inicialmente, todos os relógios têm valor 0. Os nodos representam os estados de controle. De um nodo n, se a condição de sensibilização b de uma transição $n \xrightarrow{a,b,f} n'$ é satisfeita pelos valores dos relógios, então a transição *pode* ser executada. Isto é, a ação a pode ser realizada, e o estado de controle torna-se n' após a modificação dos valores dos relógios de acordo com a função f.

Comentário: Como o objetivo final da apresentação dos grafos temporizados é a definição da semântica de RT-LOTOS neste modelo, será feita a seguir uma rápida descrição de como esta semântica é mapeada em grafos temporizados.

- Os nodos n de um grafo temporizado representam expressões de comportamento definidas em RT-LOTOS, nos seus vários estágios de seu processo de execução. Abstrai-se do tempo para esta representação.
- A tupla de relógios associada a um grafo temporizado representam os tempos locais relativos às várias ações que estão sendo oferecidas no grafo da especificação RT-LOTOS.
- O nodo n_0 representa a especificação inicial em RT-LOTOS.
- O conjunto $\mathcal{B}(T)$ representa as diversas restrições de tempo impostas para realização das transições.

Definição 14 Define-se a notação $A(G)$ para representar o conjunto que define o vocabulário de ações de um dado grafo temporizado $G = (N, T, n_0, \mathtt{urg}, \rightarrow)$. Formalmente,

$$A(G) = \{a | \forall n, n' \in N,\ n \xrightarrow{a,b,f} n' \in \longrightarrow\}$$

□

Definição 15 Para qualquer predicado $b \in \mathcal{B}(T)$, o predicado $\hat{b}$ (lê-se b-chapéu) é definido por

$$\hat{b}(T) = (\exists d \in D)\ \ b(T + d)$$

onde $T + d = (t_1 + d, \cdots, t_n + d)$. Note que $\hat{b}$ é um predicado significando "eventualmente b", isto é, "b é verdade ou deverá sê-lo no futuro". □

Definição 16 Para um nodo n em N e uma ação $a \in A(G)$, define-se a *condição de sensibilização do nodo n na ação a*, denotado por $\mathtt{en}(n)(a)$, caracterizando os valores dos relógios para os quais a ação a pode ser executada a partir de n:

$$\mathtt{en}(n/a) \stackrel{def}{=} \left\{ b \mid n \xrightarrow{a,b,f} \right\}$$

□

Estendendo o conceito da definição anterior, tem-se a seguinte definição:

Definição 17 Define-se a *condição de sensibilização de um nodo* $n \in N$, denotado por $\mathtt{en}(n)$:

$$\mathtt{en}(n) \stackrel{def}{=} \bigvee \left\{ b \mid n \xrightarrow{a,b,f} \right\}$$

□

Note-se que $\mathtt{en}(n)$ caracteriza os valores dos relógios para os quais uma transição pode ser executada a partir do nodo n.

Finalmente, define-se o conceito de *condição de atividade* de um nodo n como:

Definição 18 A *condição de atividade* de um nodo n, denotada por $\mathtt{act}(n)$, é definida como:

$$\mathtt{act}(n) \stackrel{def}{=} \widehat{\mathtt{en}(n)} \wedge \neg \mathtt{urg}(n)$$

□

O predicado $\mathtt{act}(n)$ é quem dita se uma dada transição *pode, não pode,* ou *deve* ser realizada para um dado valor de uma tupla de relógios.

II.4.1.2 Semântica Operacional de Grafos Temporizados

O predicado act é definido de maneira que o sistema possa permanecer no nó n no instante definido pelo valor de T somente se $\mathtt{act}(n)(T) = tt$. Se a condição de urgência $\mathtt{urg}(n)$ é ff, então não se exige realizações urgentes de transições no nodo n. Inversamente, $\mathtt{urg}(n) = tt$ fará com que o predicado $\mathtt{act}(n)$ seja ff e isto forçará a realização de uma transição de n, provocada pela supressão da progressão do tempo no modelo operacional de grafos temporizados (observe-se as regras que definem operacionalmente os grafos temporizados).

Formalmente, o modelo operacional de um grafo temporizado, $G = (N, T, n_0, \mathtt{urg}, \rightarrow)$ é um sistema de transições rotuladas,

$$STR_G = \langle N \times T, A \cup D_*, \{\xrightarrow{a} : a \in A(G)\} \rangle$$

Ou seja, os estados são pares (n, T), os rótulos são elementos de $A \cup D_*$ (onde $D_* \stackrel{def}{=} D - \{0\}$), e o estado inicial é $(n_0, \vec{0})$, onde n_0 é o nodo inicial do grafo temporizado. A relação de transição é definida pelas regras seguintes.

$$\frac{n \xrightarrow{a,b,f} n' \ , \ b(T)}{(n,T) \xrightarrow{a} (n', f(T))} \qquad \frac{\mathtt{act}(n)(T+d)}{(n,T) \xrightarrow{d} (n, T+d)}$$

A primeira regra diz que uma transição é executável se o seu predicado de sensibilização é avaliado como sendo verdadeiro. A segunda regra formaliza a condição à satisfazer para permanecer em um nodo esperando d unidades de tempo.

A noção de equivalência por bissimulação forte entre dois grafos temporizados G e H pode ser definida [Godskesen 92] da maneira usual sobre a união disjunta de STR_G e STR_H tal que G e H são fortemente equivalentes, isto é $G \sim H$, sempre que $((n_G, \vec{0}), (n_H, \vec{0}))$ pertençam a uma bissimulação forte, onde n_G e n_H são os nodos iniciais de G e H respectivamente. Similarmente, equivalência por bissimulação forte entre um processo E e um grafo temporizado G pode ser definida pela união disjunta de STR_E e STR_G. Escreve-se $E \sim G$ sempre que existir uma bissimulação forte contendo $(E, (n_G, \vec{0}))$.

Grafos temporizados podem ser vistos como uma representação alternativa correta para processos temporizados visto que a interpretação de uma representação em grafos temporizados de um processo RT-LOTOS é fortemente equivalente à interpretação operacional do processo (pois as definições são postas de maneira a satisfazer isto).

II.4.2 Definindo RT-LOTOS como Grafos Temporizados

Nesta seção define-se como construir grafos temporizados $G[\![E]\!]$ para expressões de comportamento E de RT-LOTOS de maneira que $E \sim G[\![E]\!]$. Mais precisamente, define-se uma álgebra de grafos temporizados sobre os operadores de RT-LOTOS.

Observação preliminar: Deve-se observar que o mapeamento de RT-LOTOS em grafos temporizados não é total. Isto deve-se ao fato de que grafos temporizados só são capazes de representar sistemas com o número finito de estados (mesmo sendo este formalismo capaz de absorver todos os estados intermediários devidos às transições temporais). Em outras palavras um sistema só poderá ter uma uma representação em grafos temporizados se a especificação RT-LOTOS do sistema sem a sua dimensão temporal levar a um sistema de transições finito. Assim, existem expressões de comportamento RT-LOTOS que apesar de válidas que não têm representação definida em grafos temporizados. Um exemplo de tal situação é o processo

$$P = Q|||i; P$$

Este processo não possui representação por um sistema de transições finito, e assim não é possível representá-lo como um grafo temporizado. Observe-se que esta é uma limitação que vale tanto para RT-LOTOS quanto para LOTOS. Em vista disto, as expressões de comportamento RT-LOTOS que são definíveis em grafos temporizados constituem-se num subconjunto próprio de RT-LOTOS que é definido levando-se em conta as duas seguintes condições:

- Toda expressão recursiva deve ser guardada.
- Toda sub-expressão da forma $A|[L]|B$ é tal que todas as derivações de A e de B não contenham $A|[L]|B$ como sub-expressão.

Estas condições são suficientes para excluir os comportamentos infinitos de LOTOS (e também de RT-LOTOS) e são as mesmas que foram estabelecidas e adotadas em [Garavel 89b, Marsan94 e Daws 94] em mapeamentos de LOTOS (e extensões temporais de LOTOS) em outros sistemas de estados finitos.

II.4.2.1 Operações sobre Grafos Temporizados

Para facilitar e padronizar as definições desta seção considerem-se $G_E = (N_E, T_E, n_E^0, \mathtt{urg}_E, \rightarrow_E)$ e $G_F = (N_F, T_F, n_F^0, \mathtt{urg}_F, \rightarrow_F)$ como dois grafos temporizados com o vocabulário $Act^{i,\delta} \cup Act^*$ e tal que $N_E \cap N_F = \emptyset$ e $T_E \cap T_F = \emptyset$. Também, $\mathcal{R}(T)$ denota a função de reinicialização do conjunto de *relógios* T; $\mathcal{R}(\emptyset)$ significa que nenhum relógio é reinicializado. Para simplificar a notação, representar-se-á por f a função que reinicializa todos os relógios de um grafo numa dada transição.

II.4.2.1.1 Inação: $G[\![stop]\!] = (N, T, n_0, \mathtt{urg}, \emptyset)$ com $N = \{n_0\}$, um único relógio $T = t$, $\mathtt{urg}(n_0) = ff$; e, evidentemente, não existe nenhuma relação de transição associada com $G[\![stop]\!]$.

II.4.2.1.2 Terminação com sucesso: $G[\![exit]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com $N = \{n_0, n_1\}$, $T = t$ e $\mathtt{urg}(n_0) = \mathtt{urg}(n_1) = ff$ onde n_1 é nodo sumidouro. A relação de transição $\rightarrow$ é definida como o menor conjunto que satisfaz:

$$n_0 \xrightarrow{\delta, t \geq 0} n_1$$

Prefixação: A operação de prefixação desempenha um papel importante em RT-LOTOS e funciona como se fosse um "motor" básico no qual estão baseados todos os mecanismos temporais do formalismo. É na operação de prefixação que são especificados e tratados os diversos tipos de restrições temporais impostas às ocorrência de ações. Em termos de Grafos Temporizados para RT-LOTOS, as restrições de Tempo são definidas a partir dos intervalos temporais associados a cada transição. Por exemplo, a prefixação $[t_1, t_2]a; E$ define o estabelecimento de um predicado do tipo $t_1 \leq t \leq t_2$ e teria-se desta maneira duas transições possíveis: uma representando a realização da ação a, isto é, $n \xrightarrow{a, t_1 \leq t < t_2, \{t\}} n'$ (para n' representando o grafo de E); e a transição $n \xrightarrow{a^*, t = t_2, \{t\}} n^*$ que representa a realização da ação a^* (para n^* representando um estado do tipo *sumidouro*).

Observe-se que as condições lógicas b que devem ser satisfeitas para realização de uma transição em um grafo temporizado são definidas a partir dos intervalos temporais associados às ações em operações de prefixação. Assim, os b's usados nas regras de transição, a serem apresentadas a seguir, correspondem exatamente às restrições de tempo impostas a realização de uma ação a através do intervalo temporal associado a esta ação.

Cada operação de prefixação possui um relógio t associado a ela. Este relógio é inicializado em zero no instante inicial do oferecimento da ação sendo prefixada, e é incrementado de acordo com a evolução do tempo. Este relógio é usado para definir formalmente os valores do predicado b associado a uma ação, e dos predicados $\widehat{\mathtt{en}(n)}$ e $\mathtt{urg}(n)$ associados aos nodos do grafo que representa a prefixação (como pode ser visto na tabela II.1).

O Grafo Temporizado para a ação prefixada $\alpha \in Act^i$ é:
$G[\![[t_1, t_2]\alpha; E]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com:

$N = N_E \cup \{n_0, n^*\}$ $(n_0, n^* \notin N_E)$

$[t_1, t_2]\alpha \equiv$	$\mathtt{urg}(n_0) =$	$\mathtt{en}(n_0) =$	$\widehat{\mathtt{en}(n_0)} =$	$\mathtt{act}(n_0) =$
$[0,0]i \equiv i$ ou $[0,0]a$	tt, para $t \geq 0$	tt, em $t = 0$ ff, para $t > 0$	tt	ff, para $t \geq 0$
$[0,\omega]i$ ou $[0,\omega]a \equiv a$	ff, para $t \geq 0$	tt, para $t \geq 0$	tt	tt, para $t \geq 0$
$[t_1,\omega]i$ ou $[t_1,\omega]a$	ff, para $t \geq 0$	ff, para $t < t_1$ tt, para $t \geq t_1$	tt	tt, para $t \geq 0$
$[t_1,t_2]i$ ou $[t_1,t_2]a$	ff, para $t < t_2$ tt, para $t \geq t_2$	ff, para $t < t_1$ tt, para $t_1 \leq t \leq t_2$ ff, para $t > t_2$	tt	tt, para $t < t_2$ ff, para $t \geq t_2$

Tabela II.1: Valores de $\mathtt{urg}(n_0)$, $\mathtt{en}(n_0)$, $\widehat{\mathtt{en}(n_0)}$ e $\mathtt{Act}(n_0)$ para vários tipos de intervalos e ações

$T = T_E \cup \{t\}$, onde t é o relógio que registra a idade do oferecimento da ação α, e t pode estar em T_E.

Se $n \in N_E$, então $\mathtt{urg}(n) = \mathtt{urg}_E(n)$

$\mathtt{urg}(n_0) = ff$, $\mathtt{urg}(n^*) = ff$

A relação transição $\rightarrow$ é definida como o menor conjunto que satisfaz:

$$n_0 \xrightarrow{a, t_1 \leq t < t_2, \mathcal{R}(T_E)} n_E^0$$

$$n_0 \xrightarrow{a^*, t = t_2, \mathcal{R}(T_E)} n^* \quad (\text{para } a \neq i)$$

$$\frac{n \in N_E \ , \ n \xrightarrow{a,b,f}_E n'}{n \xrightarrow{a,b,f} n'}$$

II.4.2.1.3 Escolha: O grafo temporizado é: $G[\![E[\,]F]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com

$N = N_E \cup N_F \cup (N_E \times N_F)$

$T = T_E \cup T_F$

$n_0 = (n_E^0, n_F^0)$

Se $n \in N_j$, então $\mathtt{urg}(n) = \mathtt{urg}_j(n)$, onde $j \in \{E, F\}$

$\mathtt{urg}(n_E, n_F) = \mathtt{urg}(n_E) \vee \mathtt{urg}(n_F)$

A relação de transição $\rightarrow$ é o menor conjunto que satisfaz:

$$\frac{n \xrightarrow{a,b,f}_j n' \ , \ a \in Act^{i,\delta} \cup Act^*}{n \xrightarrow{a,b,f} n'}$$

$$\frac{n_E \xrightarrow{a,b,f}_E n'_E \ , \ a \in Act^{i,\delta}}{(n_E^0, n_F^0) \xrightarrow{a, b \wedge \mathtt{act}(n_F^0), f} n'_E}$$

$$\frac{n_F \xrightarrow{a,b,f}_F n'_F \ , \ a \in Act^{i,\delta}}{(n_E^0, n_F^0) \xrightarrow{a,b \wedge \mathtt{act}(n_E^0),f} n'_F}$$

$$\frac{n_E \xrightarrow{a^*,b,f}_E n'_E}{(n_E, n_F) \xrightarrow{a^*,b \wedge \mathtt{act}(n_F),\mathcal{R}(\emptyset)} (n'_E, n_F)}$$

$$\frac{n_F \xrightarrow{a^*,b,f}_F n'_F}{(n_E, n_F) \xrightarrow{a^*,b \wedge \mathtt{act}(n_E),\mathcal{R}(\emptyset)} (n_E, n'_F)}$$

Comentário: Observe-se que nas duas últimas regras (as que correspondem a evolução do sistema pela ação a^*) as funções de reinicialização de relógios são ambas iguais a $\mathcal{R}(\emptyset)$ o que corresponde a não reinicializar nenhum relógio. Isto foi feito para permitir que a semântica da realização de uma ação a^* no operador de escolha ficasse bem definida, isto é, a^* não decide uma escolha. Observe-se também que não existe necessidade de reinicializar os relógios do processo que realiza a ação a^* visto que o comportamento que a segue é sempre equivalente funcionalmente ao processo *stop*.

Ocultação: O grafo temporizado é: $G[\![\text{hide } L \text{ in } E]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com:

$N = N_E$

$T = T_E$

$n_0 = n_E^0$

$\mathtt{urg}(n) = \bigvee(\{a | n \xrightarrow{a,b,f}_E n'\} \subseteq L)$

A relação de transição $\rightarrow$ é definida como o menor conjunto que satisfaz:

$$\frac{n \xrightarrow{a,b,f}_E n' \ , \ (a \in L)}{n \xrightarrow{i,b,f} n'}$$

$$\frac{n \xrightarrow{a,b,f}_E n' \ , \ (a \notin L)}{n \xrightarrow{a,b,f} n'}$$

Comentário: Observe-se que a definição do predicado urg acima impõe a urgência requerida às ações ocultadas em um grafo temporizado.

Composição Paralela: O grafo temporizado é: $G = [\![E|[L]|F]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com:

$N = N_E \times N_F$

$T = T_E \cup T_F$

$n_0 = (n_E^0, n_F^0)$

$\mathtt{urg}(n_E, n_F) = (\{a | n_E \xrightarrow{a,b,f}_E n'_E\} \cap \{a | n_F \xrightarrow{a,b,f}_F n'_F\} \subseteq L)$

Comentário: Observe-se que a definição do predicado urg acima impõem a urgência requerida às ações a sincronizar em um grafo temporizado somente quando ambos os nodos do grafo estejam prontos para isto.

A relação de transição $\rightarrow$ é definida como o menor conjunto que satisfaz as regras apresentadas a seguir:

- Caso da ausência de sincronização:

$$\frac{n_E \xrightarrow{a,b,f}_E n'_E \ , \ a \in Act^i \cup Act^* - L}{(n_E, n_F) \xrightarrow{a,b \wedge \mathtt{act}(n_F),f} (n'_E, n_F)}$$

$$\frac{n_F \xrightarrow{a,b,f}_F n'_F \ , \ a \in Act^i \cup Act^* - L}{(n_E, n_F) \xrightarrow{a,b \wedge \mathtt{act}(n_E),f} (n_E, n'_F)}$$

- Caso da presença de sincronização:
  - Ações normais:

$$\frac{n_E \xrightarrow{a,b_E,f_E}_E n'_E \ , \ n_F \xrightarrow{a,b_F,f_F}_F n'_F \ , \ a \in L \cup \{\delta\}}{(n_E, n_F) \xrightarrow{a,b_E \wedge b_F, f_E \star f_F} (n'_E, n'_F)}$$

  - Violações temporais:

$$\frac{n_E \xrightarrow{a^*,b_E,f_E}_E n'_E \ , \ n_F \xrightarrow{a^*,b_F,f_F}_F n'_F \ , \ a \in L}{(n_E, n_F) \xrightarrow{a^*,b_E \wedge b_F, f_E \star f_F} (n'_E, n'_F)}$$

$$\frac{n_E \xrightarrow{a^*,b_E,f_E}_E n'_E \ , \ \neg(\mathtt{en}(n_F/a) \vee \mathtt{en}(n_F/a^*)) \ , \ a \in L}{(n_E, n_F) \xrightarrow{a^*,b_E \wedge \mathtt{act}(n_F),f_E} (n'_E, n_F)}$$

$$\frac{n_F \xrightarrow{a^*,b_F,f_F}_F n'_F \ , \ \neg(\mathtt{en}(n_E/a) \vee \mathtt{en}(n_E/a^*)) \ , \ a \in L}{(n_E, n_F) \xrightarrow{a^*,b_F \wedge \mathtt{act}(n_E),f_F} (n_E, n'_F)}$$

Nas regras acima, $f_E \star f_F$ denota uma função que age como f_E em T_E e como f_F em T_F.

Composição seqüencial: O grafo temporizado é: $G[\![E \gg F]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com:

$N = N_E \cup N_F \cup (N_E \times \{n_F^0\})$

$T = T_E \cup T_F \cup \{t\}$, t podendo pertencer a T_E

$n_0 = (n_E^0, n_F^0)$

Se $n \in N_j$, então $\mathtt{urg}(n) = \mathtt{urg}_j(n)$, onde $j \in \{E, F\}$

$\mathtt{urg}(n_E, n_F^0) = \mathtt{urg}(n_E) \vee \mathtt{en}(n_E/\delta)$

A relação de transição $\rightarrow$ é definida como o menor conjunto que satisfaz:

$$\frac{n \xrightarrow{a,b,f}_j n' \ , \ a \neq \delta}{n \xrightarrow{a,b,f} n'}$$

$$\frac{n_E \xrightarrow{a,b,f}_E n'_E \ , \ \neg\mathtt{en}(n_E/\delta) \ , \ a \neq \delta}{(n_E, n_F^0) \xrightarrow{a,b,f} (n'_E, n_F^0)}$$

$$\frac{n_E \xrightarrow{\delta,b,f}_E n'_E}{(n_E, n_F^0) \xrightarrow{\delta,b,\mathcal{R}(T_F)} n_F^0}$$

Operador de Preempção: O grafo temporizado é: $G[\![E[> F]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com:

$N = N_E \cup N_F \cup (N_E \times \{n_F^0\})$

$T = T_E \cup T_F \cup \{t\} \quad (t \notin T_E)$

$n_0 = (n_E^0, n_F^0)$

Se $n \in N_j$, então $\mathtt{urg}(n) = \mathtt{urg}_j(n)$, onde $j \in \{E, F\}$

$\mathtt{urg}(n_E, n_F^0) = \mathtt{urg}(n_E) \vee \mathtt{urg}(n_F^0)$

A relação de transição $\rightarrow$ é definida como o menor conjunto que satisfaz as regras a seguir:

$$\frac{n \xrightarrow{a,b,f}_j n' \ , \ a \in Act^i \cup Act^*}{n \xrightarrow{a,b,f} n'}$$

$$\frac{n_E \xrightarrow{a,b,f}_E n'_E \ , \ a \in Act^i \cup Act^*}{(n_E, n_F^0) \xrightarrow{a,b,f} (n'_E, n_F^0)}$$

$$\frac{n_F^0 \xrightarrow{a,b,f}_F n_F \ , \ a \in Act^i \cup Act^*}{(n_E, n_F^0) \xrightarrow{a,b\wedge\mathtt{act}(n_E),\mathcal{R}(T_F)} n_F}$$

$$\frac{n_E \xrightarrow{\delta,b,f}_E n'_E}{(n_E, n_F^0) \xrightarrow{\delta,b,f} n'_E}$$

Preempção temporal: O grafo temporizado é: $G = [\![E <_L] \{\bigcup_j (a_j : Q_j)\}]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com:

$N = N_E \cup (\bigcup_j N_j) \cup (N_E \times N_1 \times \cdots N_k)$ onde $j \in \{1, \cdots, k\}$

$T = T_E \cup (\bigcup_j T_j) \cup \{t\}$

$n_0 = (n_E^0, n_1^0, \cdots, n_k^0)$

Se $n \in N_l$, então $\mathtt{urg}(n) = \mathtt{urg}_l(n)$, onde $l \in \{E\} \cup \{1, \cdots, k\}$

$\mathtt{urg}(n_E, n_1^0, \cdots, n_k^0) = \mathtt{urg}(n_E)$, onde $|L| = k$

Note-se que o grafo temporizado correspondente ao processo Q_j é chamado $G[\![Q_j]\!] = G_j = (N_j, T_j, n_j^0, \mathtt{urg}_j, \rightarrow_j)$ onde $j \in \{1, \cdots, k\}$.

A relação de transição $\rightarrow$ é definida como o menor conjunto que satisfaz:

$$\frac{n \xrightarrow{a,b,f}_l n' \ , \ a \neq \delta}{n \xrightarrow{a,b,f} n'}$$

$$\frac{n_E \xrightarrow{a,b,f}_E n'_E \ , \ a \notin Act^*}{(n_E, n_1^0, \cdots, n_k^0) \xrightarrow{a,b,f} (n'_E, n_1^0, \cdots, n_k^0)}$$

$$\frac{n_E \xrightarrow{a_j^*,b,f}_E n'_E}{(n_E, n_1^0, \cdots, n_k^0) \xrightarrow{a_j^*,b,f} n_j^0}$$

$$\frac{n_E \xrightarrow{\delta,b,f}_E n'_E}{(n_E, n_1^0, \cdots, n_k^0) \xrightarrow{\delta,b,f} n'_E}$$

Comentário: Na penúltima regra, o a_j^* denota a ocorrência de uma violação temporal em uma ação $a_j \in L$. Desta maneira, o comportamento que segue essa violação temporal é o abandono da execução do grafo $G[\![E]\!]$ e o início da execução do grafo $G[\![Q_j]\!]$ que está associado à ação a_j.

Instanciação de Processos: O operador de instanciação é definido a partir da função de troca de nomes de ações $\phi = [a_1/a'_1, \cdots, a_n/a'_n]$ em que para todo j, a ação a'_j pode tornar-se a_j. Para a expressão do comportamento de RT-LOTOS E, define-se o grafo temporizado da função de troca de nomes por $G[\![E\phi]\!] = (N, T, n_0, \mathtt{urg}, \rightarrow)$ com:

$N = N_E$

$T = T_E$

$n_0 = n_E^0$

$\mathtt{urg}(n) = \mathtt{urg}_E(n)$

A relação de transição $\rightarrow$ é definida como o menor conjunto que satisfaz:

$$\frac{n_E \xrightarrow{a',b,f}_E n'_E \ , \ \phi = [a_1/a'_1, \cdots, a_n/a'_n] \ , \ a/a' \in \phi \ , \ \{a, a'\} \subseteq Act \cup Act^*}{n \xrightarrow{a,b,f} n'}$$

$$\frac{n_E \xrightarrow{a,b,f}_E n'_E \ , \ a \notin \{a'_1, \cdots, a'_n\}}{n \xrightarrow{a,b,f} n'}$$

Se um processo $P[a'_1, \cdots, a'_n]$ é definido pela expressão de conportamento E, então o grafo temporizado da instanciação $P[a_1, \cdots, a_n]$ de P é exatamente o grafo temporizado

$G[\![E\phi]\!]$, que se escreve:

$$G[\![P[a_1, \cdots, a_n]]\!] = G[\![E\phi]\!]$$

II.4.3 Alguns exemplos ilustrativos

Login simplificado O primeiro exemplo dado para ilustrar a translação de RT-LOTOS para grafos temporizados é o do procedimento de *login* simplificado (de [Nicollin 92]).

Para iniciar o procedimento, o sistema envia um *prompt* de *login* p. Após este evento, o usuário tem l unidades de tempo para entrar com um nome válido v para terminar a tarefa e iniciar a fase sessão S. Quando o usuário fornece uma resposta inválida n, ou quando o tempo l expira, um novo procedimento de login é inicializado. Este comportamento pode ser representado em RT-LOTOS da seguinte maneira:

```
Login(p,n,v):= p; (v; S [] [0,l]n; Login(p,n,v)) <n] Login(p,n,v)
```

Na figura II.1 é mostrado o grafo temporizado do processo acima. Observe-se que pelo fato do processo `Login` ser seqüencial pôde-se construir o grafo com apenas um relógio. Note-se também que pelo fato do processo `Login` não envolver os operadores de ocultação e de composição paralela as funções de divergência e de urgência são estáticas.

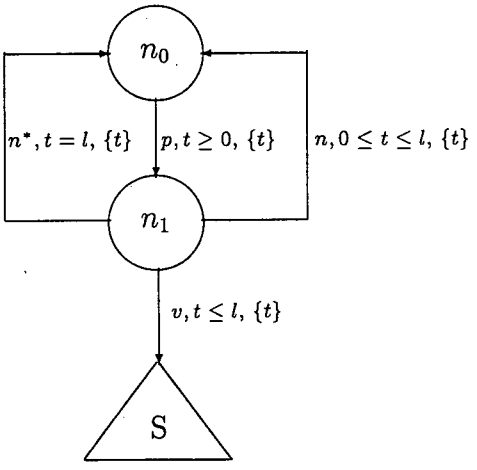


Figura II.1: Grafo temporizado de um procedimento de *login* simplificado

Login completo Para tornar mais realista o procedimento anterior, considere-se que o sistema limite a duração global do procedimento de login em um tempo máximo de d unidades de tempo e força uma exceção E a realizar-se após este tempo. A especificação em RT-LOTOS desta versão completa do procedimento *login*, chamada de `Complete_Login`, é apresentada a seguir. Utiliza-se nesta especificação o operador de preempção de RT-LOTOS juntamente com o processo `Login2`. O processo `Login2` é bastante semelhante ao processo `Login`, a única diferença está na maneira com que o processo S é acionado após a realização da ação v. Em `Login` (definido anteriormente) isto é realizado por uma prefixação simples, enquanto que em `Login2` utiliza-se o processo `exit` associado com o operador de composição

seqüencial. Login2 é definido assim para permitir que a realização da ação δ do processo exit possa "cancelar" o operador de preempção.

```
Complete_Login:= (Login2(p,n,v) [> [d]i; E) >> S
where
   process  Login2(p,n,v):= p; (v; exit [] [0,1]n; Login2(p,n,v))
                            <n] Login2(p,n,v)
   endproc
```

Na figura II.2 é mostrado o grafo temporizado do procedimento de login completo. Como o processo Complete_Login não é seqüencial, o grafo não pôde ser construído com um único relógio. Foram utilizados dois relógios para sua construção: t_1, com um limite superior igual a d, que limita a duração global do processo Complete_Login; e o relógio t do processo Login2. Como a função f é sempre a função de reinicialização, escreve-se o conjunto dos relógios a reinicializar no lugar da função.

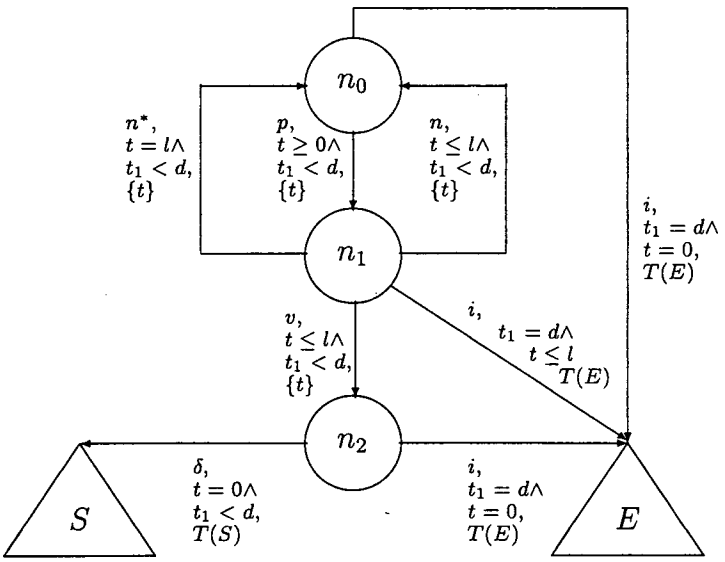


Figura II.2: Grafo temporizado de um procedimento de *login* completo

Fluxo de vídeo com dependência entre unidades de informação A especificação do fluxo de vídeo com dependência entre unidades de informação apresentada na seção II.3.2 serve bem como segundo exemplo de transformação de uma especificação RT-LOTOS para grafos temporizados. O processo "`video`" apresentado anteriormente pode ser transformado a partir das regras anteriores no grafo temporizado da figura II.3.

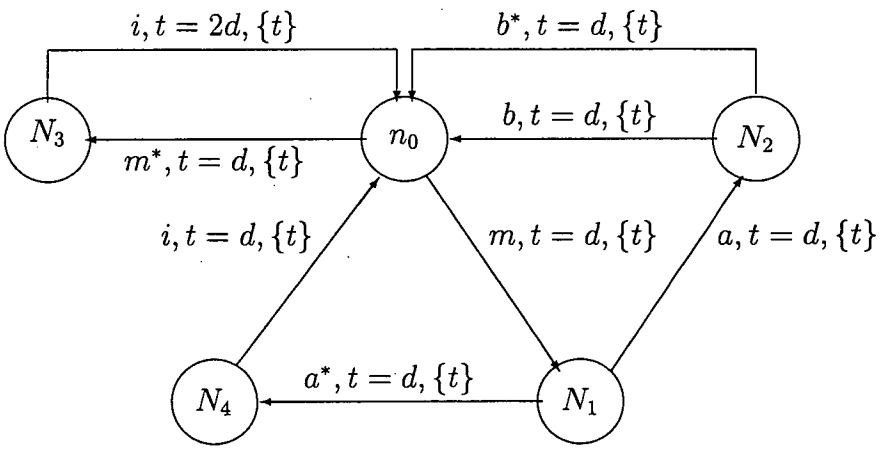


Figura II.3: Grafo temporizado do processo `video`

II.4.3.0.4 Um processo com dois relógios Apresenta-se por último um terceiro exemplo para ilustrar a situação em que existem dois relógios: um processo P resultante da composição paralela de dois processos elementares que se escrevem:

```
P := a;[0,3]b;E |[b]| c;[0,5]b;F
```

Os processos elementares são representados pelos grafos temporizados da figura II.4 ao passo que o resultado da sua composição aparece na figura II.5. Pode-se notar que tem-se dois relógios t_1 et t_2 correspondendo a cada processo elementar da composição paralela. Além disso, pode-se observar que a partir do nodo (n_1^1, n_1^2), a ação b é imediata a partir do instante em que as condições sobre os dois relógios sejam satisfeitas, pois neste instante a função $\mathtt{urg}(n_1^1, n_1^2)$ tem o valor tt; esta representação corresponde bem à semântica de RT-LOTOS.

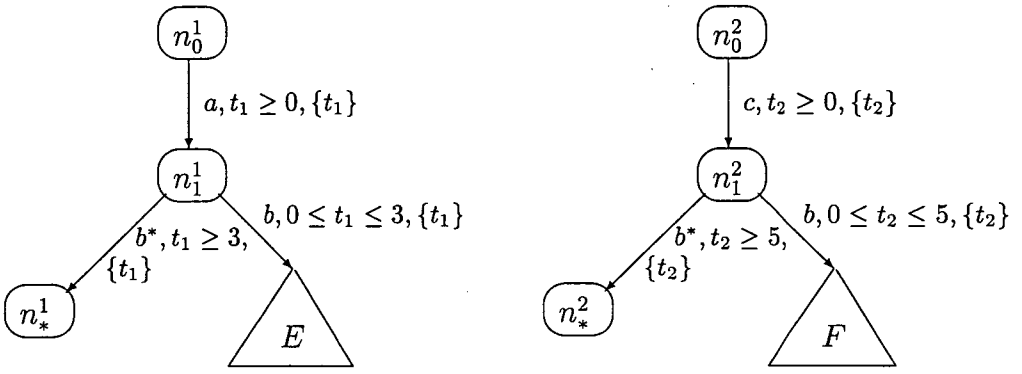


Figura II.4: Grafos temporizados dos processos elementares

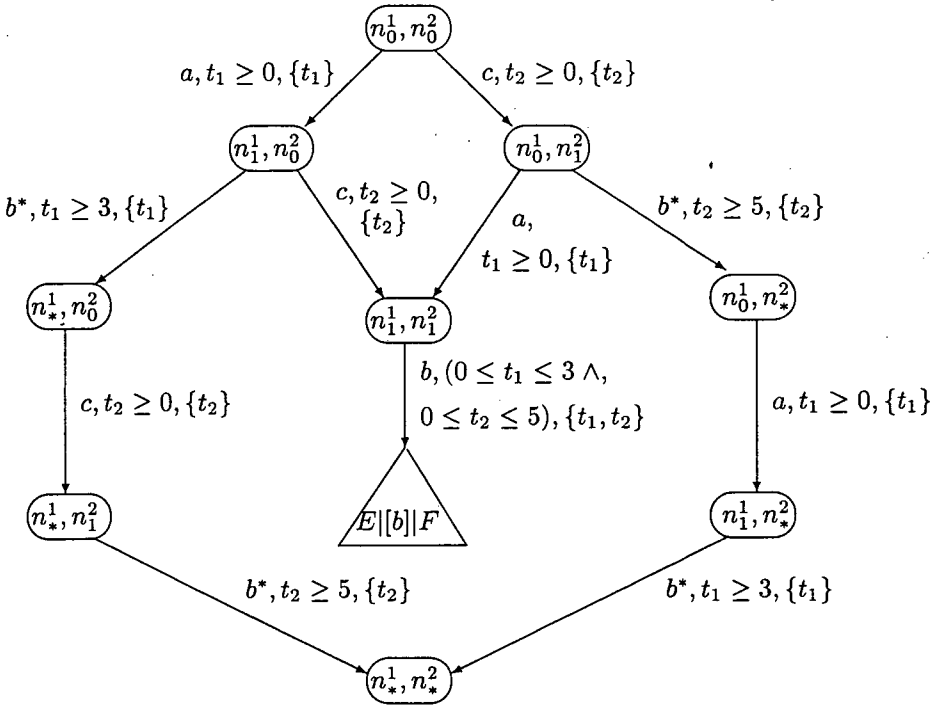


Figura II.5: Grafo temporizado do processo resultante

II.4.4 Por quê representar RT-LOTOS em Grafos Temporizados?

Dois principais fatores motivaram a redefinição da semântica de RT-LOTOS no formalismo grafo temporizado:

- representação alternativa da semântica formal de álgebras de processos temporizadas e
- verificação de sistemas dependentes do tempo.

Nas subseções que se seguem apresenta-se uma pequena discussão de cada um destes aspectos.

II.4.4.1 O ponto de vista representação semântica

Geralmente, sistemas de transições rotuladas são os formalismos básicos mais utilizados para representação da semântica formal de álgebras de processos temporizadas. Porém, os grafos temporizados vem sendo considerado como uma interessante alternativa para a descrição de sistemas tempo-real, visto que eles podem oferecer um aumento do potencial de provas de propriedades de álgebras de processos descritos semanticamente por estes; em [Godskesen 92] por exemplo, Godskesen e Larsen utilizam propriedades de grafos temporizados para provar a inexistência de teoremas de expansão em uma álgebra de processos descrita por este modelo.

II.4.4.2 O ponto de vista da verificação

A possibilidade de utilização de novas técnicas para verificação de sistemas tempo-real é, sem dúvida nenhuma, a maior motivação para utilização de grafos temporizados. Nesta seção, serão apresentados de maneira informal alguns importantes aspectos relacionados à verificação de sistemas tempo-real. Inicialmente, comenta-se as diferentes abordagens para verificação em sistemas que não dependem do tempo, e discute-se as dificuldades de tratamento do tempo nas abordagens convencionais usadas em álgebras de processos. Em seguida, são comentadas as potencialidades de algumas técnicas emergentes utilizadas para verificação de sistemas tempo-real e a sua relação com o formalismo grafos temporizados (e RT-LOTOS).

As técnicas de verificação são geralmente divididas em técnicas de verificação de equivalências e de verificação de modelos.

As técnicas de verificação fundadas em equivalências são baseadas no confronto de duas especificações do sistema a verificar: uma primeira descrevendo os detalhes de realização dos procedimentos necessários para realizar um objetivo; e uma segunda, mais simples, que descreve um comportamento mais abstrato do objetivo a realizar. Os Sistemas de Transições Rotuladas das duas especificações são então testadas para determinar se os comportamentos são equivalentes.

Verificação de modelos ("*model-checking*") é um método de verificação de sistemas concorrentes em que fórmulas de uma lógica devem ser provadas ou não sobre um modelo de

grafos de estados do comportamento do sistema. Nestas técnicas dois formalismos diferentes são usados em geral. Um primeiro é usado para descrever o sistema como formalismo de especificação (em geral usam-se máquinas de estados finitos, redes de Petri, ou autômatos). Deste formalismo são gerados grafos que representam os estados alcançáveis pelo sistema. O segundo formalismo é uma lógica (modal, em geral) que permite expressar asserções sobre propriedades do sistema que podem ser provadas ou refutadas (Por exemplo, em [Cavalli 87], Cavalli e Horn utilizam verificação de modelos de uma lógica temporal em máquinas de estados finitos como técnica de verificação).

A aplicação dessas técnicas num contexto onde o tempo intervém direta e explicitamente não é trivial. Como já foi comentado na seção II.2.2, a representação de sistemas dependentes do tempo em Sistemas de Transições Rotuladas implica em adicionar um número muito elevado de transições para denotar a progressão do tempo. Por exemplo, para representar um mecanismo de timeout onde o tempo de espera é de 2 unidades de tempo (ex.: `a;E[][2]i;F`), teríamos que ter três ramificações representando as possibilidades de realização de a nos instantes 0,1 e 2 no caso do domínio de tempo ser o dos números naturais; ou teríamos um número infinito de ramificações para representar as possibilidades de ocorrência de a transições de progressão do tempo caso o domínio de tempo fosse o dos números racionais positivos. Este exemplo mostra a possibilidade de crescimento elevada, com relação ao número de estados, que a introdução do tempo provoca nos Sistemas de Transições Rotuladas, sobretudo quando deseja-se utilizar um modelo de tempo denso.

A partir da discussão acima, pode-se concluir que, pelo menos com as técnicas atuais, técnicas de verificação baseadas em Sistemas de Transições Rotuladas não são, em geral, de grande utilidade para tratamento de sistemas dependentes do tempo. Em particular, apesar da sua importância na definição completa de uma Álgebra de Processo Temporizada as bissimulações com tempo que podem ser definidas para uma álgebra de processos temporizadas parecem difíceis de serem verificadas quando o domínio de tempo for denso.

Contudo, a problemática de verificação para tempo-real não é recente e outras técnicas de verificação para este tipo de sistemas são descritas na bibliografia. Em particular, as técnicas de verificação baseadas em grafos e autômatos temporizados descritas em [Alur 90] e [Alur 92] têm um importância diferenciada no contexto de RT-LOTOS; isto é, ambas as técnicas são fundadas em formalismos próximos e foi mostrado que é possível transladar álgebras de processos temporizadas para estes modelos [Nicollin 92b].

Pelas razões evocadas acima, é natural de ver como uma alternativa interessante a representação semântica de álgebras de processos temporizadas por grafos temporizados. Os trabalhos apresentados em [Nicollin 92b, Daws 94] vão nesta direção. Uma ferramenta chamada KRONOS desenvolvida pelo laboratório VERIMAG [6] já existe e permite realizar a verificação simbólica de modelos para sistemas tempo-real, a partir da utilização da lógica temporal tempo-real TCTL [Daws 94]. Pretende-se em trabalho futuros a partir da definição apresentada anteriormente construir um tradutor de especificações RT-LOTOS em grafos temporizados que possa ser adaptado à KRONOS e a ferramentas similares.

Concluindo, a associação de grafos temporizados e de álgebras de processos temporizadas é uma alternativa interessante pois ela se beneficia das facilidades de análise e prova do modelo subjacente (grafos temporizados) utilizando apenas uma extensão temporal (no nosso

[6] VERIMAG é um laboratório conjunto do CNRS, INPG, Université Joseph Fourier, e Verilog S/A associado ao IMAG, Grenoble, França.

caso RT-LOTOS) de uma linguagem de especificação padronizada LOTOS que contém todas as características necessárias para representar as restrições temporais que se encontram nos sistemas tempo-real.

II.5 Conclusão

Neste capítulo foi apresentada formalmente a extensão temporal de Basic LOTOS chamada RT-LOTOS. Definiu-se RT-LOTOS como uma álgebra de processos; isto é, como uma quádrupla composta de um conjunto de operadores, um conjunto de nomes de ações, um conjunto de regras da semântica operacional transicional e uma equivalência (congruência) observacional sobre modelos. Apresentou-se também um conjuto de propriedades que RT-LOTOS satisfaz, e através de pequenos exemplos verificou-se a capacidade do formalismo em expressar os principais requisitos e mecanismos temporais encontrados em sistemas tempo-real, multimídia e protocolos de comunicação. Finalmente, apresentou-se uma representação alternativa da semântica de RT-LOTOS a partir dos grafos temporizados e discutiu-se sua utilização para a verificação.

No próximo capítulo indica-se como o formalismo básico RT-LOTOS pode ser estendido para incorporar a passagem de valores e todos os outros recursos presentes na linguagem LOTOS Completa [ISO 88].

Capítulo III

A Linguagem RT-LOTOS$^+$ Completa

Introdução

Este capítulo tem dois objetivos maiores: estender o formalismo básico RT-LOTOS para incorporar a passagem de valores temporais e os dados; e apresentar alguns exemplos de especificação com a linguagem completa assim definida.

Inicialmente, apresenta-se os conceitos básicos de um operador especial que permite o registro do tempo de espera de uma ação até a sua realização, isto é, o tempo medido desde o oferecimento de uma ação até a sua execução. Em seguida, este operador é incluído na semântica formal de RT-LOTOS, resultando assim, no formalismo chamado RT-LOTOS$^+$. Em seguida, a atenção é centrada para a incorporação dos dados em RT-LOTOS, mais precisamente, da representação Act One em RT-LOTOS$^+$. Tendo em vista a complexidade deste objetivo, a discussão sobre a introdução de dados é encaminhada da seguinte forma: inicialmente, apresenta-se um estudo sobre a introdução de passagem de valores simples em RT-LOTOS para verificar o alto grau de independência entre os problemas de introdução de dados e introdução do tempo em LOTOS; em seguida, discute-se resumidamente o problema global de acoplamento de Act One a RT-LOTOS; e finalmente, apresenta-se a sintaxe da linguagem RT-LOTOS$^+$ Completa. Concluindo este capítulo, apresenta-se alguns exemplos de especificação de aplicações dependentes do tempo utilizando RT-LOTOS$^+$ Completa.

III.1 De $\lambda x.f$ para $ax;E$ e $ax@t;E$

Nesta seção é introduzido no formalismo RT-LOTOS o operador $ax@t;E$ que permite registrar o tempo que a ação a esperou até a sua ocorrência e passá-lo para o comportamento pós-fixado desta ação a. Este operador é necessário para poder representar situações onde o comportamento futuro depende de alguma condição temporal estabelecida sobre o instante de realização de uma ação ocorrida anteriormente. A introdução deste operador combinado com a passagem de valores, utilização de dados e de todos os operadores do LOTOS completo dará ao modelo resultante um elevado poder de expressão. Esta seção é organizada da seguinte maneira: primeiro, apresenta-se a intuição na qual é baseado o operador em questão; e depois, apresenta-se a forma como esse operador é introduzido formalmente na semântica de RT-LOTOS.

III.1.1 $a@t;E$: Gravando o tempo de espera até a ocorrência da ação a

No modelo *Timed CCS*, descrito em [Yi 91] Wang Yi introduz a interessante construção "$a@t.E$" que permite registrar em uma variável (no caso t) no processo E o tempo que a ação a teve que esperar desde seu oferecimento até sua realização. Desta maneira, é possível em *Timed CCS* utilizar a variável t para parametrizar outros construtores do processo E, isto é, o comportamento que segue à ocorrência de ação a. Nosso objetivo nesta seção é redefinir o atributo "$@t$" no contexto de RT-LOTOS.

Para bem entender a base da construção "$a@t.E$" do *Timed CCS* vejamos como Wang Yi o define [Yi91]:

> "Recorde-se do significado de um programa escrito através de λ-termos, $\lambda x.f$. O programa espera uma entrada e quando o dado de entrada v é apresentado, ele computa o valor de $f\{v/x\}$. Isto motivou a construção $\alpha x.E$ do CCS com passagem de valores. Similarmente, o agente $\alpha x.E$ está esperando por uma mensagem no canal α e quando uma mensagem v está disponível em α, ele deve comportar-se como $E\{v/x\}$. Note que o comportamento seguinte a $\alpha x.E$ depende do conteúdo v da mensagem recebida, mas não do tempo no qual v está disponível. Como enfatizado anteriormente, o comportamento futuro de um sistema tempo-real depende do tempo no qual um estímulo externo tal como uma mensagem chega. Assim, a construção $\alpha x.E$ não é muito bem adequada para modelar Sistemas Tempo-Real. Para satisfazer este requisito, é introduzida uma variável extra t que grava o tempo de espera até que uma mensagem esteja disponível em α. Usa-se $\alpha x@t.E$ para representar um agente que está esperando por uma mensagem no canal α. Quando uma mensagem chega ele deve tornar-se $E\{v/x, d/t\}$ onde
>
> - v é o conteúdo da mensagem e
> - d é o tempo de espera até que a mensagem estivesse disponível.
>
> Note que a informação de temporização sobre uma mensagem que chega também é considerada como parte da entrada. A idéia é estendida para a construção $\bar{\alpha}v.E$ de CCS e utiliza-se $\bar{\alpha}v@t.E$ para representar um agente que está apto para entregar uma mensagem no canal α. Quando o canal α estiver habilitado, ele deve entregar v e tornar-se $E\{d/t\}$ onde
>
> - d é o tempo de espera até que a mensagem v seja entregue.
>
> Quando o conteúdo de uma mensagem recebida não afeta o comportamento futuro do agente, ele pode ser ignorado. Isso é a sincronização pura do CCS. Utiliza-se
>
> $$\alpha@t.E$$
>
> para denotar um agente que está esperando pelo seu ambiente para sincronizar em α e então tornar-se $E[d/t]$ assim fazendo-o, onde d é o tempo de espera até que a sincronização ocorra."

Concluindo a citação de Yi mostraremos a sua descrição de um cronômetro digital (utiliza-se a notação $\alpha!v.E$ ao invés de $\bar{\alpha}v.E$).

$$
\begin{aligned}
S &\stackrel{def}{=} start.G \\
G &\stackrel{def}{=} halt@t.D(t) \\
D(t) &\stackrel{def}{=} display!t.S
\end{aligned}
$$

O cronômetro está sempre pronto para aceitar um sinal *start* e então espera por um sinal *halt*. Quando o sinal *halt* chega, o tempo de espera entre o *start* e o *halt* é exibido.

III.1.2 RT-LOTOS$^+$: Incorporando '$a@t; E$' em RT-LOTOS

O atributo $@t$ é incorporado às prefixações em RT-LOTOS de maneira optativa possibilitando sua utilização apenas em casos onde ele for útil. Para isto, é necessário estender a sintaxe e a semântica formal de RT-LOTOS, definindo assim o que chamar-se-á de RT-LOTOS$^+$.

A sintaxe de RT-LOTOS$^+$ é a mesma de RT-LOTOS, definida na seção II.1.2.1, exceto pelo enriquecimento desta por um novo tipo de prefixação $[t_1, t_2]a@t; E$.

Para definição formal na semântica desta nova forma de prefixação precisa-se acrescentar às regras da semântica operacional de RT-LOTOS para a prefixação as quatro regras seguintes:

1.
$$\frac{-}{[0,d]a@t; E \xrightarrow{a} E\{0/t\}} \quad (d, t \in D^\omega) \quad (a \in Act^i)$$

2.
$$\frac{-}{[0,0]a@t; E \xrightarrow{a^*} stop} \quad (a \in Act)$$

3.
$$\frac{-}{[0,d+s]a@t; E \xrightarrow{s} [0,d]a; E\{t+s/t\}} \quad (d, s, t \in D^\omega \text{ e } s > 0) \quad (a \in Act^i)$$

4.
$$\frac{-}{[d_1+s, d_2+s]a; E \xrightarrow{s} [d_1, d_2]a; E\{t+s/t\}} \quad (s, d_1, d_2, t \in D^\omega \text{ e } s > 0) \quad (a \in Act^i)$$

As regras 1 e 2 definem o comportamento deste tipo de prefixação no caso da evolução por ocorrência de uma ação a pertencente a $Act^i \cup Act^*$ e as regras 3 e 4 definem o comportamento devido à progressão do tempo.

A especificação do cronômetro digital citado anteriormente e que utiliza este novo operador num contexto onde RT-LOTOS$^+$ seja definido com passagem de valores é dada por:

```
Process Cronometro:=
     start;
     halt@t;
     display!t;
     Cronometro
endproc
```

No estado em que se encontra, a prefixação do tipo $[t_1, t_2]a@t; E$ ainda não atinge a sua melhor utilização para especificação de processos pois ainda falta definir a passagem e utilização de valores em RT-LOTOS$^+$; isto é tratado na próxima seção.

III.2 Os dados em RT-LOTOS

Esta seção trata da introdução dos dados em RT-LOTOS$^+$. O objetivo maior aqui é obter uma técnica de descrição formal completa como em LOTOS, isto é, que acopla a técnica de descrição de dados Act One com RT-LOTOS$^+$. Porém, o acoplamento formal de Act One a RT-LOTOS$^+$ é extenso e complexo. Todavia, este acoplamento não difere substancialmente do que é feito formalmente para LOTOS na sua definição [ISO 88]. Desta forma, ao invés de definirmos um tal acoplamento (o que foge um pouco do objetivo principal deste trabalho - o tempo), preferiu-se estender RT-LOTOS com passagem de valores simples, e então verificar informalmente que os dados representam uma dimensão ortogonal ao comportamento temporal. E desta forma, pretende-se evitar o tratamento formal da introdução dos dados a RT-LOTOS. A definição formal da sintaxe do RT-LOTOS$^+$ Completo encerra esta seção.

III.2.1 Introduzindo passagem de valores simples em RT-LOTOS$^+$

Esta seção é dedicada a discussão de como redefinir formalmente a semântica de RT-LOTOS$^+$ de maneira a obter um modelo com passagem de valores simples, ou seja, valores tomados de um único tipo de dados. Neste ponto, uma pergunta cabível é: porque não se considerar o acoplamento de Act One ao invés de um modelo tão simples de dados? A resposta é que através da introdução de dados simples pode-se evidenciar (sem perda de generalidade) a partir da partir da construção do formalismo assim estendido que a problemática de introdução do tempo é essencialmente ortogonal com a problemática de introdução de dados em LOTOS, ou seja, são problemas não correlatos e independentes. Isto implica então que o problema de acoplar Act One a RT-LOTOS é muito próximo daquele do acoplamento desta linguagem de dados a LOTOS (exceto em algumas situações onde o dado tratado for o próprio tempo). Desta forma, tem-se fortes argumentos para não tratar formalmente neste trabalho o acoplamento de Act One a RT-LOTOS, visto que apesar deste ser bastante complexo e extenso, este é já um problema bem resolvido [ISO 88].

Conseqüentemente, ao invés de apresentarmos uma redefinição completa da semântica de RT-LOTOS, optamos por uma discussão informal de como aplicar, em RT-LOTOS, o método proposto por Robin Milner [Milner 80, Milner 89] para a redefinição da álgebra de processos CCS num contexto de passagem de valores. Esta seção é uma adaptação para o contexto de RT-LOTOS das seções correspondentes nas referências citadas anteriormente.

Para simplificar a discussão, assumiremos que todos os valores são tomados de um conjunto de valores fixado V. Porém, utilizaremos nos exemplos valores tomados dos vários tipos de dados definidos (e definíveis) em LOTOS completo. O acoplamento da linguagem de definição de tipos de dados de LOTOS ao seu modelo de comportamento está completamente definida na norma [ISO 88] que define completamente o modelo LOTOS padrão.

Um exemplo de RT-LOTOS com passagem de valores

Para simplificar a discussão, iniciaremos mostrando em pequenos exemplos de RT-LOTOS com passagem de valores como estes podem ser reduzidos ao cálculo básico. Considere o exemplo abaixo que corresponde à definição de uma célula de buffer com capacidade de armazenar um único elemento o qual é definida nas duas expressões de comportamento seguintes:

$$\begin{aligned} C &\stackrel{def}{=} in?x; C'(x) \\ C'(x) &\stackrel{def}{=} out!x; C \end{aligned}$$

A constante C' é transladada em uma família de constantes C'_v, uma para cada valor de $v \in V$. Analogamente, o prefixo de recepção parametrizado '$out!x$;' é transladada em uma família de prefixos de recepção 'out_v;', um para cada valor de $v \in V$. E assim, a definição da equação para C' torna-se uma família de equações de definição:

$$C'_v \stackrel{def}{=} out_v; C \qquad (v \in V)$$

Ortogonalmente, o prefixo '$in?x$;' é transladado em uma escolha múltipla da forma

$$in_{v_1}[]in_{v_2}[]\cdots[]in_{v_i}[]\ldots$$

para $v_k \in V$, que denotaremos por $[]_{v\in V} in_v$, para refletir o fato de que ele pode aceitar qualquer valor de V como entrada, pois ele liga a variável x. Assim, o uso de uma variável ligada x em '$in?x$;' é substituído por uma escolha múltipla que oferece todas as possibilidades de x no conjunto V. A definição de C torna-se:

$$C \stackrel{def}{=} []_{v\in V} in_v$$

Com a introdução de variáveis e valores nas expressões de comportamento de RT-LOTOS, pode-se acrescentar facilmente ao modelo estruturas condicionais de maneira a possibilitar execuções condicionais de uma expressão de comportamento dependendo dos valores que variáveis assumem dentro das expressões. Em LOTOS [ISO 88], usa-se as expressões guardadas para este fim. Um exemplo de tais expressões guardadas é dado a seguir. Neste exemplo, o processo P representa a recepção de um valor $v \in V$ e em seguida ativa o processo parametrizado $Q(x)$ que por sua vez ativa o comportamento E se o valor entrado for diferente do valor $c \in V$.

$$\begin{aligned} P &\stackrel{def}{=} input?x; Q(x) \\ Q(x) &\stackrel{def}{=} [x \neq c] -> E \end{aligned}$$

Para transladar estes processos para o cálculo básico é necessário redefinir o processo P como uma escolha múltipla, um termo da escolha para cada elemento $v \in V$, como o processo C do exemplo anterior. Por sua vez a translação do processo Q pode ser vista como uma definição condicional, por partes. Isto é, para cada valor $v \in V$ define-se um processo Q_v como:

$$Q_v \stackrel{def}{=} \begin{cases} E & \text{se } v = c \\ stop & \text{caso contrário} \end{cases}$$

Assim, expressões *booleanas* internas a uma expressão de comportamento tornam-se condições externas que descrevem membros de uma família indexada, neste caso a família que define as equações.

Translação das expressões de comportamento de RT-LOTOS$^+$

Passa-se a apresentar a translação como um todo. Inicialmente, recorde-se que $\mathcal{E}$ representa o conjunto das expressões de comportamento geradas pelo modelo RT-LOTOS básico. Define-se agora o conjunto $\mathcal{E}^+$ das expressões de comportamento do cálculo completo. Assume-se que para cada nome de expressão de comportamento A é atribuído um número inteiro, a *aridade* de A, representando o número de parâmetros que lhe são atribuídos. Assume-se também a existência de expressões de valor **e** e expressões booleanas **b**, construídas a partir de valores de variáveis $x, y, \ldots$ junto com valores constantes v e todos os símbolos de operadores convenientes. Assim, $\mathcal{E}^+$ é o menor conjunto de todos os nomes de expressões de comportamento parametrizadas do cálculo $A(e_1, \cdots, e_n)$ (com $n < \infty$) e também as seguintes expressões, onde E, E', E_i, E_i' estão em $\mathcal{E}^+$:

1. *stop* - Inação
2. *exit* - Término com sucesso
3. $[t_1, t_2]a?x; E$, $[t_1, t_2]a!\mathbf{e}; E$, $[t_1, t_2]i; E$, $[t_1, t_2]a?x@t; E$, $[t_1, t_2]a!\mathbf{e}@t; E$, $[t_1, t_2]i@t; E$ - Prefixações ($a \in Act$)
4. $E[\,]E'$ - Escolha
5. $E|[L]|E'$ - Composição Paralela ($\delta \in L \subseteq Act$)
6. *hide* L *in* E, Ocultação ($L \subseteq Act$)
7. $E >> E'$ - Composição Sequencial
8. $E [> E'$ - Preempção
9. $E <L] \{a_1{:}E_1', \cdots, a_n{:}E_n'\}$ - Preempção Temporal ($L \subseteq Act$)
10. $P[a_1, \cdots, a_n]$ - Instanciação de Processos
11. [**b**] $->$ E - Expressão guardada

Além disso, cada nome de expressão de comportamento A com aridade n tem uma equação de definição

$$A(x_1, \cdots, x_n) \stackrel{def}{=} E$$

onde o lado direito E não contém valores de variáveis livres exceto os $x_1, \cdots, x_n$ (que devem ser distintos).

Na translação de $\mathcal{E}^+$ para $\mathcal{E}$, às situações onde expressões de comportamento não tenham variáveis de valores livres $x, y, \ldots$. Se $E \in \mathcal{E}^+$ contém a variável x livre, então ela pode ser considerada como uma família de expressões $E\{v/x\}$, uma para cada valor constante v. A translação de $\mathcal{E}^+$ para $\mathcal{E}$ baseia-se na idéia de que cada nome de ação no cálculo completo corresponda a uma família $\{l_v : v \in V\}$ de nomes de ações do cálculo básico. Assim, pensa-se em uma única ação com nome l como um conjunto de ações de nome l_v, uma para cada valor $v \in V$.

A translação será dada recursivamente na estrutura das expressões de comportamento. Para fazer isso, será necessário considerar apenas expressões de valor **e** e expressões booleanas **b** que não contém variáveis. Isto é devido a translação de uma Prefixação de entrada $[t_1, t_2]a?x; E$ que substitui x por uma escolha múltipla sobre todos os valores $v \in V$, eliminando assim todas as ocorrências da variável de ligação x. Além disso, cada expressão de valor **e** será considerada como idêntica ao valor v de sua avaliação, e cada expressão booleana **b** como idêntica a um dos valores booleanos $\{tt, ff\}$.

Para cada expressão $F \in \mathcal{E}^+$ sem variáveis de valor livre, sua forma transladada $\hat{F} \in \mathcal{E}$ é dada como segue:

E	$\hat{E}$
$[t_1, t_2]a?x; E$	$[]_{v \in V} a_v; \widehat{E\{v/x\}}$
$[t_1, t_2]a!\mathbf{e}; E$	$a_{\mathbf{e}}; \hat{E}$
$[t_1, t_2]a?x@t; E$	$[]_{v \in V} a_v@t; \widehat{E\{v/x\}}$
$[t_1, t_2]a!\mathbf{e}@t; E$	$a_{\mathbf{e}}@t; \hat{E}$
$[t_1, t_2]i; E$	$i; \hat{E}$
$E[\,]E'$	$\hat{E}[\,]\widehat{E'}$
$E\|[L]\|E'$	$\hat{E}\|[\{l_v : l \in L, v \in V\}]\|\widehat{E'}$
$hide\ L\ in\ E$	$hide\ \{l_v : l \in L, v \in V\}\ in\ \hat{E}$
$E >> F$	$\hat{E} >> \hat{F}$
$E [> F$	$\hat{E} [> \hat{F}$
$E <L]\ \{l_1{:}Q_1, \cdots, l_n{:}Q_n\}$	$\hat{E} <\{l_{i_v} : l_i \in L, v \in V\}]\ \{l_{1_v}{:}\widehat{Q_1}, \cdots, l_{n_v}{:}\hat{Q_n}\}$
$P[l_1, \cdots, l_n]$	$P[\widehat{l_{1_v}, \cdots, l_{n_v}}]$
$[\mathbf{b}] -> E$	$\begin{cases} E & \text{se } \mathbf{b} = tt \\ stop & \text{caso contrário} \end{cases}$
$A(e_1, \cdots, e_n)$	$A_{e_1, \cdots, e_n}$

Além disso, a simples equação de definição de uma constante $A(\tilde{x}) \stackrel{def}{=} E$, onde $\tilde{x} = x_1, \cdots, x_n$, é transladada em um conjunto indexado de equações de definição

$$\{A_{\tilde{v}} \stackrel{def}{=} E\{\widehat{\tilde{v}/\tilde{x}}\} : \tilde{v} \in V^n\}$$

Como um exemplo, suponha-se que V é o conjunto dos números naturais $\mathbf{N}$, e considere a expressão de comportamento

$$F \equiv in?x; ([x < 5] -> out!x; stop)$$

Então a translação é

$$\hat{F} \equiv [\,]_{v \in V} E_v$$

onde as expressões $E_0, E_1, \cdots$ (uma para cada número natural) são dadas esquematicamente por

$$E_v \equiv \begin{cases} out_v; stop & \text{se } v < 5 \\ stop & \text{caso contrário} \end{cases}$$

Comentários: Embora longe de estar completa, a translação apresentada esquematiza com exatidão o procedimento básico a se realizar para redefinir formalmente o modelo RT-LOTOS$^+$ com passagem de valores (de um único tipo) como extensão estrita de RT-LOTOS. Restaria ainda definir formalmente alguns outros aspectos existentes em LOTOS Completo tais como Escolhas Generalizadas, Predicados de Seleção, *exit* com passagem de valores e funcionalidades, a cláusula *accept*, etc. Porém, como pode ser verificado durante toda esta seção, não houve dificuldades maiores no estabelecimento da translação de RT-LOTOS$^+$ com passagem de valores para o modelo RT-LOTOS$^+$ Básico. Observe-se que em nenhum instante, a presença do tempo ou dos operadores temporais apresentados provocou dificuldades para a translação (muito provavelmente, a única ressalva que se faria a esta afirmação seria com relação ao operador de escolha generalizada de LOTOS Completo, que exigiria um tratamento mais elaborado, mas também sem ocasionar um problema grave). Desta maneira, constatou-se que o acoplamento do tratamento de dados a RT-LOTOS (e também a outras extensões temporais de LOTOS) é um problema que não é agravado pelo tratamento do tempo no modelo.

III.2.2 RT-LOTOS$^+$ e Act One

O acoplamento da linguagem de tipos abstratos de dados Act One a RT-LOTOS, bem como a definição completa de seus operadores complementares é complexa e extensa; porém, ele não apresenta maiores dificuldades técnicas, e é muito similar ao que é feito na norma de definição de LOTOS [ISO 88] e em outros modelos com tempo, como por exemplo os descritos em [Leduc 94] e [Bolognesi 93]. Em resumo, o acoplamento de Act One a LOTOS (e também em extensões temporais deste) apesar de complexa é um problema já resolvido e completamente descrito na sua norma de definição [ISO 88], sua adequação a ambientes temporizados encontram-se também bem descritas em [Leduc 94] e [Bolognesi 93]. Desta maneira, esta questão não será abordada neste trabalho.

Na seqüência, definiremos a sintaxe abstrata da linguagem de especificação formal RT-LOTOS$^+$ Completa, e descreveremos informalmente os operadores desta linguagem que não estão presentes em RT-LOTOS.

III.2.3 A sintaxe de RT-LOTOS$^+$ Completa

Os termos da sintaxe de RT-LOTOS$^+$ Completa são gerados pela seguinte BNF:

$E ::=$ *stop* (* inação *)
| *exit* | *exit*($\mathrm{e}_1, \cdots, \mathrm{e}_n$) (* terminação com sucesso *)

	$g;E \mid g?x{:}s;E \mid g!\mathbf{e};E \mid i;E \mid [t_1,t_2]i@t;E$	(* prefixação *)
	$E[\,]E'$	(* escolha *)
	$E\|[L]\|E' \mid E\|\|\|E' \mid E\|\|E'$	(* composição paralela *)
	hide L *in* E	(* ocultação *)
	$E >> E' \mid E >>$ *accept* $x_1{:}s_1,\cdots,x_n{:}s_n\,E'$	(* composição seqüencial *)
	$[\mathbf{b}] -> E$	(* guarda *)
	$E\,[> E'$	(* preempção *)
	$E<L]\,\{a_1{:}E'_1,\cdots,a_n{:}E'_n\}$	(* preempção temporal *)
	$P[a_1,\cdots,a_n](\mathbf{e}_1,\cdots,\mathbf{e}_m)$	(* instanciação de processos *)
	let $x_1{:}s_1 = \mathbf{e}_1,\cdots,x_n{:}s_n = \mathbf{e}_n$ *in* E	(* definição local *)
	choice a *in* $[a_1,\cdots,a_n][\,]E \mid$ *choice* $x{:}s[\,]E$	(* escolha generalizada *)

Onde as seguintes convenções foram adotadas:

- os $\mathbf{e}, \mathbf{e}_1, \cdots, \mathbf{e}_n, \mathbf{e}_m$ denotam expressões de valor,
- o $\mathbf{b}$ denota uma expressão booleana,
- os $a, a_1, \cdots, a_n$ denotam ações (identificadores de portas),
- L é um conjunto de ações da forma $L = \{a_1, \cdots, a_n\}$,
- os g's denotam abreviaturas de ações dos seguintes tipos: $a(\equiv [0,\omega]a)$, ou '$[t_1,t_2]a$', ou '$a@t$', ou '$[t_1,t_2]a@t$',
- os $s, s_1, \cdots, s_n$ denotam identificadores de tipos (*sorts*), e
- os $x, x_1, \cdots, x_n$ denotam identificadores de valores.

A seguir apresenta-se uma curta descrição informal sobre os operadores de RT-LOTOS$^+$ Completa que ainda não foram tratados anteriormente.

- $E >>$ *accept* $x_1 : s_1, \cdots, x_n : s_n\,E'$: A composição seqüencial possibilita a passagem de valores do processo E para o processo E'. Os valores a serem passados de E para E' são descritos num operador *exit* com uma lista de parâmetros como mostrado em seguida.
- $exit(\mathbf{e}_1, \cdots, \mathbf{e}_n)$: A terminação com sucesso com uma lista de expressões é que indica que valores devem ser passados para um outro processo numa composição seqüencial.
- $[\mathbf{b}] -> E$: Qualquer expressão pode ser precedida de um predicado chamado guarda. A interpretação é a seguinte: se o predicado é verdadeiro então o comportamento descrito por E é possível, e se o predicado é falso, então a expressão como um todo é equivalente a um *stop*.
- *let* $x_1 : s_1 = \mathbf{e}_1, \cdots, x_n : s_n = \mathbf{e}_n$ *in* E: Como ressaltado pela sintaxe, este oper-dor permite realizar atribuições de valores à variáveis dentro de uma expressão de comportamento E.
- *choice* a *in* $[a_1, \cdots, a_n][\,]E$ ou *choice* $x : s[\,]E$: Este operador de escolha permite generalizar o operador de escolha ordinário. No primeiro, caso a escolha é realizada sobre um conjunto de ações, enquanto no segundo, a escolha dá-se sobre o domínio de valores especificado por s.

III.3 Representações de aplicações dependentes do tempo em RT-LOTOS$^+$ Completa

Esta seção é dedicada a representação de algumas aplicações dependentes do tempo na linguagem RT-LOTOS$^+$ Completa e também a uma comparação entre a formas de representação de alguns mecanismos temporais clássicos nesta linguagem e na linguagem ET-LOTOS [Leduc 94] (descrita na seção I.3.3) proposta para padronização da extensão temporal de LOTOS. Esta comparação se faz necessária devido à alguns pontos em comum existentes nessas duas linguagens. Em todos os exemplos desta seção se utilizará uma sintaxe simplificada, omitindo-se os "process" e os "endproc" no inicio e fim dos processos.

Objetivando isenção na comparação a ser realizada a seguir, optou-se pela adoção dos mesmos exemplos usados em [Leduc 93] como exemplos base para representação em RT-LOTOS$^+$ Completa. Assim, utilizar-se-ão os exemplos mais significativos descritos naquela referência para evidenciar as características a serem comparadas.

III.3.1 Representações de operações temporais de base e comparação entre RT-LOTOS e ET-LOTOS

Exemplo 1: Temporização A temporização (correspondente ao exemplo da figura 1 em [Leduc 93]) é um mecanismo temporal simples utilizado freqüentemente em protocolos de comunicação. Numa situação simplificada analisa-se por exemplo o comportamento de um emissor e constata-se a seguinte situação: após a emissão de uma mensagem, ele fica disposto a esperar uma confirmação durante um certo tempo, ou a reenviar sua mensagem após este mesmo tempo.

Em RT-LOTOS, tem-se duas maneiras alternativas para especificar este mesmo comportamento (ambas mais simples que a equivalente em ET-LOTOS que deve usar um operador de limitação de oferta na ação `Ack`). A primeira delas usa uma composição do operador de escolha com um retardo de `waiting-time` unidades de tempo na ação interna `i` como abaixo:

```
SAFE:= Req?s:SDU; SEND(s)
SEND(s):= Send!s; (Ack; SAFE
                   []
                   [waiting-time]i;SEND(s))
```

E a segunda é baseada no operador de preempção temporal `<Ack]`.

```
SAFE:= Req?s:SDU; SEND(s)
SEND(s):= Send!s; ([0,waiting-time]Ack; SAFE
                   <Ack]
                   SEND(s))
```

Exemplo 2: Escolha com temporizações diferentes Neste exemplo (correspondente ao da figura 2 em [Leduc 93]) deseja-se descrever um processo, chamado de `SP`, que pos-

sibilite a realização de uma escolha entre dois processos diferentes ACCEPT-EXP-DATA, e ACCEPT-DATA, sendo que são impostos retardos diferentes antes das ações iniciais de cada um deles; ou seja, um retardo small_delay para o primeiro processo e um retardo high_delay para o segundo. Assim, as ações iniciais do processo ACCEPT-EXP-DATA serão oferecidas (high_delay − small_delay) unidades de tempo antes que as do processo ACCEPT-DATA. A escolha entre a realização de um ou de outro processo só deve ser resolvida no instante de realização de uma primeira ação de um ou de outro processo. Após a execução de qualquer um dos dois processos, o comportamento que deve seguir é o relançamento do processo SP.

RT-LOTOS não possui um operador de imposição de retardos sobre processos como ET-LOTOS, e devido a este fato o comportamento de SP descrito a seguir, em RT-LOTOS, é mais complexo que o seu correspondente em ET-LOTOS apresentado em [Leduc 93].

```
SP:= [small_delay]i; ACCEPT-EXP-DATA >> SP
     []
     [high_delay - small_delay]i; (ACCEPT-DATA >> SP
          []
          ACCEPT-EXP-DATA >> SP)
```

Exemplo 3: Isocronismo com tratamento de exceção em caso de faltas O isocronismo é um mecanismo que exige que os eventos ocorram em instantes de tempo precisos. Neste exemplo (correspondente ao exemplo da figura 4 em [Leduc 93]), deseja-se especificar um processo que só aceita as primitivas DataReq em instantes precisos, espaçados regularmente no tempo. Além disso, deseja-se também incorporar um mecanismo de recuperação em caso de impossibilidade de realização da ação oferecida DataReq.

Em RT-LOTOS pode-se especificar mecanismos de recuperação com muita simplicidade. O comportamento descrito acima pode ser especificado como:

```
ISOCHRONOUS:= [period]i;([0]DataReq;ISOCHRONOUS
                         []
                         ISOCHRONOUS)
```

ou como:

```
ISOCHRONOUS:= [0]DataReq;ISOCHRONOUS
              |||
              [period]i;ISOCHRONOUS
```

ou ainda usando o operador de preempção temporal, que parece o mais natural para um usuário expressar uma exceção, e que é mais legível.

```
ISOCHRONOUS:= [period]DataReq;ISOCHRONOUS
              <DataReq]
              ISOCHRONOUS
```

Deve-se destacar a naturalidade desta última forma de representação que mostra a facilidade do uso de RT-LOTOS.

Exemplo 4: Retardo não determinista ou aleatório Neste exemplo (correspondente ao exemplo 5 em [Leduc 93]) objetiva-se especificar em um processo um retardo aleatório dentro de um intervalo $[min, max]$.

```
TRANSM:=DataReq?s:SDU; [min,max]i; DataInd!s; stop
```

Como no exemplo 1, a forma de representar a situação acima é mais simples em RT-LOTOS pois não necessita usar operadores de atraso e um limitador de oferta como em ET-LOTOS.

Exemplo 5: "Watchdog": Escolheu-se para ilustrar este mecanismo o procedimento de entrada em uma sessão (*login*). O exemplo a seguir descreve então uma rotina básica de entrada em uma sessão tal qual descrito na figura 6 em [Leduc 93]. O sistema solicita do usuário a introdução do seu nome e senha em menos que `logtime` unidades de tempo. Em caso de ultrapassagem do tempo ou de senha incorreta, o sistema é reiniciado, senão o sistema aceita a solicitação. O usuário pode realizar várias tentativas de entrada em sessão, mais o tempo total destas tentativas não deve ultrapassar a `maxtime`. Observe-se que desta forma, existem dois *watchdogs* parametrizados por `logtime` e `maxtime`. Este processo pode ser especificado em RT-LOTOS como:

```
SESSION :=
(LOGIN-PROCEDURE [> [maxtime]i;stop) >> SESSION-PHASE
where
LOGIN-PROCEDURE :=
PHASE1 >> accept b:bool in
            ([b] -> exit) [] ([not(b)] -> LOGIN-PROCEDURE)
where
PHASE1 :=
prompt?log:login; prompt?pwd:password; exit(correct(log,pwd))
[> [logtime]i; PHASE1)
```

As representações são similares nos dois formalismos, sendo que no caso de ET-LOTOS utiliza-se o operador de atraso e uma ação interna enquanto RT-LOTOS usa somente a ação interna temporizada para representar os *watchdogs*.

Exemplo 6: Temporização simétrica Este exemplo (correspondente ao exemplo 13 em [Leduc 93]) consiste em sincronizar dois processos onde cada um deles:

- executa tarefas próprias de duração desconhecida (representadas respectivamente pelas ações `d1` e `d2`);

- propõe ao outro de se sincronizar (via a ação sync a partir do instante em que isso seja possível (isto é, desde que o outro esteja pronto);

- não está disposto a esperar pelo outro processo mais do que um certo tempo (respectivamente to1 e to2).

A versão RT-LOTOS deste comportamento é dada por:

```
SYMETRICAL_TIMEOUT[d1,d2](to1,to2:time):=
PROCESS[d1,sync](to1) |[sync]| PROCESS[d2,sync](to2)
where
PROCESS[d,sync](t:time):= d; [0,t]sync; PROCESS[d,sync](t)
```

A especificação acima é mais geral que a sua correspondente em ET-LOTOS, visto que este formalismo requer na sua semântica de sincronização que as ações sincronizáveis sejam ocultadas pelo operaror *hide* para se obter a urgência na realização desta.

Diferenças básicas entre RT-LOTOS e ET-LOTOS: Concluindo esta seção, se apresentará uma sinopse das principais diferenças entre as linguagens de especificação formal RT-LOTOS e ET-LOTOS [Leduc 93] do ponto de vista da representação de elementos temporais de base.

A primeira diferença básica entre essas duas linguagens é que em RT-LOTOS a própria ação é temporizada e possue um intervalo temporal de possibilidade da sua realização que lhe é diretamente associado, enquanto em ET-LOTOS existem dois conceitos alternativos para representação da temporização: um operador de retardo Δ^t que prefixa retardos a processos em t unidades de tempo (por exemplo, a expressão $\Delta^t a; E$ representa um processo que inicialmente é retardado por t unidades de tempo e, após isto, transforma-se no processo $a; E$), e um parâmetro d que é pós-fixado às ações representando o tempo que a ação fica disponível para ocorrer. Outra característica de ET-LOTOS é que a realização de uma operação Δ^t é invisível e não decide a escolha entre processos.

A segunda diferença básica entre RT-LOTOS e ET-LOTOS é que na primeira linguagem as operações de *hide* e de sincronização de ações são urgentes, isto é, têm prioridade em relação às ocorrências de ações de progressão do tempo (ocorrem tão logo sejam possíveis). Em ET-LOTOS somente as operações com *hide* são urgentes. Desta maneira, para se modelizar em ET-LOTOS um processo com ações a sincronizar (como por exemplo o processo `a;E |[a]| b;a;F`) é necessário aplicar um *hide* antes da operação de sincronização propriamente dita (ex.: `hide a in (a;E |[a]| b;a;F)`) o que consideramos uma limitação substancial de ET-LOTOS, visto que isto faz com que toda sincronização fique obrigatoriamente confinada no interior do *hide*. Em RT-LOTOS não existe esta limitação à sincronização e a situação anterior pode ser modelada como o processo `a;E |[a]| b;a;F`.

A terceira e mais importante diferença entre essas duas linguagens reside no fato de RT-LOTOS possuir um operador de tratamento de exceções temporais que simplifica enormemente a representação de situações de recuperação de faltas ocasionadas por violações de restrições temporais impostas para uma aplicação. Por outro lado, ET-LOTOS não possui nenhum operador que tenha uma funcionalidade parecida.

Em virtude dessas três diferenças básicas pode-se afirmar que RT-LOTOS permite representar as operações temporais de base de forma mais simples e mais natural que ET-LOTOS; os exemplos de representações destas em RT-LOTOS anteriormente apresentados e sua comparação com as representações em ET-LOTOS mostradas em [Leduc 93] corroboram esta afirmação. A seguir destacam-se algumas situações que ilustram nos exemplos anteriores a comparação feita.

No exemplo 1, a representação em RT-LOTOS exige que apenas um lado da operação de escolha seja temporizado com o tempo "`waiting-time`", enquanto na representação em ET-LOTOS é necessário a temporização em ambos os lados da operação de escolha, o que nos parece um pouco contra-intuitivo. Ainda no exemplo 1, uma alternativa de representação da temporização em RT-LOTOS utiliza o operador de tratamento de violações temporais, o que torna essa representação bastante elegante e compreensível.

No exemplo 4, para se representar um retardo aleatório, ET-LOTOS requer a combinação do operador Δ^t com a ação interna i. Enquanto em RT-LOTOS apenas a ação interna i temporizada é requerida. Esta mesma situação é encontrada para a representação do watchdog no exemplo 5.

O exemplo 6 mostra a fragilidade do operador de composição paralela e sincronização de ET-LOTOS. Para representar esse exemplo nesta linguagem é necessário a utilização do operador *hide* para tornar urgente a sincronização. Em contrapartida, esta necessidade não existe em RT-LOTOS.

Entretanto, o exemplo 2 exibe a única situação onde a representação do processo em ET-LOTOS é mais simples que a representação em RT-LOTOS. Contudo, a situação exemplificada nos parece bastante singular e de pouca aplicabilidade prática.

III.3.2 Exemplos de representação de construções temporais para aplicações multimídia

Nesta subseção são descritos alguns exemplos de especificações características de aplicações multimídias; estes exemplos são similares àqueles apresentados em [Leduc 93].

Isocronismo - Variante Retoma-se mais uma vez o mecanismo de isocronismo do exemplo 3 (seção III.3.1, página 93) mas com uma variação: substitui-se o oferecimento pontual de uma ação `a` por um oferecimento por `d` (`d < period`) unidades de tempo, e impõem-se de compatibilizar os oferecimentos bem sucedidos e os não realizados a fim de poder adaptar o período. A função `new` não detalhada aqui realiza o cálculo do novo período.

Como em RT-LOTOS a expiração de um intervalo de tempo atribuído a uma ação (violação temporal da ação) não decide uma escolha, então, o comportamento descrito acima pode ser representado neste formalismo como:

```
behaviour ISOCHRONOUS(0,0,period,0) where
ISOCHRONOUS(sucessful,unsucessful,period,t1):=
([period-t1, period-t1+d]a@t; ISOCHRONOUS(sucessful+1,unsucessful,
                    new(period,sucessful,unsucessful),t)
          []
          ISOCHRONOUS(sucessful,unsucessful+1,
                    new(period,sucessful,unsucessful),0))
```

A representação deste comportamento em RT-LOTOS não difere substancialmente da correspondente em ET-LOTOS; a diferença básica é que na prefixação da ação a utiliza-se aqui um intervalo temporal para representar a restrição de tempo imposta a esta ação, enquanto que na representação em ET-LOTOS utiliza-se o operador de retardo Δ e mais o redutor de vida de ações para esta restrição de tempo. E em vista disso, a especificação em RT-LOTOS parece mais simples.

Controle de "Jitter" O controle de jitter sobre o retardo de transmissão é um requisito bastante típico dos serviços fornecidos em aplicações multimídia. Estas aplicações não podem tolerar variações muito grandes sobre esses retardos. Abaixo é descrito em RT-LOTOS um processo encarregado de controlar o jitter e de iniciar a desconexão no caso de um desvio muito elevado. Supõe-se que este processo controlador receba as informações sobre cada retardo de trânsito (medido por um outro processo) no ponto de interação `control` e que ele inicie a desconexão por `DiscInd`. Mesmo não contendo nenhuma restrição temporal este processo ilustra bem este tipo de problema.

```
JITTER-CONTROL:=
Control?t:time; JITTER-CONTROL2 (t,t)
where
JITTER-CONTROL2(tmin,tmax:time):=
   Control?t:time;
         ([max(t,tmax)-min(t,tmin) ≤ maxjit] ->
                JITTER-CONTROL2(min(t,tmin),max(t,tmax))
         []
          [max(t,tmax)-min(t,tmin) > maxjit] -> DiscInd; stop)
```

A representação deste processo é análoga à sua correspondente em ET-LOTOS.

Sincronização entre um emissor e um temporizador Uma outra situação comum a representar em aplicações multimídia é a sincronização entre uma entidade emissora e um temporizador. O comportamento que se deseja modelizar é o seguinte: tem-se um processo `Timer` que realiza a função de temporização e que deve interagir com um processo emissor `Sender` em três pontos de interação: `set` (para armar a temporização), `reset` (para parar a temporização) e `timeout` para indicar o fim da temporização. A representação de tal comportamento em RT-LOTOS pode ser escrita como:

```
SYSTEM:=
hide set, reset, timeout in (SENDER
                             |[set,reset,timeout]|
                             TIMER)
where
SENDER := req?s:sdu; SENDER2(s)
SENDER2(s:sdu) :=
send!s; set!waiting-time; (ack; reset; SENDER
                           []
                           timeout; SENDER2(s)
TIMER:= set?t:time; (reset; TIMER
                     []
                     [t]i; timeout; TIMER
```

A diferença básica entre esta especificação e a sua correspondente em ET-LOTOS é que aqui utiliza-se a temporização da ação i no processo TIMER enquanto que em ET-LOTOS utiliza-se o operador de retardo Δ para este fim, o que implica no mesmo nível de complexidade para a expressão desse comportamento.

Ressincronização de componentes de um fluxo multimídia O exemplo a seguir trata da ressincronização de um fluxo multimídia (voz e imagem). O comportamento desejado é composto de dois processos paralelos onde cada um gera uma componente do fluxo (a voz ou a imagem). Cada processo é suposto receber um fluxo de *frames* quase-isócrono (isto é, isocrono a menos de um jitter) e reenviar este fluxo de maneira completamente isócrona. Para realizar isso, introduz-se um retardo igual ao jitter máximo. De qualquer maneira, se o jitter do fluxo de entrada é tal que um frame se faz esperar por muito tempo, então o processo não poderá sincronizá-lo e sinaliza um erro.

- Uma versão RT-LOTOS da especificação desse comportamento pode ser escrita como:

```
SYNC-CONTROL[soundrec,videorec,out1,out2,errosignal]:=
   ((CONTROL[soundrec,out1](sound,speriod,sjitter) [> exit(video))
    |||
   (CONTROL[videorec,out2](video,vperiod,vjitter) [> exit(sound))
   ) >> accept err:ersource in errorsignal!err; stop
where
   CONTROL[r,s](ers:ersource, per, jit:time):=
        [0,jit]r@t; [jit-t]s; [per-jit]i; CONTROL[r,s](ers,per,jit)
        []
        [jit]i; exit(ers)
```

As diferenças básicas entre esta especificação e a sua correspondente em ET-LOTOS é que na prefixação das ações com restrição de tempo utiliza-se, na especificação em RT-LOTOS, intervalos temporais representando estas restrições, enquanto que na representação em ET-LOTOS utiliza-se o operador Δ e mais o redutor de vida de ações nestas situações.

As diferenças básicas entre esta especificação e a sua correspondente em ET-LOTOS é

que na prefixação das ações com restrição de tempo utiliza-se, na especificação neste formalismo, o operador Δ e mais o redutor de vida de ações representando estas restrições, enquanto que na representação em RT-LOTOS utiliza-se intervalos temporais nestas situações, o que é mais simples.

Concluíndo esta seção, pode-se afirmar, em vista as especificações apresentadas nos parágrafos anteriores, que RT-LOTOS$^+$ é também uma linguagem bem adaptada para especificação de aplicações multimídia.

III.4 A representação de alguns problemas clássicos em RT-LOTOS$^+$ Completa

Nesta seção são apresentados três exemplos de sistemas dependentes do tempo representados em RT-LOTOS: um escalonador *time-slice* [Yi 91], o algoritmo de sincronização de lábios [Regan 93] e o protocolo do bit alternante [Garavel 89]. Escolheu-se exemplos simples das três áreas de aplicações visadas neste trabalho: sistemas tempo-real, sistemas multimídia e protocolos de comunicação.

III.4.1 Um escalonador *time-slice*

Nesta seção apresenta-se a especificação em RT-LOTOS de um escalonador *time-slice* no qual o tempo intervém. O exemplo tratado aqui é uma adaptação daquele apresentado em [Yi 91] o qual, por sua vez, é uma variante do escalonador apresentado em [Milner 89].

A especificação informal do escalonador é dada a seguir. Um número fixo de processos compartilham o uso de uma CPU. Para cada processo é atribuído um intervalo de tempo chamado seu *quantum*, durante o qual é permitido que ele utilize a CPU. Se o processo não conclui sua execução (*job*) no fim do seu quantum, a CPU é tomada dele e passada para o próximo processo. Se o processo termina sua execução antes do término do seu quantum, a CPU é passada diretamente para o processo seguinte.

Detalhando o algoritmo, assuma-se que n processos P_i compartilham uma CPU e que o quantum para todos os processos é igual a `quantum`. O escalonador aloca a CPU para os processos em uma ordem cíclica iniciando com P_1 e então $P_2 \ldots$ etc. Cada processo P_i obtém a CPU pela realização de uma ação $\texttt{begin}_i$. Quando P_i obtém a CPU, é permitido que ele se execute por $\texttt{quantum} - t$ unidades de tempo, onde t é o retardo entre o instante de tempo quando a CPU está disponível para P_i e o instante de tempo quando a solicitação é feita por P_i. Se no final deste intervalo de tempo o processo continua em execução, então a CPU lhe é retirada pela realização da ação $\texttt{susp}_i$ e dada ao próximo processo P_{i+1}[1]. Então o processo P_i é suspenso para esperar até o próximo *time slice*. Se P_i concluiu sua execução antes do intervalo de tempo expirar, então ele deve informar isto ao escalonador pela realização da ação $\texttt{end}_i$, e assim, a CPU é comutada diretamente para P_{i+1}.

Quando P_i não está sendo executado, ele fica em dois estados possíveis: *passivo* ou *suspenso*. Se P_i iniciou uma execução então ele é suspenso; no outro caso ele é passivo. Se um processo passivo não faz uma solicitação da CPU, pela realização da ação $\texttt{begin}_i$,

[1] Assume-se a que a soma dos índices é calculada módulo n.

em tmax (tmax < quantum) unidades de tempo, então a CPU é comutada para o processos P_{i+1}. Um processo suspenso P_i pode ser despertado pela realização de uma ação $\texttt{assig}_i$ e permitido rodar por quantum unidades de tempo.

Na especificação RT-LOTOS apresentada a seguir assume-se que existem n processos a serem escalonados. nproc $\leq n$ denota um número índice do processo. nproc é incrementado a cada vez que o escalonador passa a CPU para outro processo através de uma adição módulo n. Outras notações utilizadas são:

- quantum: denota a quantidade de tempo (*time slice*) que os processos podem ocupar a CPU quando são escalonados.

- tmax: denota o tempo máximo que é aguardado por um processo nproc passivo para solicitar a CPU (realização de uma ação begin!nproc, se isto não ocorrer até este tempo a CPU é dada ao processo nproc+1.

- suspended: denota o conjunto dos processos que estão suspensos pelo escalonador. suspended não é definido formalmente na especificação a seguir, mas assume-se que ele seja um type ACT ONE com uma estrutura de conjunto, munido das operações de inclusão +, de exclusão -, e de um predicado booleano de pertinência "in".

- Os significados das ações usadas na especificação do escalonador são:

  1. begin!nproc: o processo de número nproc inicia sua execução e adquire a CPU.

  2. end!nproc: o processo de número nproc completa sua execução corretamente e libera a CPU.

  3. susp!nproc: é tomada a CPU do processo de número nproc e também este processo é suspenso.

  4. assig!nproc: o processo de número nproc é despertado e a CPU é atribuída para ele.

A especificação do escalonador é dada a seguir. Observe-se que para funcionar corretamente o escalonador deve ser inicializado com valores de parâmetros nproc = 1 e suspended = $\emptyset$ e conseqüentemente Scheduler(1, $\emptyset$).

```
process Scheduler(nproc:integer, suspended:set):  noexit :=
        [nproc in suspended] → WaitForReq(nproc,suspended)
       []
        [not(nproc in suspended)] → WakeUp(nproc,suspended)
where
      process WaitForReq(nproc:integer, suspended:set):  noexit :=
            begin@t!nproc; Run(nproc, quantum-t, suspended+nproc)
           []
            [tmax]i; Scheduler(nproc+1 MOD n, suspended)
      where
          process Run(nproc:integer, time_slice:time, suspended:set):  noexit :=
                end!nproc; Scheduler(nproc+1 MOD n,suspended-npro)
               []
                [time_slice]i; Suspend(nproc,suspended)
          where
                process Suspend(nproc:integer, suspended:set):  noexit :=
                      susp!nproc; Scheduler(nproc+1 MOD n, suspended) :=
                endproc
          endproc
      endproc
      process WakeUp(nproc:integer, suspended:set):  noexit :=
            assig!nproc; Run(nproc, quantum, suspended) :=
      endproc
endproc
```

Observe-se que dois operadores temporais de RT-LOTOS foram cruciais para o estabelecimento da especificação: a prefixação temporal e a prefixação temporal estendida com $@t$ que permite o registro do tempo de espera até a ocorrência de uma ação, no caso a ação begin.

III.4.2 Algoritmo de sincronização de lábios

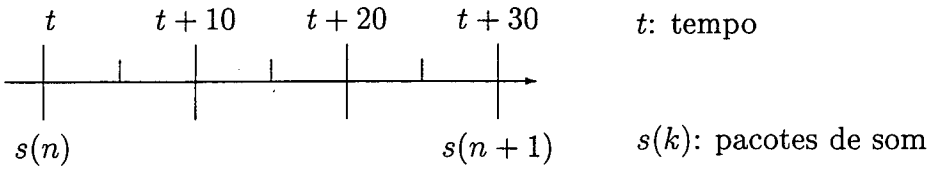
Esta seção é dedicada a apresentação de um exemplo clássico na área de aplicações multimídia: o algoritmo de sincronização de lábios. Este algoritmo tornou-se uma referência como exemplo de aplicação multimídia devido a sua descrição em [Regan 93] e [Blair 93] onde os autores apresentam especificações deste algoritmo em extensões temporais de LOTOS. Os requisitos que são apresentados abaixo para este algoritmo foram sumariados destas referências. A granularidade do tempo usada neste exemplo será o mili-segundo.

III.4.2.1 Requisitos funcionais do algoritmo

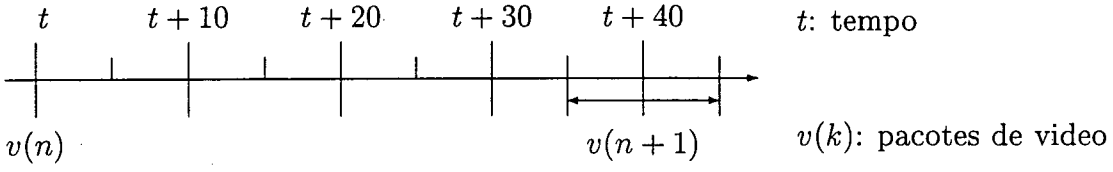
Os requisitos funcionais do algoritmo de sincronização de lábios são os seguintes [Blair93, Regan 93]:

- **Som:** A trilha sonora consiste de som digital com amostragem de 12MHz. Isto é dividido em entradas onde cada uma delas contém 400 amostras de maneira que um

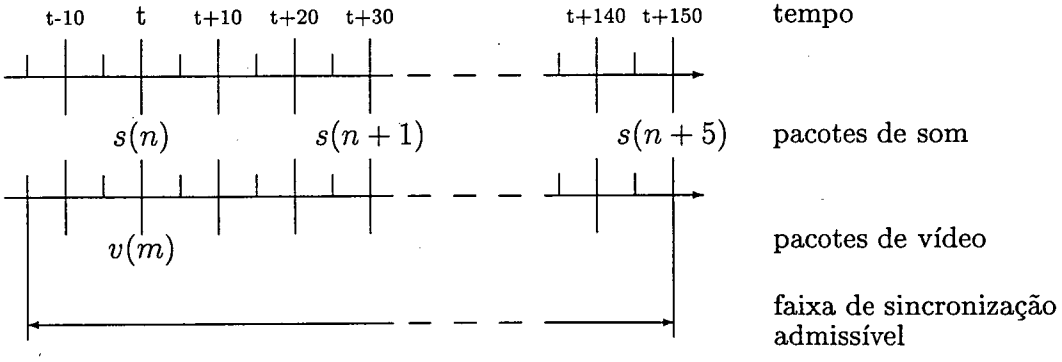
pacote deve ser apresentado a cada 30ms. Não admite-se jitter no fluxo sonoro, sendo que imprescindivelmente uma entidade sonora deverá ser apresentada a cada 30ms.



- **Imagem:** O fluxo do vídeo tem uma taxa de 25 quadros por segundo ou um quadro a cada 40 ms. Diferente da preparação do fluxo de som, os dados de vídeo envolvem maior processamento (descompactação, por exemplo) e o tempo que isto toma não pode ser garantido especificamente. Ou seja, pode existir um jitter, podendo ser neste caso igual a ±5ms, isto é o tempo mínimo após a apresentação de um quadro de vídeo e antes da apresentação do próximo quadro deve ser de 35ms, e o tempo máximo após a apresentação de um quadro de vídeo e antes da apresentação do próximo quadro deve ser de 45ms.



- **Sincronização do som e da imagem:** Os requisitos impostos para que o sincronismo do som com a imagem seja considerado satisfatório são:
  - O quadro de vídeo pode atrasar-se com relação ao som associado em até 150ms.
  - O quadro de vídeo pode preceder o seu som associado em até metade do incremento do som, isto é 15ms.



onde $s(n)$ é o pacote de som associado ao pacote de vídeo $v(m)$.

III.4.2.2 A estrutura da especificação do algoritmo de sincronização de lábios

A figura III.1 mostra a estrutura geral da especificação, `LIP_SYNC`, em RT-LOTOS do algoritmo de sincronização de lábios. Esta figura mostra duas fontes de dados, cada uma das quais

comunica-se com seu próprio gerenciador quando um item de dados está disponível (através de s_avail, v_avail). Quando estão prontos, os gerenciadores sinalizam ao controlador geral (CONTROLLER) (através de s_ready, v_ready) e esperam por um sinal de ok (ok_s, ok_v) antes de solicitar ao dispositivo de apresentação que apresente o dado (s_present, v_present). Finalmente, o dispositivo de apresentação (monitor, display) responde com um reconhecimento (s_presented, v_presented) quando o item de dado foi apresentado com sucesso.

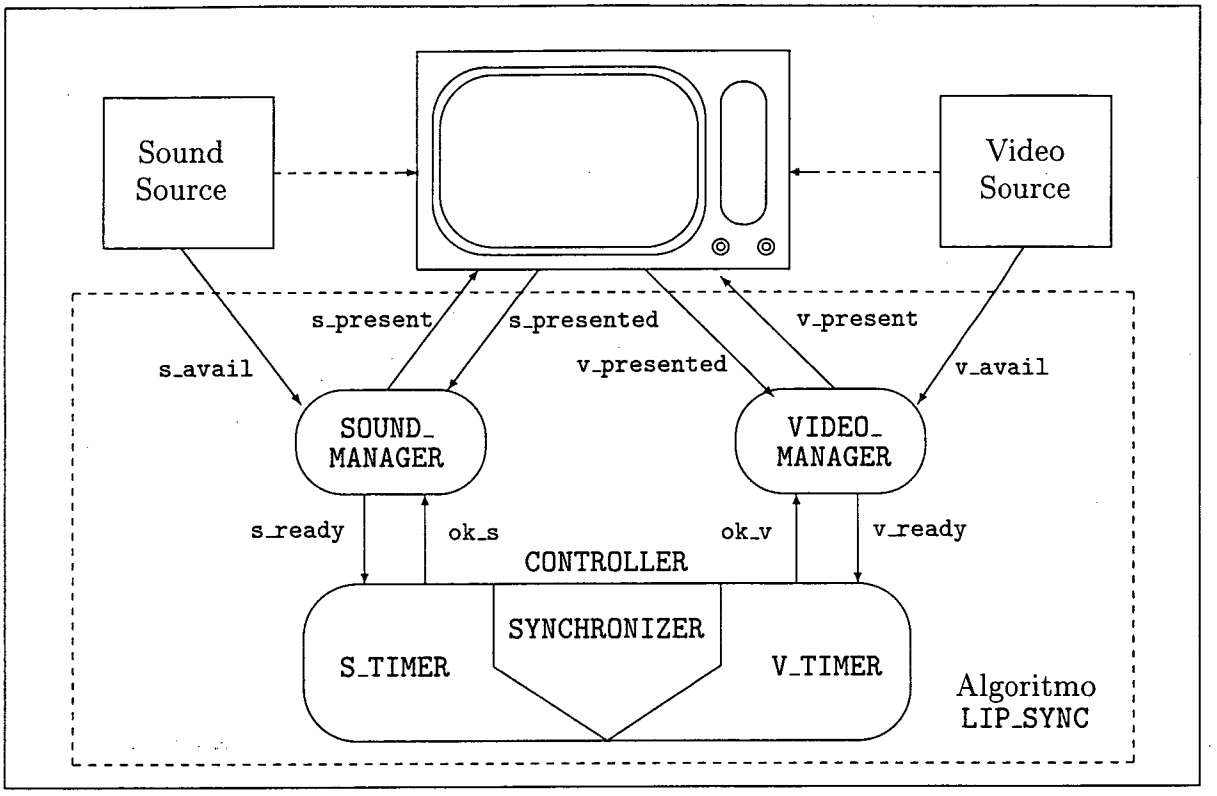


Figura III.1: Estrutura geral do algoritmo de sincronização de lábios

Neste sistema, é o controlador (CONTROLLER) que tem a tarefa de manter a sincronização de lábios entre o som e a imagem.

III.4.2.3 Especificação do algoritmo LIP_SYNC em RT-LOTOS

Esta subseção é dedicada a apresentação de uma especificação em RT-LOTOS do algoritmo de sincronização de lábios. A especificação aqui apresentada é uma adaptação para RT-LOTOS da versão correspondente deste algoritmo descrito em Temporal LOTOS apresentada originalmente em [Regan 93] (e resumido na seção I.3.3). Da mesma maneira que na adaptação para LOTOS deste algoritmo apresentada em [Blair 93], preserva-se aqui toda estrutura original do algoritmo de maneira a permitir uma eventual comparação direta dessas três representações.

Com o objetivo de tornar as especificações mais legíveis, se omitirá a especificação das listas das portas usadas nas definições dos processos quando elas forem evidentes.

A especificação do algoritmo LIP_SYNC em RT-LOTOS de acordo com a estrutura da

figura III.1 é mostrada a seguir.

```
process LIP_SYNC:=
          hide s_ready,v_ready, ok_s,ok_v in
          ((SOUND_MANAGER ||| VIDEO_MANAGER)
             |[s_ready,v_ready,ok_s,ok_v]|
          CONTROLLER
endproc
```

Os processos SOUND_MANAGER, VIDEO_MANAGER e CONTROLLER serão definidos nas próximas subseções.

III.4.2.3.1 Definições de SOUND_MANAGER e VIDEO_MANAGER

Inicia-se pela definição de um gerenciador de fluxo genérico que será instanciado para se obter os gerenciadores de fluxo do som (SOUND_MANAGER) e da imagem (VIDEO_MANAGER).

```
process MANAGER[avail,ready,present,presented,ok] noexit:=
          avail;ready;ok;
          MANAGER_LOOP[avail,ready,present,presented,ok]
endproc

process MANAGER_LOOP[avail,ready,present,presented,ok] noexit:=
          present;(avail;exit ||| presented;exit) >>
          ready;ok;
          MANAGER_LOOP[avail,ready,present,presented,ok]
endproc
```

O gerenciador é inicialmente desencadeado pela disponibilidade de um item de dado avail. Ele então informa ao controlador que está pronto (ready). Quando ele recebe o *OK* do controlador ele entra num ciclo infinito (*loop*) no qual a primeira ação é solicitar ao dispositivo de apresentação que apresente a próxima entidade. Ele então espera por uma confirmação do dispositivo de apresentação indicando que a entidade foi apresentada e um sinal da fonte indicando que um outro item de dado está disponível. Tendo realizado isto com sucesso o gerenciador sinaliza ao controlador que uma outra entidade está pronta, e espera por um *OK* antes de reiniciar o *loop*. Os gerenciadores do som e do vídeo são definidos como:

```
process SOUND_MANAGER noexit:=
          MANAGER[s_avail,s_ready,s_present,s_presented,s_ok]
endproc

process VIDEO_MANAGER noexit:=
          MANAGER[v_avail,v_ready,v_present,v_presented,v_ok]
endproc
```

Como os processos acima não possuem qualquer dependência temporal, eles são similares àqueles apresentadas em Temporal LOTOS [Regan 93].

III.4.2.3.2 Definição do CONTROLLER

O controlador é construído a partir de dois temporizadores (S_TIMER para o som, e V_TIMER para a imagem), de um sincronizador (SYNCHRONIZER), e de um relógio interno comum aos processos. A especificação do processo CONTROLLER em RT-LOTOS é dada por:

```
process CONTROLLER:=
          (S_TIMER ||| V_TIMER)
             |[ok_s,ok_v]|
          SYNCHRONIZER
endproc
```

Cada temporizador assegura que os limites dos jitters em cada fluxo não sejam violados. O relógio serve de referência de tempo. O sincronizador assegura que o fluxo de vídeo mantenha-se sincronizado com o fluxo de som através da verificação de que

- se uma entidade de som chega ela é seguida por um quadro de vídeo associado em não mais que 150ms;
- se um quadro de vídeo chega ele é seguido por sua entidade de som associada em não mais que 15ms
- o que significa que cada quadro de vídeo é apresentado em não mais que 15ms antes de sua posição na trilha do som e não mais que 150ms depois.

Definição dos temporizadores

Um temporizador genérico pode ser definido como:

```
process TIMER[trigger,error](not_before, not_after:time,mssg:an_error):=
          [not_before, not_after]trigger;
          TIMER[trigger,error](not_before, not_after,mssg)
        <trigger]
          (error!mssg:an_error;exit)
endproc
```

A partir do processo TIMER, os temporizadores do som e do vídeo são definidos por instanciação como:

```
process S_TIMER:=
          ok_s; TIMER[ok_s,error](30,30,"soundlate")
endproc

process V_TIMER:=
          ok_v; TIMER[ok_v,error](35,45,"videolate")
endproc
```

Observe-se que na especificação do processo TIMER a utilização dos operadores prefixação

temporizada e preempção temporal de RT-LOTOS tornaram a especificação daquele comportamento bastante simples e compreensível. Comparando com a especificação em Temporal LOTOS [Regan 93] tem-se deste lado um ganho expressivo.

Definição do sincronizador

A definição do sincronizador é dada a seguir. Antes porém, outros processos auxiliares são definidos. Observe-se que devido ao fato da apresentação do som estar ancorada, isto é não é permitido jitter neste fluxo, pode-se medir a apresentação dos quadros de vídeo a partir do tempo de início da apresentação do som. No caso das entidades de som pararem de chegar o processo S_TIMER tratará a situação emitindo a mensagem de erro "soundlate". Esse tempo (a duração da apresentação do som) será acessado utilizando-se o processo SOUND_QUERY.

```
process SOUND_CLOCK noexit:=
          sound_clock;SOUND_QUERY(0)
endproc

process SOUND_QUERY(s_time:time):=
          sound_query@t!(s_time+t);SOUND_QUERY(s_time+t)
endproc
```

Observe-se que o operador de prefixação estendido com $@t$ de RT-LOTOS foi usado na especificação do SOUND_QUERY. Este processo supervisiona o tempo da apresentação do som e o fornece quando solicitado.

Sincronizador de vídeo: O sincronizador de vídeo é iniciado quando o sinal v_ready é emitido pelo gerenciador de vídeo. Ele então atualiza o tempo para apresentação do próximo quadro de vídeo. Este tempo pode então ser comparado com o tempo presente a apresentação do som que deverá apresentar uma das três saídas.

- Se o quadro do vídeo está além do som por mais que o jitter permitido (ou seja, o tempo do vídeo está mais que 150ms após o tempo presente do som) então é propagado um erro.

- Se o quadro do vídeo está aquém do permitido (ou seja, o tempo do vídeo está mais que 15ms antes do tempo presente do som) então a apresentação é retardada em 1ms, e neste caso a diferença do tempo resultante é submetida a um outro teste.

- Finalmente, se o tempo do vídeo está suficientemente próximo do tempo do som associado, então o *OK* é dado para o gerenciador do vídeo.

O processo sincronizador de vídeo escreve-se então como:

```
process VIDEO_SYNCHRONIZER(v_time:time):=
          v_ready;VIDEO_CHECK(v_time+40)
endproc
```

```
process VIDEO_CHECK(v_time:time):=
     sound_query?s_time:time;
     [s_time-v_time > 150] → error!"out of sync":an_error;exit []
     [v_time-s_time > 15] → [v_time-s_time]i;VIDEO_CHECK(V_TIME + v_time-s_time) []
     [¬(s_time-v_time > 150) ∧ ¬(v_time-s_time > 15)] →
                                        ok_v;VIDEO_SYNCHRONIZER(v_time)
endproc


process INITIAL_VIDEO_CHECK:=
          v_ready;ok_v;VIDEO_SYNCHRONIZER(0)
        []
          [150]i;(error!"out of sync":an_error;exit)
endproc
```

Note-se que o operador de prefixação temporal de RT-LOTOS foi utilizado na especificação dos processos `INITIAL_VIDEO_CHECK` e `VIDEO_CHECK`.

Sincronizador de som: A especificação do sincronizador de som é bastante simples devido ao requisito que não permite jitter no som. O sincronizador do som precisa tratar com erros somente no caso onde o quadro de vídeo chega antes da primeira entidade de som. Nos outros casos o temporizador do som assegura que não ocorrerá jitter no dispositivo de som, já que o vídeo pode ser apresentado até 15ms antes do som associado. Desta forma, um erro seria propagado somente se a primeira entidade de som não chegasse em 15ms.

O peocesso sincronizador de som escreve-se então:

```
process SOUND_SYNCHRONIZER noexit:=
          s_ready;ok_s;SOUND_SYNCHRONIZER
endproc


process INITIAL_SOUND_CHECK:=
          s_ready;ok_s;SOUND_SYNCHRONIZER(0)
        []
          [15]i;(error!"out of sync":an_error;exit)
endproc
```

Note-se, mais uma vez, que a prefixação temporizada de RT-LOTOS foi utilizada para representar a situação de timeout no processo `INITIAL_SOUND_CHECK`.

Sincronizador: Os processos anteriores são combinados para produzirem o sincronizador:

```
process SYNCHRONIZER:=
       s_ready;ok_s;
           hide sound_query in SOUND_QUERY(0)
          |[sound_query]|
           (INITIAL_VIDEO_CHECK
          |||
           SOUND_SYNCHRONIZER))
         []
       v_ready;ok_v;
          hide sound_query, sound_clock in (SOUND_CLOCK
          |[sound_query, sound_clock]|
           (SOUND_SYNCHRONIZER(0)
          |||
           INITIAL_SOUND_CHECK))
endproc
```

O processos que compõem o processo SYNCRONIZER podem ser separados em dois grupos: os processos usados no início do algoritmo; e os que mantém o regime de sincronização. Assim, os processos INITIAL_SOUND_CHECK e INITIAL_VIDEO_CHECK fazem parrte do primeiro grupo e servem essencialmente para estabelecer uma referência inicial de tempo no qual o resto do algoritmo está baseado. Os processos SOUND_CLOCK e SOUND_QUERY fazem parte do segundo grupo e funcionam como relógios e fornecem os valores de tempo necessários aos processos sincronizadores. Por fim, os processos SOUND_SYNCHRONIZER e SOUND_SYNCHRONIZER são os responsáveis pela manutenção da sincronização.

III.4.2.3.3 LIP_SYNC: o algoritmo com tratamento de erros

Os erros definidos nos processos podem necessitar de algum tratamento, e optou-se pelo procedimento sumário de parar o algoritmo (ação shut) e informar a razão da parada na porta error_mssg.

```
process ERROR exit:=
   error?key:an_error; shut;
   [key="soundlate"] → error_mssg! "The last sound entity was available too late"
                                                              :string;exit
  []
   [key="videolate"] → error_mssg! "The last video frame was available too late"
                                                              :string;exit
  []
   [key="out of sync"] → error_mssg! "Algorithm unable to maintain synchronization slot"
                                                              :string;exit
endproc
```

E finalmente, o tratador de erros acima pode ser combinado com o algoritmo de sincronização de lábios resultando em uma especificação mais completa. Neste tratador de erros a ação shut representa o evento de "parada" do algoritmo LIP_SYNC no caso da ocorrência de erros.

```
process LIP_SYNC_error_handling:=
            hide error,shut in
            (LIP_SYNC [> shut; exit)
               |[error,shut]|
            ERROR
endproc
```

Como esta especificação do tratamento de erro não contém nenhuma referência direta ao tempo ela é a mesma apresentada em [Regan 93].

III.4.2.3.4 Comentários e comparação com Temporal LOTOS

Como a especificação apresentada neste trabalho tem a mesma estrutura da especificação correspondente dada na linguagem *Temporal LOTOS* [Regan 93], cabe aqui uma breve comparação entre aquela especificação e a apresentada aqui. Inicialmente, tem-se a observar que nenhuma modificação foi requerida nos processos que não requerem a representação explícita do tempo. Porém, nos processos onde é necessário o uso de operadores temporais a especificação correspondente em RT-LOTOS mostrou-se muito mais clara e intuitiva, visto que foi possível representar todas as situações do algoritmo enquanto em Temporal LOTOS alguns artifícios sintáticos foram necessários para atingir este objetivo. Observe-se também que em [Regan 93], o próprio autor afirma explicitamente que sua linguagem não permite representar corretamente algumas situações encontradas no algoritmo de sincronização de lábios como por exemplo no mecanismo de timeout encontrado no processo `TIMER` onde é necessário a especificação da ação de retardo unitário σ 150 vezes [Regan 93, pág.137].

III.4.3 Protocolo do bit alternante

Nesta seção é apresentado a especificação em RT-LOTOS do clássico exemplo do protocolo do bit alternante. Este protocolo (*alternating bit protocol*) faz parte da camada de transporte (camada quatro do modelo OSI) e permite a transferência de dados entre um par de entidades ligadas por uma conexão bi-direcional previamente estabelecida a qual pode perder mensagens.

Embora já tendo sido tratado em trabalhos anteriores [Camargo 90, Camargo 92], optou-se nesta dissertação por adaptar a descrição e notação utilizadas para o protocolo do bit alternante utilizada em [Garavel 89] para facilitar comparações e tornar evidente o alto poder expressivo de RT-LOTOS.

O serviço oferecido pelo protocolo do bit alternante

Observado de uma camada superior, o serviço fornecido pelo protocolo do bit alternante consiste no encaminhamento de uma série de mensagens da entidade emissora (T) para a entidade receptora (R). A transmissão é confiável, isto é: mensagens não podem ser duplicadas nem perdidas e são entregues na ordem em que são emitidas. Assim, o serviço fornecido por este protocolo pode ser especificado como:

```
specification ALTERNATING_BIT_SERVICE : noexit behaviour
   SERVICE[put,get]
where
   process SERVICE[put,get] :  noexit :=
      put?m:msg; (* aquisicao de uma mensagem *)
      get!m;     (* entrega da mensagem *)
      SERVICE[put,get]
   endproc
endproc
```

As mensagens são modeladas por números inteiros de 1 até N e são especificadas pelo tipo abstrato seguinte (não detalhado):

```
type MESSAGE is
   sorts msg   (* type msg=1..N *)
endtype
```

Descrição informal do protocolo do bit alternante

Passa-se a seguir à descrição do protocolo do bit alternante. O funcionamento "ideal" do protocolo do bit alternante é a seguinte: a entidade emissora T envia uma mensagem à entidade receptora R; na recepção desta mensagem R envia de volta uma confirmação a T. Contudo, o meio de transmissão entre T e R não é confiável, e é possível que mensagens ou confirmações possam ser perdidas. No caso de perda o meio pode, de maneira facultativa, sinalizar esta perda ao destinatário (T ou R) através do envio de uma indicação de perda. Para detectar perdas não sinalizadas, as mensagens e as confirmações contém um bit de controle. O bit de controle de cada confirmação é igual ao bit de controle da mensagem que ele confirma. Os bits de controle alternam a cada mensagem emitida e assim, têm valores distintos. Se T recebe uma indicação de perda de confirmação ou uma confirmação com bit de controle errado ou a sinalização de que o tempo de espera máximo para recepção da confirmação expirou-se, então ele retransmite a última mensagem enviada. Reciprocamente, se R recebe uma indicação de perda de mensagem ou uma mensagem com bit de controle errado, ele retransmite a última confirmação enviada.

A arquitetura geral do protocolo do bit alternante é mostrada na figura III.2. O processo TRANSMITTER representa a entidade emissora, o processo RECEIVER representa a entidade receptora, o processo MEDIUM1 representa a transmissão das mensagens de T para R, e o processo MEDIUM2 representa a transmissão das confirmações de R para T.

A tabela III.1 mostra os nomes de ações que serão utilizados na especificação do bit alternante juntamente com seus significados. As únicas ações oferecidas pelo serviço são put e get; todas as outras devem ser internas. Na tabela referida, m representa uma mensagem e b o bit de controle de uma mensagem.

O emissor e o receptor funcionam de maneira completamente assíncrona. Os dois meios funcionam também desta maneira. A especificação RT-LOTOS do protocolo do bit alternante é dada a seguir. Os processos TRANSMITTER, RECEIVER, MEDIUM1 e MEDIUM2 são definidos na seqüência. Observe-se que o parâmetro efetivo 0 nos processos TRANSMITTER e

RECEIVER servem para inicializarem o valor do bit de controle que será 0 para a primeira mensagem.

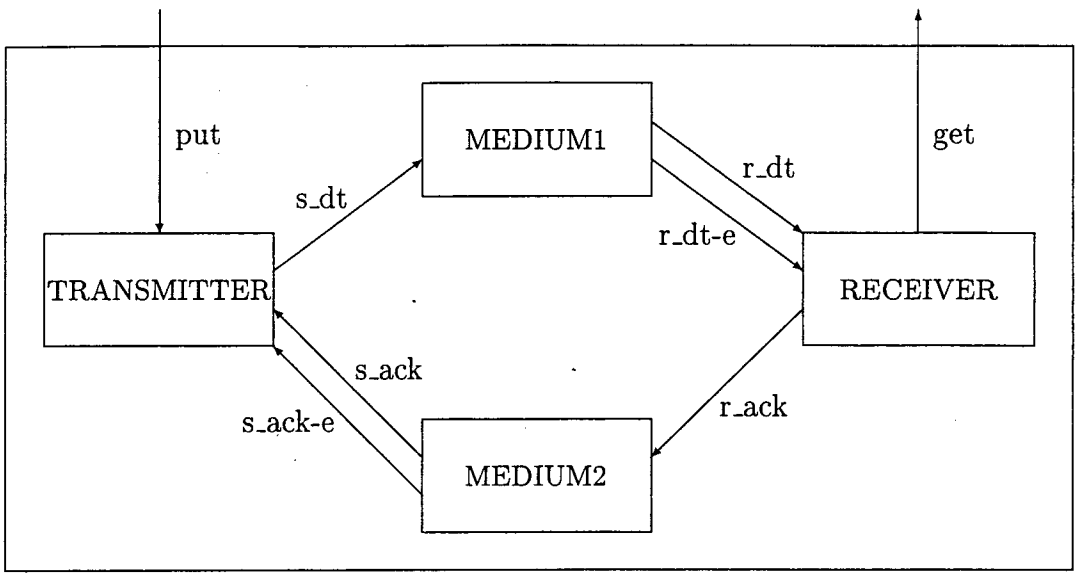


Figura III.2: Estrutura geral do algoritmo bit alternante [Garavel 89]

Nome	Origem	Destino	Significado
put!m	usuário	TRANSMITTER	emissão de uma mensagem
s_dt!m!b	TRANSMITTER	MEDIUM1	envio da mensagem
r_dt!m!b	MEDIUM1	RECEIVER	transmissão da mensagem
r_dt-e	MEDIUM1	RECEIVER	perda da mensagem
get!m	RECEIVER	usuário	recepção da mensagem
r_ack!b	RECEIVER	MEDIUM2	envio de uma confirmação
s_ack!b	MEDIUM2	TRANSMITTER	transmissão da confirmação
s_ack-e	MEDIUM2	TRANSMITTER	perda da confirmação

Tabela III.1: Significado das ações utilizadas no bit alternante [Garavel 89]

A especificação global do protocolo

A especificação do protocolo do bit alternante é dada a seguir:

```
specification ALTERNATING_BIT_PROTOCOL[put,get] :  noexit behaviour
   hide s_dt, r_dt, r_dt-e, r_ack, s_ack, s_ack-e in
     (
       (
       TRANSMITTER[put,s_dt,s_ack,s_ack-e](0)
       |||
       RECEIVER[get,r_dt,r_dt-e,r_ack](0)
       )
     |[s_dt,r_dt,r_dt-e,r_ack,s_ack, s_ack-e]|
       (
       MEDIUM1[s_dt,r_dt,r_dt-e]
       |||
       MEDIUM2[r_ack,s_ack,s_ack-e]
       )
     )
where
   type bit is
   sorts bit
   opns 0 :  -> bit
        1 :  -> bit
        not :  bit -> bit
   endtype
endspec
```

Hipóteses temporais: Se levará em conta ainda os retardos envolvidos nas transmissões das mensagens e confirmações. Será assumido que todas as mensagens têm um tamanho máximo fixado; desta maneira os tempos gastos para se colocar mensagens na linha de transmissão terão um limite superior. Para simplificar o exemplo, escolheu-se valores numéricos inteiros (expressos em mili-segundos) adequados para representar os tempos gastos nas diferentes ações do protocolo. Por exemplo, o valor utilizado para representar o tempo gasto na ação de colocar uma mensagem na linha de comunicação é de no máximo 80ms, e a duração da colocação de um reconhecimento na linha é de no máximo 20ms; o tempo devido aos retardos da linha de comunicação é de no máximo 100ms. Desta maneira, o tempo entre a emissão de uma mensagem até a recepção da sua confirmação será de no máximo 300ms. Estes valores serão utilizados nas especificações dos processos `TRANSMITTER`, `RECEIVER`, `MEDIUM1` e `MEDIUM2`.

O meio de comunicação é representado pelos processos `MEDIUM1` e `MEDIUM2` sendo que o primeiro representa o meio onde transitam as mensagens e o segundo representa o meio onde transitam as confirmações.

Ao receber uma mensagem `m` com um bit de controle igual a 1, o processo `MEDIUM1` fica sujeito a um atraso aleatório de no máximo 100ms (também, a colocação de uma mensagem e de uma confirmação no meio está sujeita a atrasos aleatórios máximos de 80ms e 20ms respectivamente), e depois pode reagir de três maneiras distintas:

- transmitir corretamente a mensagem e seu bit de controle; em nenhum caso o meio pode mudar o valor do bit de controle

- perder a mensagem e enviar uma indicação de perda ao processo RECEIVE
- perder a mensagem silenciosamente.

O processo MEDIUM1 pode ser representado por

```
process MEDIUM1[s_dt, r_dt, r_dt-e] :  noexit :=
   s_dt?m:msg?b:bit;     (* recepcao de uma mensagem *)
   [0,80]i;              (* atraso colocacao da mensagem *)
   [0,100]i;             (* atraso devido ao meio *)
     (
     r_dt!m!b;           (* transmissao correta *)
        MEDIUM1[s_dt, r_dt, r_dt-e]
     []
     r_dt-e;             (* perda com indicacao *)
        MEDIUM1[s_dt, r_dt, r_dt-e]
     []
     i;                  (* perda silenciosa *)
        MEDIUM1[s_dt, r_dt, r_dt-e]
     )
   endproc
endproc
```

O processo MEDIUM2 tem o comportamento análogo ao MEDIUM1 exceto pela mudança no nome das ações e também ao fato de que as confirmações só têm como informação o bit de controle.

```
process MEDIUM2[r_ack, s_ack, s_ack-e] :  noexit :=
   r_ack?b:bit;     (* recepcao de um bit *)
   [0,20]i;              (* atraso colocacao da confirmacao *)
   [0,100]i;        (* atraso devido ao meio *)
     (
     s_ack!b;       (* transmissao correta *)
        MEDIUM2[r_ack, s_ack, s_ack-e]
     []
     s_ack-e;       (* perda com indicacao *)
        MEDIUM2[r_ack, s_ack, s_ack-e]
     []
     i;             (* perda silenciosa *)
        MEDIUM2[r_ack, s_ack, s_ack-e]
     )
   endproc
endproc
```

A entidade emissora inicia a transmissão de uma mensagem a partir de um pedido do usuário via put e a transmite ao MEDIUM1 após ter juntado a mensagem o valor corrente b do bit de controle. Se ela recebe como resposta uma confirmação com um bit de controle b, então a transmissão teve sucesso, caso contrário é necessário retransmitir a mensagem.

Existem três causas possíveis para a retransmissão:

- o emissor recebeu uma confirmação tendo ¬b como bit de controle
- o emissor recebeu uma indicação de perda de confirmação s_ack-e
- o emissor após ter esperado por uma confirmação durante o tempo máximo de ida e vinda de uma mensagem (300ms) e não ter recebido nenhuma confirmação retransmite a mensagem com o mesmo bit de controle. Esta retransmissão é devida ao *timeout* provocado pela expiração da temporização armada quando da colocação da mensagem na linha. A expiração desta confirmação pode ter sido provocada pela perda silenciosa da mensagem ou da confirmação no meio de transmissão.

O processo TRANSMITTER a seguir descreve a entidade emissora.

```
process TRANSMITTER[put, s_dt, s_ack, s_ack-e] (b:bit) :  noexit :=
   put?m:msg;              (* aquisicao da mensagem *)
     TRANSMIT[put, s_dt, s_ack, s_ack-e] (b,m)
where
   process TRANSMIT[put, s_dt, s_ack, s_ack-e] (b:bit ,m:msg):  noexit :=
     s_dt!m!b;             (* emissao da mensagem *)
     (
     [0,300]s_ack?ok:bit;
        (
        [b=ok] -> TRANSMITTER[put, s_dt, s_ack, s_ack-e] (not(b))
       []                  (* bit de controle correto *)
        [b=not(ok)] -> TRANSMIT[put, s_dt, s_ack, s_ack-e] (b,m)
        )                  (* bit incorreto => reemissao *)
        <s_ack] TRANSMIT[put, s_dt, s_ack, s_ack-e] (b,m)
    []                     (* timeout => reemissao *)
     s_ack-e; TRANSMIT[put, s_dt, s_ack, s_ack-e] (b,m)
     )                     (* indicacao de perda => reemissao *)
   endproc
endproc
```

Note-se que na especificação do processo TRANSMITTER utilizou-se dois operadores temporais de RT-LOTOS: o operador de prefixação temporizado na ação "[0,300]s_ack?ok" e o operador de preempção temporal "<s_ack]". O uso destes operadores simplificou e deu maior estrututa à especificação.

A especificação da entidade receptora é descrito pelo processo RECEIVER e o seu comportamento é seguinte: ao receber uma mensagem com o bit de controle b correto a este processo entrega a mensagem ao usuário via get e responde enviando uma confirmação com o bit de controle igual a b. Nos outros casos é enviado uma confirmação incorreta tendo ¬b como bit de controle; esses casos são dois:

- o receptor recebeu uma mensagem tendo ¬b como bit de controle
- o receptor recebeu uma indicação de perda de mensagem r_dt-e

O processo RECEIVER é apresentado a seguir.

```
process RECEIVER[get, r_dt, r_ack, r_ack-e] (b:bit) :  noexit :=
   r_dt?m:msg!b;           (* bit de controle correto *)
      get!m;               (* entrega da mensagem *)
         r_ack!b;          (* envio de uma confirmação correta *)
            RECEIVER[get, r_dt, r_ack, r_ack-e] (not(b))
   []
   r_dt?m:msg!(not(b));    (* bit de controle incorreto *)
      r_ack!(not(b));      (* envio de uma confirmação incorreta *)
         RECEIVER[get, r_dt, r_ack, r_ack-e] (b)
   []
   r_dt-e;                 (* indicacao de perda *)
      r_ack!(not(b));      (* envio de uma confirmação incorreta *)
         RECEIVER[get, r_dt, r_ack, r_ack-e] (b)
endproc
```

Concluíndo a apresentação do exemplo do protocolo do bit alternante, pode-se observar que, com relação a especificação correspondente em LOTOS apresentada em [Garavel 89], a especificação em RT-LOTOS goza de vantagens evidentes. Em RT-LOTOS pôde-se representar todos os aspectos temporais. Por exemplo, representou-se os retardos aleatórios devidos à colocação e transmissão das mensagens, e também o mecanismo de *timeout* necessário para representar o protocolo com fidelidade e maior simplicidade.

III.5 Conclusão

Este capítulo foi dedicado à apresentação de uma extensão de RT-LOTOS para incorporar o tratamento de dados e também à apresentação de alguns exemplos nesta linguagem completa. Inicialmente, incorporou-se à sintaxe e semântica de RT-LOTOS o operador de prefixação com registro e captura do tempo de espera até a realização da ação '$a@t; E$'. Depois, procurou-se compreender as dificuldades envolvidas no acoplamento da linguagem de dados Act One ao formalismo RT-LOTOS. Observou-se que este problema é ortogonal ao de integrar um modelo de tempo em álgebras de processos e assim, evitou-se a apresentação formal da semântica de RT-LOTOS Completa, definindo apenas a sua sintaxe e a semântica informal. Finalmente foi apresentada uma bateria de exemplos de especificações em RT-LOTOS Completa construídos a partir de exemplos conhecidos na literatura. Isto permitiu realizar uma breve comparação do ponto de vista da representação entre RT-LOTOS e outros existentes. No próximo capítulo será apresentada uma comparação sobre vários aspectos entre RT-LOTOS e outras álgebras de processos existentes na literatura.

Capítulo IV

Conclusões e Perspectivas

Resumo do trabalho apresentado

Neste trabalho foi proposta uma extensão temporal para a linguagem de especificação formal LOTOS. A extensão proposta, chamada RT-LOTOS, incorpora o tempo explicitamente e permite representar sistemas onde o tempo intervém explicitamente.

Esta tese pode ser resumida da maneira que segue.

No capítulo I apresentou-se os conceitos básicos relacionados ao tempo e aos sistemas dependentes dele. Em particular, foram apresentados os requisitos temporais e características gerais dos sistemas tempo-real, protocolos de comunicação e aplicações multimídia. Apresentou-se também neste capítulo, uma visão geral da abordagem formal para desenvolvimento de sistemas dependentes do tempo, e um estudo comparativo das principais extensões temporais de LOTOS.

No capítulo II foi introduzida a álgebra de processos temporizada RT-LOTOS como uma extensão conservativa de LOTOS. Uma semântica operacional é definida para RT-LOTOS e um conjunto de propriedades básicas são estabelecidas e provadas para o formalismo. Estende-se também para o contexto temporal as principais noções de bissimulação de LOTOS e definem-se outras que utilizam explicitamente a dimensão do tempo. Definida completamente a álgebra de processos RT-LOTOS, apresenta-se uma série de pequenos exemplos básicos de processos dependentes do tempo como forma de provar informalmente o poder de expressão alcançado pelo formalismo definido. Este capítulo é finalizado com uma definição da semântica de RT-LOTOS em Grafos Temporizados. Isto é feito com o objetivo de abrir para RT-LOTOS novas possibilidades de se utilizar novas técnicas de verificação de sistemas dependentes do tempo que combinariam especificações de processos em RT-LOTOS, especificações de propriedades temporais na Lógica Temporal Tempo-Real TCTL e a verificação de modelos desta lógica nos Grafos Temporizados correspondentes às especificações dos processos.

A seguir, apresenta-se no capítulo III, uma extensão de RT-LOTOS, chamada de RT-LOTOS$^+$, que incorpora um operador de passagem de valores relacionado ao tempo que permite registrar o tempo ($@t$) que uma ação foi oferecida, e passar este valor para o processo que segue a esta ação. Na continuação deste capítulo, apresenta-se uma discussão sobre a problemática e dificuldades relacionadas ao acoplamento da linguagem de definição de dados Act One e posterior definição de uma linguagem RT-LOTOS$^+$ Completa que incorpora

o operador @t e todas as outras características existentes em LOTOS Completa. Concluindo o capítulo, apresenta-se um conjunto de exemplos de especificações em RT-LOTOS$^+$ Completa com a finalidade dupla de: exemplificar o alto poder de expressão desta linguagem; e compará-la com uma outra extensão temporal de LOTOS que lhe é mais próxima.

Comparação com outras extensões temporais de LOTOS

As principais diferenças entre RT-LOTOS e as principais extensões temporais de LOTOS existentes na bibliografia são sumariadas na tabela IV.1. Deve-se observar que os nomes das extensões temporais de LOTOS usados na tabela IV.1 referem-se aos formalismos apresentados nos trabalhos seguintes: Temporal LOTOS [Regan 93], TIC-LOTOS [Azcorra 90], T-LOTOS [Bolognesi 93] e ET-LOTOS [Leduc 94].

Características de RT-LOTOS	Iguais a RT-LOTOS	Diferentes de RT-LOTOS
Modelo computacional assíncrono	T-LOTOS, ET-LOTOS	TIC-LOTOS, Temporal LOTOS
Sintaxe e semântica independentes do domínio de tempo	T-LOTOS, ET-LOTOS	TIC-LOTOS, Temporal LOTOS
Ações Temporizadas	TIC-LOTOS	Temporal LOTOS, ET-LOTOS, T-LOTOS (em T-LOTOS pode-se temporizar ações opcionalmente com o operador "time")
Intervalos de tempo nas ações	TIC-LOTOS, T-LOTOS (pelo operador "time")	Temporal LOTOS, ET-LOTOS
Representação de violações temporais		Todos os outros (esta característica é inexistente em todos os outros formalismos)
Operador de tratamento de violações temporais		Todos os outros (esta característica é inexistente em todos os outros formalismos)
Sincronização urgente	T-LOTOS (pelo operador "time")	Temporal LOTOS, ET-LOTOS, TIC-LOTOS
Urgência no "hide"	Todos	
Não necessita de operadores especiais para representar delay, timeout e watchdog	TIC-LOTOS	T-LOTOS, ET-LOTOS, Temporal LOTOS

Tabela IV.1: Comparação de RT-LOTOS com outras extensões temporais de LOTOS

Tendo em vista a tabela anterior, observa-se que RT-LOTOS concentra em si importantes características que existem isoladamente em outros formalismos, e além disso é o único deles que define e permite tratar falhas temporais. Estas características mostraram-se úteis conforme visto nos exemplos apresentados nos capítulos II e III.

Contribuições e limitações

Os destaques da proposta Os pontos mais relevantes que foram alcançados neste trabalho são resumidos a seguir.

Características Gerais: Definiu-se completamente uma álgebra de processos temporizada que é uma extensão conservativa de LOTOS, com uma semântica operacional consistente e

que satisfaz as principais propriedades normalmente exigidas de uma álgebra de processos temporizada: aditividade, persistência e determinismo temporal (porém, outras propriedades como a variabilidade finita não são válidas em RT-LOTOS). Outrossim, a álgebra de processos definida mostrou-se expressiva o suficiente para representar uma grande variedade de sistemas dependentes do tempo.

Definiu-se uma extensão do operador de prefixação de LOTOS para tratar a temporização de ações. Esta extensão permitiu a introdução de dois conceitos importantes em RT-LOTOS:

- o intervalo de tempo que é associado às ações e que permite definir os instantes de tempo em que estas podem se realizar; e
- a expiração do tempo em que uma ação pôde ser realizada que define o conceito de violação temporal.

Estes dois conceitos permitem representar com fidelidade as situações de falhas temporais existentes comuns em sistemas dependentes de tempo e que não são tratadas em outras extensões temporais de LOTOS.

Expressividade: Os intervalos de tempo associados às ações e o conceito de violação temporal permitiram definir uma semântica de não urgência para as ações que não são forçadas a ocorrer no final dos intervalos que lhes são atribuídos. Adotando esta semântica conseguiu-se preservar o paradigma das álgebras de processos no qual as ações não são autônomas.

Com a adoção do conceito de ações temporizadas e do operador de prefixação estendido, obteve-se uma expressividade suficiente para representar situações de atrasos (*delays*), temporizações (*timeouts*) e *watchdogs* utilizando-se apenas a prefixação temporizada e os outros operadores ordinários de LOTOS.

Definiu-se ainda um operador especial para tratar ocorrências de violações temporais, ou seja, os casos em que uma ação não se realiza no intervalo de tempo que lhe é atribuído. Este operador é de grande utilidade no caso de representação e tratamento de falhas temporais.

Definiu-se também uma semântica para a sincronização de ações temporizadas que torna estas sincronizações urgentes e ocorrendo tão logo sejam possíveis. Esta solução permite que uma sincronização ocorra no primeiro instante em que seja possível para todas os processos participantes que se sincronizam. Nos casos em que a sincronização não é possível ocorrer devido à expiração do intervalo de tempo de uma ação participante, então a não possibilidade de sincronização provocará uma violação temporal, que poderá, eventualmente, ser tratada pelo operador de preempção temporal.

Poder de análise: Compatibilizaram-se os conceitos das bissimulações clássicas de LOTOS para o caso temporal, e definiram-se outras (bissimulações temporais forte e fraca, e a bissimulação temporal direta) em que o tempo tem participação ativa. Sob algumas restrições de ordem prática (adoção de um domínio de tempo esparso e atribuição de intervalos de tempo reduzidos às ações) estas novas bissimulações podem ser de grande valia para a verificação de sistemas dependentes do tempo, pois permitiriam o uso de técnicas ordinárias de verificação de bissimulações.

Foi ainda estabelecida uma semântica para RT-LOTOS baseada em Grafos temporizados que permitirá, a partir de RT-LOTOS, a utilização de importantes técnicas de verificação

para sistemas dependentes do tempo que não poderiam ser analisadas de outra forma no caso de domínio de tempo denso ou no caso geral de domínio de tempo esparso. Mais especificamente, a obtenção de Grafos Temporizados a partir de especificações RT-LOTOS permitirá a utilização de todas as ferramentas existentes para verificação de sistemas dependentes do tempo existentes para aquele formalismo.

As limitações da proposta Por outro lado, a proposta apresentada neste trabalho possui algumas limitações devidas principalmente à algumas escolhas feitas para a semântica e ao atual estado da arte nesta área de conhecimento. Entre as principais limitações deste trabalho pode-se incluir:

A semântica de RT-LOTOS é muito mais complexa que a de outras extensões temporais de LOTOS, como por exemplo T-LOTOS e ET-LOTOS. Isto deve-se à incorporação à semântica das ações especiais de violações temporais e a conseqüente introdução de novas regras semânticas. Se por um lado, estas ações de violação temporal adicionam uma expressividade extra; por outro, o preço a ser pago para alcançar estes resultados é alto em termos de complexidade da semântica.

RT-LOTOS não consegue, a exemplo de todas as outras extensões temporais de LOTOS, representar o conceito de ações não atômicas e com duração não nula. Durante a realização deste trabalho tentou-se suprir esta limitação para RT-LOTOS da seguinte forma: adotou-se uma semântica de ordem causal [Costa 92] e definiu-se um semântica temporal simplificada para o modelo resultante desta adoção. O que observou-se do modelo assim obtido é que ele possuía uma semântica muito complexa devido à junção da semântica temporal junto com a semântica de ordem causal, e isto desencorajou momentaneamente o prosseguimento dos estudos nesta direção. Contudo, uma vez que a semântica temporal de RT-LOTOS está completamente definida deve-se prosseguir com estudos nesta direção.

A semântica de sincronização de ações sob restrições temporais adotada para RT-LOTOS estabelece que uma sincronização deve ocorrer tão logo seja possível. E embora esta semântica seja mais flexível que aquelas adotadas em Temporal LOTOS, TIC-LOTOS e ET-LOTOS (ver item "urgência" na tabela comparativa apresentada na página I.3), ela é muito rígida se comparada à opção de semântica de sincronização feita em T-LOTOS.

Definiu-se o operador $@t$ como operador de RT-LOTOS Básico para registro do tempo. Porém, uma questão importante deixou de ser tratada, relacionada à definição deste operador através de uma semântica de grafos temporizados; observou-se recentemente que esta dificuldade atinge também ET-LOTOS (que também tem este operador), como pode ser observada nas limitações da transposição para grafos temporizados daquela extensão temporal de LOTOS [Daws 94]. Também em [Daws 94] foi levantada a possibilidade de definir o operador $@t$ como grafo temporizado através da definição da variável t como um relógio que pode ser parado. Contudo, tal solução deixa o grafo temporizado indecidível para verificação através da lógica TCTL. Assim sendo, pode-se afirmar que este problema não parece ter uma solução simples.

Enfin, ainda com relação à obtenção de grafos temporizados a partir de especificações RT-LOTOS, não pode-se deixar de destacar a impossibilidade de obtenção de grafos de especificações RT-LOTOS com comportamentos infinitos. Mas, isto se deve a uma limitação inerente ao formalismo Grafos temporizados.

Perspectivas futuras

Enumera-se a seguir os principais caminhos que serão seguidos para continuação dos trabalhos iniciados aqui, sendo que o planejamento para continuação do presente trabalho inclui as seguintes atividades principais:

1. *Semântica formal de RT-LOTOS$^+$ Completa*: A definição semântica de RT-LOTOS$^+$ Completa, isto é, o acoplamento de Act One ao formalismo básico, é uma tarefa que deverá ser empreendida futuramente. Em essência, para se realizar esta definição completa é necessário definir um mapeamento entre RT-LOTOS$^+$ Completa e um sistema de transição rotulado estruturado. Mapeamentos assemelhados ao que será necessário para o caso de RT-LOTOS$^+$ Completa são encontrados em [ISO 88] (para LOTOS), em [Bolognesi 93] (para T-LOTOS) e em [Leduc 94] (para ET-LOTOS) e envolvem geralmente três fases: o *flattening mapping* para produzir uma especificação canônica do formalismo; a geração do sistema de derivações de uma representação de dados e a interpretação da especificação algébrica canônica para produzir o sistema de derivações completo; e finalmente o mapeamento da especificação canônica do formalismo em um sistema de transições rotuladas.

2. *Incorporação de ações com duração*: Deve-se continuar os estudos já iniciados para incorporação de ações com duração no formalismo básico. Com a semântica temporal já bem definida, pretende-se incorporar uma semântica de paralelismo verdadeiro que permita a representação adequada de ações com uma duração não nula. Para isto, deve-se retomar os estudos para incorporação do modelo de semântica causal definido em [Costa 93] em RT-LOTOS básico. Com isto, apesar de perder a característica de ser uma extensão conservativa de LOTOS, será possível permitir a atribuição de durações às ações quando da suas especificações no formalismo assim estendido. Insiste-se neste aspecto, pois num contexto temporal as ações com duração são naturais em sistemas dependentes do tempo, sendo então interessante que sejam nativas em um formalismo que objetiva a representação deste tipo de sistemas.

3. *Realização de avaliação de desempenho a partir de RT-LOTOS*: A introdução de uma noção de probabilidades em RT-LOTOS possibilitaria a utilização deste formalismo como base para realização de estudos de desempenho. Um primeiro passo nesta direção já foi dado em [Camargo 92], onde estudou-se a possibilidade de se associar funções de densidade de probabilidades aos intervalos de tempo definidos para as ações e definiu-se um formalismo baseado em LOTOS que permite a especificação de operações de prefixação da forma

$$< t_1, t_2 > \ f\, a; E$$

onde a é o nome da ação, f é a função de densidade de probabilidade e $< t_1, t_2 >$ é o intervalo no qual a ação pode ocorrer e sobre o qual a função f está definida. Esta abordagem parece interessante do ponto de vista da expressividade pois permitiria representar com fidelidade alguns problemas físicos relacionados ao ambiente no qual o sistema irá funcionar. Contudo, do ponto de vista prático, esta alta expressividade se reflete na dificuldade em se construir ferramentas de análise para realização da avaliação de desempenho, sendo contudo, possível de se obter resultados realizando-se um conjunto de simulações sucessivas.

Recentemente, observando-se os avanços nesta área, optou-se por uma outra alternativa que parece mais interessante para incorporação das probabilidades em RT-LOTOS: a introdução no formalismo de um operador de escolha probabilista, onde seriam associadas pesos de probabilidade a cada uma das alternativas oferecidas em uma operação de escolha, nos moldes do que foi introduzido no CCS em [Hansson 91]. Esta abordagem é menos expressiva que a anterior, porém, a partir dela pode-se obter a geração de um grafo probabilista, o que parece mais interessante sob dois pontos de vista:

(a) permitirá a utilização da teoria de Markov para desenvolvimento de uma ferramenta para avaliação de desempenho, o que certamente possibilitará construir uma ferramenta poderosa para estudos de desempenho;

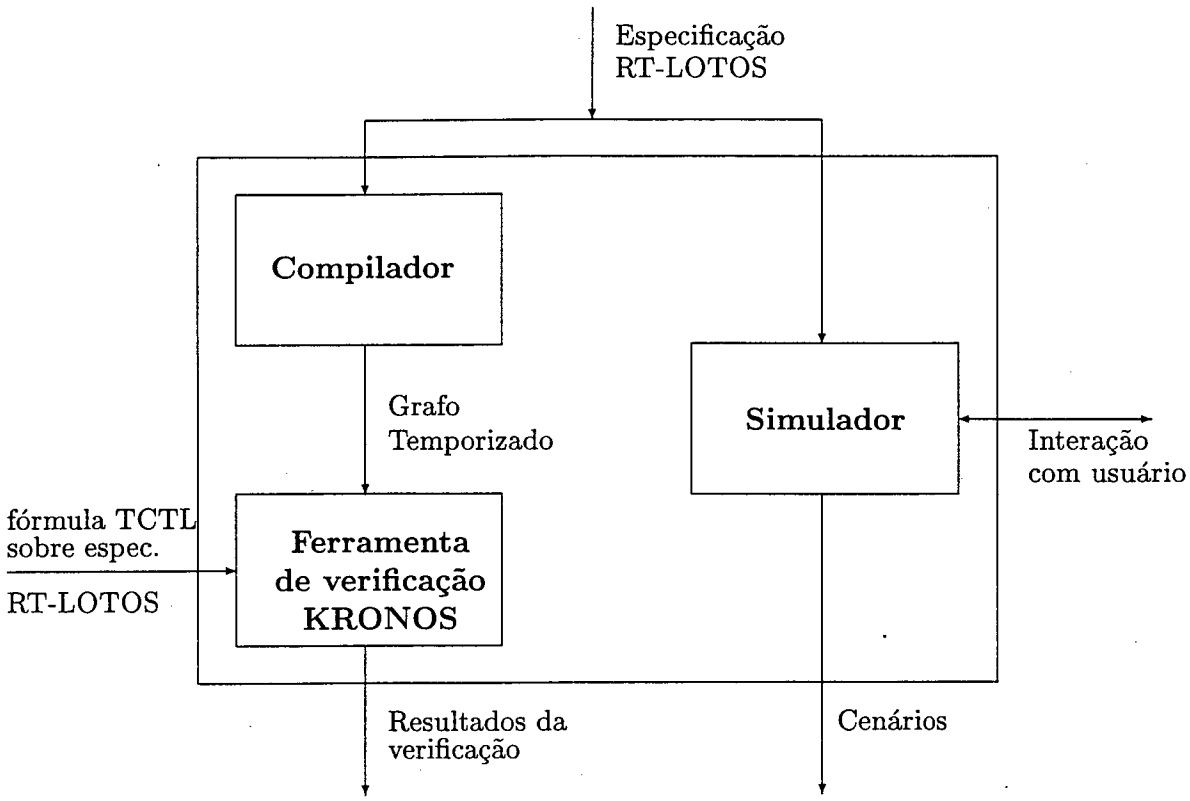
(b) permitirá a utilização da lógica temporal probabilista TPCTL [Hansson 91] (extensão probabilista da lógica temporal CTL) para verificação de propriedades quantitativas do tipo: "após uma solicitação de um serviço existe uma probabilidade de no mínimo 98% para que o serviço seja concluído em 2 segundos".

4. *Verificação de sistemas dependentes do tempo a partir de RT-LOTOS*: A definição de uma semântica de Grafos Temporizados para RT-LOTOS possibilitará a utilização de toda uma abordagem que utiliza a Lógica Temporal Tempo-Real TCTL [Alur 90] para especificação de propriedades temporais de sistemas especificados em RT-LOTOS e posterior verificação dos modelos nos Grafos Temporizados correspondentes às especificações. Recentemente, foi desenvolvido pelo laboratório VERIMAG uma ferramenta chamada KRONOS [Daws 94] que implementa algoritmos para verificação de modelos de TCTL em Grafos Temporizados e que poderá ser utilizada como ferramenta intermediária para verificação a partir de RT-LOTOS.

   Para isto, prevê-se a definição e a implementação de um ambiente computacional que possibilite a integração das atividades relacionadas à especificação e verificação de sistemas dependentes do tempo utilizando RT-LOTOS que deverá incluir: especificação de sistemas em RT-LOTOS, tradutor em Grafos Temporizados destas especificações, verificação da satisfação ou não das propriedades temporais especificadas em TCTL e um simulador para obtenção de cenários. Conseqüentemente, deve-se:

   - Definir e implementar uma ferramenta para compilação de especificações RT-LOTOS em Grafos Temporizados para poder utilizar diretamente ferramentas do tipo KRONOS. Esta atividade envolve algumas outras intermediárias como: análises lexicográfica e sintática de especificações RT-LOTOS, e a tradução de especificações em RT-LOTOS para Grafos Temporizados.
   - Definir e implementar um simulador RT-LOTOS.
   - Definir e implementar uma interface Homem-Máquina de fácil uso.

A figura que segue apresenta um esquema simplificado do ambiente pretendido.



Atualmente, este ambiente computacional para verificação de sistemas dependentes do tempo está em fase de especificação no contexto de uma dissertação de mestrado e pretende-se ter num prazo curto um protótipo deste ambiente em funcionamento.

5. *Aplicações reais usando RT-LOTOS*: Finalmente, tanto o formalismo, quanto o ambiente computacional desenvolvido deverão ser validados em aplicações reais nas áreas de sistemas tempo-real, protocolos de comunicação, sistemas multimídia e sistemas híbridos.

Apêndice A

Redes de Petri

A teoria de *redes de Petri* [Brams 83] (RdP) foi um dos primeiros formalismos introduzidos para tratar com concorrência, não-determinismo, e conexões causais entre eventos. Duas características das redes de Petri são a compreensibilidade dos modelos gerados, e a capacidade de expressar, de maneira natural, os problemas e propriedades referentes: ao paralelismo, à concorrência, ao seqüenciamento, ao conflito, ao não determinismo, e à sincronização.

A.1 Redes de Petri ordinárias

O modelo clássico de redes de Petri é descrito a seguir.

Existem dois grupos de conceitos elementares em uma rede de Petri:

1. Os conceitos que definem sua estrutura:
   - Os *lugares* que permitem a modelagem das atividades de um sistema; e
   - As *transições*, que ligadas a lugares de entrada e de saída, permitem representar a ocorrência de eventos num sistema.
2. Os conceitos que regem a sua evolução:
   - As *fichas* (dentro dos lugares) modelam recursos ou atividades em curso, ou seja, o estado do sistema; e
   - O *disparo de transições*, representado pela retirada de fichas de lugares de entrada e pelo seu depósito nos lugares de saída, significando a ocorrência de um evento no sistema.

Graficamente, em RdP os lugares são representados por círculos, as transições por barras, e as fichas por pontos dentro dos lugares.

Formalmente, define-se o modelo básico de redes de Petri (ou RdP ordinária) [Brams 83] como uma quíntupla $RdP ::= \langle P, T, I, O, M_0 \rangle$ onde:

- P é um conjunto de lugares ($P \neq \emptyset$);

- T é um conjunto de transições ($T \neq \emptyset$) e ($P \cap T = \emptyset$);
- $I: P \times T \rightarrow \mathbf{N}$, função de entrada das transições . I_{ij} indica o peso do arco que sai do j-ésimo lugar para a i-ésima transição ($\mathbf{N}$ representa o conjunto dos números naturais);
- $O: T \times P \rightarrow \mathbf{N}$, função de saída das transições. O_{ij} indica o peso do arco que sai da i-ésima transição para o j-ésimo lugar;
- $M_0: P \rightarrow \mathbf{N}$, marcação (número de fichas) inicial de cada lugar.

A dinâmica da RdP é representada pela evolução das fichas , causada pelos disparos das transições sensibilizadas, que provocam a transferência das fichas entre os lugares da rede, alterando sua marcação e o estado do sistema modelado.

Uma transição está *sensibilizada* pela marcação M quando todas as suas pré-condições forem satisfeitas:

$$\forall\, p_j \in P\ M(p_j) \geq I(t_i, p_j), \text{ ou seja, } M \geq I(t_i)$$

O *disparo* de t_i a partir da marcação M gera uma nova marcação M':

$$M'(p_j) = M(p_j) + O(t_i, p_j) - I(t_i, p_j) \quad \forall p_j \in P$$

ou seja, $M \xrightarrow{t_i} M'$, onde $M' = M + O(t_i) - I(t_i)$.

A figura A.1 mostra o disparo de uma transição. A transição t_1 da rede com marcação $M = (3, 2, 0)$ (parte (a) da figura A.1) está sensibilizada; o disparo de t_1 modifica a marcação da rede para $M' = (1, 1, 4)$ (parte (b) da figura A.1).

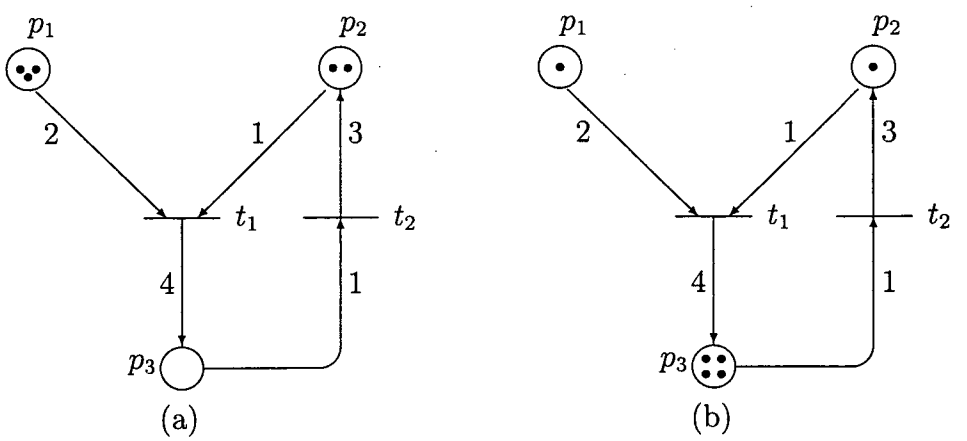


Figura A.1: Disparo de uma transição em uma RdP ordinária

Uma *seqüência* s de transições (t_j, $t_{j+1}, \cdots, t_{j+k}$) é uma seqüência de disparos a partir da marcação M_i se, e somente se existem marcações M_i, $M_{i+1}, \cdots, M_{i+k}$ tais que:

$$M_i \xrightarrow{t_j} M_{i+1} \xrightarrow{t_{j+1}} \cdots M_{i+k} \xrightarrow{t_{j+k}} M_{i+k+1}$$

escreve-se então:

$$M_i \xrightarrow{s} M_{i+k+1}$$

O conjunto $[M_0]$ das *marcações acessíveis* a partir da marcação inicial M_0 é definido como:

$$[M_0] = \{M_i \mid \exists s \text{ tal que } M_0 \xrightarrow{s} M_i\}$$

Este conjunto pode ou não ser finito, dependendo da topologia da rede e da marcação inicial M_0. O *grafo de marcações acessíveis* G associado à uma RdP com $[M_0]$ finito é formado pelo conjunto de marcações accessíveis M_0 e pelo conjunto U de arcos que representam os disparos de transições que levam de uma marcação a outra:

$$U = \{(M_j, t_i, M_k) \mid \exists t_i \in T \text{ tal que } M_j \xrightarrow{t_i} M_k,\ \forall\ M_j, M_k \in [M_0]\}.$$

Do exame deste grafo pode-se concluir a respeito das propriedades da rede, e este pode ainda ser utilizado para fins de verificação.

Redes de Petri têm sido criticadas por não serem aptas a tratar com *fairness* e com estruturas de dados, mesmo que um número de fichas em um determinado lugar possam uma varável local de programa [Ostroff 92]. Mecanismos de estruturação tais como operadores de composição não são naturais na teoria de redes de Petri. Um outro problema é que os grafos de alcançabilidade sofrem explosão de éstados quando uma rede de Petri torna-se grande, e isto torna difícil a análise de sistemas grandes.

A.2 Redes de Petri Temporais - o modelo de Merlin

O modelo redes de Petri Temporais de Merlin [Merlin 76] estende o modelo básico de redes de Petri associando a cada transição um intervalo de números, (t_{min}, t_{max}), definido sobre os números racionais não negativos. Neste modelo o disparo de transições não consome tempo (instantânea). Para que o disparo de uma transição possa ocorrer ela deve estar sensibilizada por um tempo mínimo t_{min}, quando se torna possível o seu disparo; ela pode ficar sensibilizada até um tempo máximo t_{max}, quando então ela *deve* necessariamente ser disparada, se ainda não o fez. Uma transição sensibilizada pode ser desabilitada pela chegada de uma ficha que sensibilize outra transição que possua um tempo máximo de disparo inferior ao da primeira transição. Esta característica do Modelo de Merlin torna-o expressivo o suficiente para representar situações de *timeouts* como ilustra o exemplo da figura A.2.

Formalmente, uma Rede de Petri Temporal (RdP-T) é definida [Diaz 91] como uma dupla:

$$RdP\text{-}T ::= \langle RdP, IT_0 \rangle$$

onde:

- RdP é o modelo de Rede de Petri ordinária subjacente; e
- $IT_0 : T \rightarrow \mathcal{Q}^+ \times \mathcal{Q}^+$.

  $t_i \rightarrow IT_i = [a_i, b_i]$ com $0 \leq a_i \leq b_i$

  $\forall i\ ,\ 1 \leq i \leq n$ com $n = Card(T)$

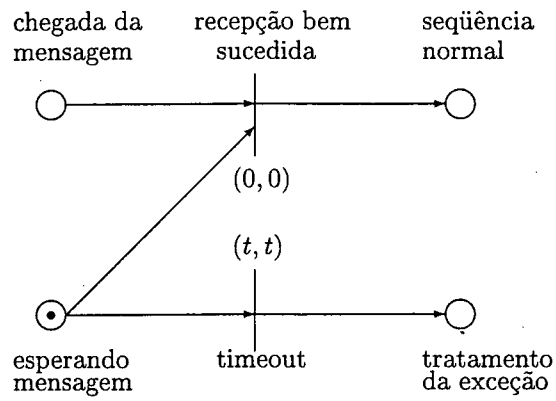


Figura A.2: Uma situação de *timeout* representada por uma rede de Petri temporal

IT_i é a função de intervalo de disparo inicial ($\mathcal{Q}^+$ é o conjunto dos números racionais não-negativos).

A figura A.3 mostra um exemplo de uma RdP-T.

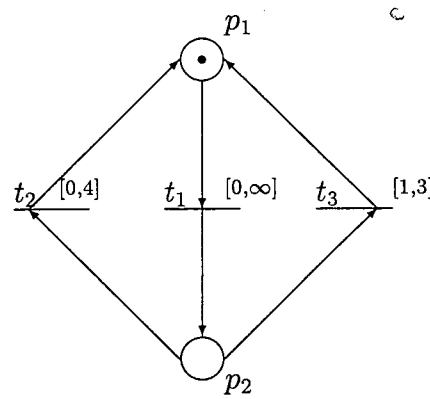


Figura A.3: Exemplo de uma rede de Petri temporal

Em uma rede de Petri temporal, um estado S é um par $S = (M, IT)$, onde M é a função de marcação corrente e IT é a função intervalo de disparo corrente associando com cada transição sensibilizada na rede o intervalo de tempo no qual a transição pode ser disparada.

O comportamento de uma rede de Petri temporal é regido pela regra de disparo de transições dada a seguir. De um estado $S = (M, IT)$, uma transição t é disparável no tempo θ se, e somente se as duas condições abaixo são verificadas:

1. a transição t está sensibilizada pela marcação M; e

2. o tempo θ deve ser maior ou igual ao tempo mínimo de disparo (t_{min}) da transição t e menor ou igual ao menor dos tempos máximos de disparo (t_{max}) de todas as transições sensibilizadas por M.

O próximo estado $S' = (M', IT')$ obtido do disparo da transição t no tempo θ é calculado através do seguinte algoritmo:

1. A nova marcação M' é obtida como nas redes de Petri ordinárias ordinárias.
2. A nova função de intervalo de tempo IT' é definida como:
   - para todas transições t_i não sensibilizadas pela marcação M', IT' é vazia;
   - para todas transições t_i sensibilizadas pela marcação anterior M e não em conflito com a transição t
     $$IT'_i = [max(0, t^i_{min} - \theta), t^i_{max} - \theta];$$
   - todas outras transições t_i (isto é, as transições t_i novamente sensibilizadas pela marcação M') recebem seus intervalos de disparo originais.

Dois fatos importantes envolvendo os valores θ merecem comentários [Diaz 91]:

1. Na regra de disparo, a origem do tempo para θ é a data no qual o estado S é atingido. Assim, um tempo relativo é usado dentro de cada estado.
2. A escala de tempo é densa, e geralmente θ pode variar dentro de um intervalo $[a, b]$, com $a \leq b$; e desta maneira, pode existir um número infinito de instantes de disparo θ para uma dada transição t.

O item 2) acima provoca um grave problema para a realização de uma análise por enumeração: o grafo de estados torna-se infinito. Entretanto, este problema é parcialmente contornado pela utilização do conceito de *grafo de classes de estados*. Neste grafo, cada classe de estado agrupa os estados com uma mesma marcação M, e um domínio de disparo da classe o qual é definido como a união dos intervalos de disparo associados a cada estado da classe. Em [Diaz 91] são discutidas técnicas de análise de redes de Petri temporais que utilizam este tipo de grafo.

Apêndice B

CCS e algumas de suas extensões temporais

Neste apêndice é realizada uma análise da incorporação do tempo e de outras características em diversas extensões de CCS [Milner 80, Milner 89].

A organização deste apêndice é a seguinte. Inicialmente o modelo de base do CCS é apresentado com o fim de facilitar a apresentação de suas extensões temporizadas. Em seguida, são apresentados resumos das características dos modelos.

B.1 CCS - Calculus of Communicating Systems

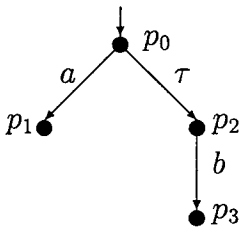
O CCS (*"Calculus of Communicating Systems"*) [Milner 80, 89] é um cálculo algébrico que descreve de forma abstrata e assíncrona, os conceitos do paralelismo e não determinismo de *agentes* (processos) seqüenciais comunicantes. O conjunto de agentes é visto como uma estrutura algébrica sobre a qual são definidas um certo número de leis de composição: por exemplo, se p e q são agentes, o agente $p+q$ é tal que pode comportar-se ou como p ou como q [Bougé 88].

Associado ao CCS, o conceito de *equivalência observacional* permite agrupar, em uma mesma classe de equivalência, agentes diferentes que tenham comportamentos idênticos; isto é, agentes cujo comportamento é indistinguível para um observador externo.

Em CCS, um agente se comunica com outro agente ou com seu ambiente através de interações. A forma elementar de uma interação é uma *ação*. Interações ocorrem se, ao mesmo tempo, um agente *oferecer* um evento e outro agente (ou seu ambiente) o *aceitar*, ou reciprocamente.

O CCS é baseado em três princípios básicos:

1. *Observação*: Descreve-se completamente um sistema determinando, de forma precisa, qual comportamento deve ser visto ou experimentado por um observador externo.
2. *Comunicação Síncrona*: Em CCS a comunicação (interação) é síncrona e bipartite; isto é, envolve exatamente dois agentes. A interação entre dois agentes ocorre quando eles oferecem ações complementares *observáveis* (por exemplo: a e $\bar{a}$) que quando



$ST = \langle P, A, T = (\xrightarrow{a}) \subset P \times P, p_0 \rangle$, onde
$P = \{p_0, p_1, p_2, p_3\}$
$A = \{a, b, \tau\}$
T é dado por: $\xrightarrow{a} = \{(p_0, p_1)\}$
$\xrightarrow{\tau} = \{(p_0, p_2)\}$
$\xrightarrow{b} = \{(p_2, p_3)\}$

Figura B.1: Exemplo de um sistema de transição

combinadas geram uma ação especial τ, não-observável, chamada ação *invisível* ou *silenciosa.*

3. *Computação Assíncrona*: Em CCS os agentes evoluem, computacionalmente, de maneira independente uns dos outros; neste sentido, CCS é um modelo que representa os processos de modo assíncrono.

B.1.1 O modelo de Base

Os agentes do CCS podem ser representados por *Sistemas de Transições Rotuladas.* Um Sistema de Transições Rotuladas (ST) é uma quádrupla:

$$ST = \langle P, A, T = (\xrightarrow{a})_{a \in A} \subset A \times A, p_0 \rangle$$

Onde: P é um conjunto de processos (agentes), $p \in P$ é um processo ou um estado de um processo; p_0 é o estado inicial; A é um conjunto de ações (ou observações), e $\xrightarrow{a}$ é uma relação que representa o efeito de uma ação $a \in A$.

Escreve-se

$$p \xrightarrow{a} q$$

com $p, q \in P$ se, e somente se a realização da ação a, sobre o processo p, transforma p em q.

Em complemento à definição de ST, um Sistema de Transições é dito *finito* se, e somente se os conjuntos P e A são finitos. A figura B.1 mostra um exemplo de um ST finito.

Outra forma de representação em CCS é a das *árvores de comportamento.* Tais árvores visam representar comportamentos de sistemas como sendo seqüências de escolhas de ações que são ordenadas no tempo de acordo com a profundidade do nó. Nestas árvores, a raiz representa o estado inicial do processo e as arestas são rotuladas com os nomes das ações. Assim, os rótulos na saída de cada vértice representam possíveis próximos passos do processo. A figura B.2 mostra exemplos de árvores de comportamento associadas a expressões CCS.

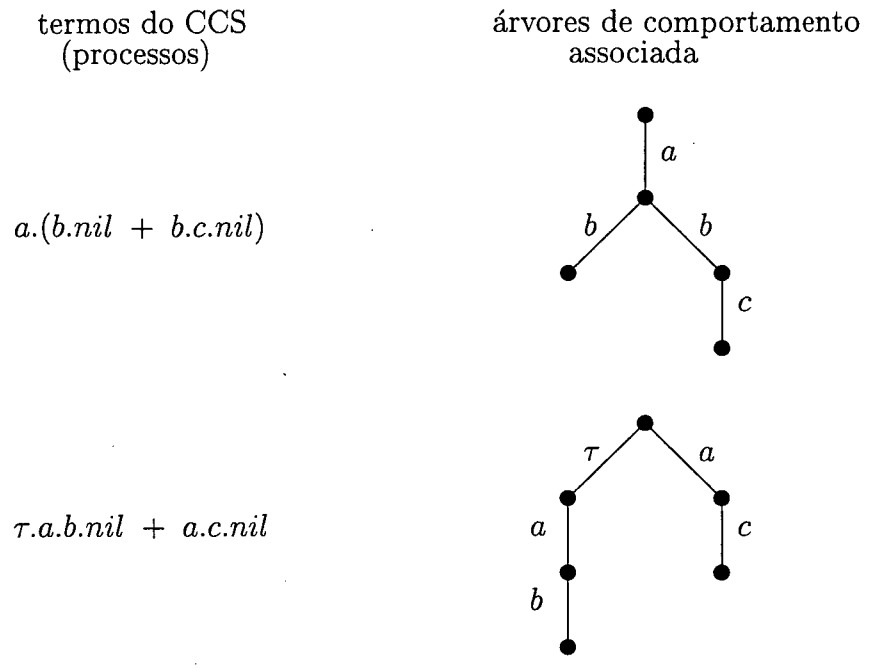


Figura B.2: Exemplos de processos CCS e de suas árvores de comportamento

O conjunto de ações do CCS é estruturado da forma que segue. Seja Δ um conjunto fixo de nomes cujos elementos são representados por a, b, $c, \cdots$, $\overline{\Delta}$ um conjunto de co-nomes representados por $\overline{a}, \overline{b}, \overline{c}, \cdots$, disjunto de Δ, e uma bijeção denotada por "$\bar{\cdot}$":

$$a \ (\in \Delta) \ \mapsto \ \overline{a} \ (\in \overline{\Delta}) \ ,$$

e tal que $\overline{\overline{a}} = a$. Desta forma, a e $\overline{a}$ são ditos complementares. Define-se o conjunto de ações A como $A = (\Delta \cup \overline{\Delta}) \cup \{\tau\}$; que representa o conjunto das ações observáveis e a ação invisível τ.

As ações complementares, juntamente com a ação τ, têm um significado importante na sincronização (comunicação) entre processos. Se dois processos distintos oferecem ações complementares, então estas podem ser ligadas (transformando-se em uma ação τ) permitindo que os dois processos se comuniquem sincronamente. Esta comunicação é observável somente pelos processos envolvidos na interação; do ponto de vista de um observador externo (um terceiro processo) esta sincronização corresponde a uma ação silenciosa τ.

B.1.2 O Cálculo de Processos CCS

O cálculo de processos CCS permite a representação de sistemas de transições ou árvores de comportamento a partir dos seus termos. Os termos CCS são formados tipicamente por aplicações de operadores (construtores de processos) sobre processos ou outros termos mais elementares.

A coleção de termos de um CCS "básico" é definida pela expressão BNF abaixo:

$$p ::= nil \mid a.p \mid p + p \mid p|p \mid p[S] \mid p\backslash a \mid recx.p$$

onde $p \in P$, $a \in A$, e S é uma função $S : A \mapsto A$, tal que $\overline{S(a)} = S(\overline{a})$.

Segue abaixo, a semântica transicional (operacional *à la Plotkin*) dos operadores de um CCS "básico":

- *nil*: Não existe nenhuma regra de inferência para *nil*. (Não permite nenhuma ação.)
- $a.p$: O operador *ação* representa uma ação de prefixação de um processo p por uma ação a. Sua regra transicional é:

$$\frac{-}{a.p \xrightarrow{a} p}$$

- $p + q$: O operador de *adição* representa a escolha entre dois processos. Se p e q são processos, então o processo $p + q$ representa um processo que se comporta ou como p, ou como q. Suas regras de transição são:

$$\frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

- $p|q$: O operador de *concorrência* entre dois processos é definido para duas situações: o entrelaçamento de ações independentes de cada um dos dois processos (as duas primeiras regras), ou a sincronização entre eles (a última regra).

$$\frac{p \xrightarrow{a} p'}{p|q \xrightarrow{a} p'|q} \qquad \frac{q \xrightarrow{a} q'}{p|q \xrightarrow{a} p|q'}$$

$$\frac{p \xrightarrow{a} p' \ , \ q \xrightarrow{\overline{a}} q' \ , \ a \in A}{p|q \xrightarrow{\tau} p'|q'}$$

- $p[S]$: O operador de *re-rotulação* consiste na aplicação de uma função de troca dos nomes de ações sobre processos. Sua regra é dada abaixo:

$$\frac{p \xrightarrow{a} q}{p[S] \xrightarrow{S(a)} p[S]}$$

- $p\backslash L$: A operação de *restrição* de um conjunto de ações $L \subset A$ consiste em ocultar as ações a e $\overline{a} \in L$, do processo p, de todos outros processos ou do seu ambiente.

$$\frac{p \xrightarrow{a} q}{p\backslash L \xrightarrow{a} q\backslash L} \quad (a, \overline{a} \notin L)$$

B.1.3 O Conceito de equivalência observacional

O conceito de equivalência observacional é uma característica do CCS, e é atualmente um importante instrumento utilizado na verificação de especificações de sistemas. Intuitivamente, equivalência observacional é uma relação de equivalência, onde dois processos são considerados equivalentes se for impossível distingui-los através de experimentos realizados por um observador.

Informalmente, dois processos p e q são equivalentes observacionalmente se, e somente se para toda seqüência s de ações de A^*[1]:

[1] A^* denota o conjunto de todas as cadeias de caracteres possíveis formadas com elementos do conjunto A

1. para todo resultado p' de um s-experimento em p, existe um *resultado equivalente* q' de um s-experimento em q; e
2. para todo resultado q'' de um s-experimento em q, existe um *resultado equivalente* p'' de um s-experimento em p.

Em [Milner 80], Milner define três tipos de equivalência para o CCS: equivalência observacional fraca (denotada por $\approx$), congruência observacional (denotada por $\cong$), e equivalência observacional forte (denotada por $\sim$). A relação existente entre elas é mostrada pelo seguinte resultado (prova em [Milner 80]): Se p e q são processos, então:

$$p \sim q \implies p \cong q \implies p \approx q \qquad \text{(ou seja, } \sim \subseteq \cong \subseteq \approx\text{)}$$

Finalizando esta seção, seguem alguns exemplos de equivalências em CCS:

a) $a.\tau.nil \approx a.nil$, mas, $a.\tau.nil \not\sim a.nil$

b) $a.\tau.b.nil \approx a.b.nil$

c) $a.(b.nil + \tau.b.nil) \approx a.b.nil$

d) $a.(b.nil + \tau.(c.nil + \tau.d.nil)) \approx a.(b.nil + \tau.(c.nil + \tau.d.nil)) + a.(c.nil + \tau.d.nil) + a.d.nil$

e) $a.(b.nil + \tau.c.nil) \not\approx a.(b.nil + c.nil)$

f) $a.(b.nil + \tau.c.nil) \not\approx a.(b.nil + c.nil) + a.c.nil$

g) $a.nil + b.nil + \tau.b.nil \not\approx a.nil + b.nil$

Isto termina a apresentação do modelo de base de CCS. Na seção que segue são apresentadas algumas importantes extensões temporais de CCS.

B.2 Extensões Temporais de CCS

Esta seção é dedicada a um estudo de algumas extensões temporais de de CCS. Para cada modelo, será apresentado um resumo de suas características e seguido de alguns comentários.

B.2.1 SCCS e ASCCS [Milner 83]

O "Synchronous Calculus of Communicating Systems" (SCCS) e o "Asynchronous Calculus of Communicating Systems" (ASCCS) [Milner 83] são cálculos algébricos que permitem representar sistemas síncronos e assíncronos. O SCCS é um cálculo síncrono, e o ASCCS é um sub-cálculo assíncrono do SCCS. Segundo Milner, estes cálculos não são bem apropriados para descrições de sistemas tempo-real. Como veremos adiante, o SCCS possui um modelo de tempo implícito, e a sua representação explícita de conceitos temporais é precária. Por outro lado, o arcabouço teórico desenvolvido neste trabalho é base de outros importantes modelos orientados ao tempo-real (ex. [Cardelli 82]), e esta é a principal razão para sua discussão neste capítulo, dedicada a modelos algébricos para STR.

SCCS

Como no seu modelo de base CCS, o SCCS modela um sistema em termos de agentes. Estes agentes podem efetuar ações que são consideradas como indivisíveis no tempo (atômicas) e instantâneas. Mas estas não são indivisíveis em todos os sentidos [Dzierzgowski 90]. De fato, sejam duas ações a e b, a ação $a.b$, chamada o produto de a e b, consiste em efetuar simultaneamente as ações a e b. SCCS exige que o produto "." faça que o conjunto das ações operadas por ele seja um grupo abeliano[2]. Considerando Act o conjunto de ações de um dado sistema a ser descrito por SCCS, isso quer dizer que $(Act, ., 1)$ tem as seguintes características:

- o produto $a.b$ de duas ações quaisquer a e b sempre deve existir;
- existe uma ação (unidade) 1, que corresponde ao efeito de não realizar nada;
- para cada a existe uma ação simétrica (complementar) $\bar{a}$ tal que $a.\bar{a} = 1$ (a exemplo das ações complementares e a ação τ do CCS);
- o produto de duas ações a e b, é independente da ordem em que escrevem-se as ações, $a.b = b.a$.

A cada agente é associado um conjunto de ações possíveis. Na realização de uma destas ações, este agente se transforma em um outro, que também tem seu conjunto de ações possíveis. As realizações das ações dos agentes se sucedem até a transformação num agente cujo conjunto de ações possíveis é vazio, e que não pode assim realizar mais nenhuma ação. Como no CCS, utiliza-se em SCCS a notação $P \xrightarrow{a} P'$ para dizer que *"P é capaz de realizar a ação a e transforma-se em P' ao fazê-lo"*.

As expressões SCCS são construídas a partir das ações, fazendo uso dos operadores que passamos a descrever a seguir. Milner descreve os operadores do seu cálculo através de semântica operacional; para cada operador é descrito seu significado informal e a regra transicional associada a ele. Os operadores são:

- *Ação*: se a designa uma ação e E uma expressão, então "$a : E$" representa um agente que é capaz de realizar a ação a e transformar-se em E ao fazê-lo.

$$a : E \xrightarrow{a} E$$

  O operador unário "$a :$" tem o efeito de prefixar a ação atômica a ao comportamento representado por E.

---

[2]Um grupo abeliano é um conjunto $\mathcal{G}$ munido de uma lei de composição interna "." que satisfaz as cinco condições seguintes:

1. $\forall a, b \in \mathcal{G},\ a.b \in \mathcal{G}$ ($\mathcal{G}$ é fechado sob ".");
2. $\forall a, b, c \in \mathcal{G},\ (a.b).c = a.(b.c)$ (associatividade);
3. $\exists 1 \in \mathcal{G}$, tal que $\forall a \in \mathcal{G}\ \ a.1 = a$ (existência de uma unidade);
4. $\forall a \in \mathcal{G}\ \exists a^{-1}$, tal que $a.a^{-1} = 1$ (simetria); e
5. $\forall a, b \in \mathcal{G},\ a.b = b.a$ (comutatividade).

- *Somatório*: $\sum \tilde{E}$ é uma expressão onde $\tilde{E} = \langle E_i \mid i \in I \rangle$ é alguma família (possivelmente infinita) de expressões indexadas. Escreve-se também $\sum_{i \in I} E_i$. Sua regra de derivação é

$$\frac{E_i \xrightarrow{a} E}{\sum_{i \in I} E_i \xrightarrow{a} E} \quad (i \in I)$$

Isto é, a ação da linha de baixo pode ser inferida de qualquer ação $E_i \xrightarrow{a} E$, para algum $i \in I$.

Dois casos especiais são importantes, para $I = \emptyset$ e quando I tem dois elementos. Escreve-se

$$0 \text{ para } \sum_{i \in \emptyset} E_i \ ,; \qquad E_0 + E_1 \text{ para } \sum_{i \in \{0,1\}} E_i.$$

0 é chamado de *inação*; ele não tem ações, desde que nada pode ser inferido.

O somatório representa uma superposição de agentes. Cada ação de uma soma determina uma parcela como uma fonte de um comportamento futuro.

- *Produto*: Se E e F são duas expressões, então $E \times F$ denota a evolução paralela dos agente E e F; sua regra de derivação é

$$\frac{E \xrightarrow{a} E' \ , \quad F \xrightarrow{b} F'}{E \times F \xrightarrow{a.b} E' \times F'}$$

- *Ação Restrição*: $E \backslash A$ é uma expressão, para qualquer subconjunto $A \subset Act$ que contenha o elemento 1. Sua regra de derivação é

$$\frac{E \xrightarrow{a} E'}{E \backslash A \xrightarrow{a} E' \backslash A.}$$

Assim o efeito de $\backslash A$ é inibir qualquer ação que não seja de A.

- *Definições mutuamente recursivas de constantes*: se $E_0, \cdots, E_n$ são expressões, então $K_0 \Leftarrow E_0, \cdots, K_n \Leftarrow E_n$ formam um conjunto de definições mutuamente recursivas de constantes $K_0, \cdots, K_n$, os K_i podem aparecer nos E_j. As ações que um K_i é capaz de efetuar são dadas pela seguinte regra:

$$\frac{E_i \xrightarrow{a} E'}{K_i \xrightarrow{a} K'}$$

- *"Delay"*: por meio de uma definição mutuamente recursiva ($\Leftarrow$), Milner define um operador *delay*: se E é uma expressão, δE é definido por $\delta E \Leftarrow 1 : \delta E + E$ [Dzierzgowski 90]. A exemplo das anteriores, as regras derivadas por Milner para δE são:

$$\delta E \xrightarrow{1} \delta E \qquad \frac{E \xrightarrow{a} E'}{\delta E \xrightarrow{a} E'}$$

Intuitivamente, δE pode ficar ocioso indefinidamente (isto é, por uma incontável quantidade de unidades de tempo "1") antes de se comportar como E. Esta é uma maneira de modelar uma espera (que pode não terminar); e no momento onde δE efetua uma ação de E, ele se transforma em um outro agente.

O tempo em SCCS é discreto, e como pode ser observado acima não é definido explicitamente na sua semântica formal. No modelo assume-se a existência de um relógio global no qual todos os agentes estão sincronizados. Mesmo assim, Milner sugere que *delays* podem ser representados explicitamente por $(1:)^n$, e um *delay* limitado pode ser escrito como δ_n, onde $\delta_0(P) \equiv P$ e $\delta_{n-1}(P) \equiv P + \delta_n(P)$.

O SCCS é um modelo qualificado de *síncrono*. O sentido que é dado a este termo é discutido a seguir [Milner 83, Dzierzgowski 90]. Considere, por exemplo o agente "$a : E$". Este agente torna-se E, *imediatamente após* ter realizado a ação a. Similarmente, se $1 \in Act$ é considerada uma ação silenciosa ou de espera, então "$a : 1 : E$" torna-se E exatamente um instante após ter realizado a ação a. Pode-se pensar nos agentes como sendo sincronizados por um relógio universal. Por outro lado, $a : \delta E$ pode ficar ocioso indefinidamente, após ter realizado a ação a, e antes de tornar-se E.

ASCCS

O ASCCS é um sub-cálculo assíncrono do SCCS. Seus agentes são também definidos por expressões. Estas expressões são construídas a partir de um conjunto de operadores que são aqueles mesmos do SCCS, mas com os operadores "$a :$" sendo eliminados, e substituídos por operadores "$a.$", definidos por $a.E = a : \delta E$. "$a.E$" significa que E pode ficar ocioso um tempo indefinido após a realização da ação a; o que para Milner caracteriza uma computação assíncrona.

B.2.2 Real Time Agents [Cardelli 82]

O cálculo algébrico apresentado em [Cardelli 82] é inspirado nos cálculos síncrono e assíncrono de Milner descrito em [Milner 83]. As principais características que diferenciam o *Real Time Agents* dos SCCS e ASSCS de Milner é utilização de uma representação explícita e densa do tempo e a definição de um operador de não-determinismo no modelo de Cardelli.

A idéia central do modelo Real Time Agents é o uso explícito de informação de tempo na representação do comportamento de agentes. O tempo é denso, isto é, $\forall t_1, t_2$ representando instantes de tempo, com $t_1 < t_2$, $\exists$ um instante t tal que $t_1 < t < t_2$. As ações de agentes tempo-real são observadas por uma duração de tempo e não por instantes de tempo (ou seja, não existem intervalos com medida nula); as variáveis que denotam o tempo são definidas no conjunto dos reais estritamente positivos, $\Re^+$.

Existem dois tipos de ações: ações determinísticas e ações não-determinísticas. As ações determinísticas são usadas para descrever o comportamento de agentes determinísticos. Tais agentes são, informalmente, todos aqueles que tem um único desenvolvimento possível no tempo.

Seja A um conjunto de ações. Os agentes determinísticos são construídos a partir de um conjunto inicial de operadores. Tais operadores consistem de:

- uma constante ll representando um agente neutro que sempre realiza a ação neutra 1;
- um operador unário de prefixação "$a[t]$:" que representa o fato de realizar a ação a por uma duração de tempo t; e

- um operador binário infixo X representando a composição síncrona (coexistência) de dois agentes.

O comportamento dos agentes é definido por um conjunto de relações binárias " $\xrightarrow[t]{a}$ " (para $a \in A$ e $t \in \Re^+$) sobre o conjunto de agentes. Escreve-se "$p \xrightarrow[t]{a} q$ " para representar: "p vai a q realizando a por uma quantidade de tempo t", ou "p leva t para ir sob a influência da ação a até q". As regras de redução para os agentes determinísticos são:

$$(ll \rightarrow) \qquad ll \xrightarrow[t]{1} ll$$

$$(a[] \rightarrow) \qquad a[t] : p \xrightarrow[t]{a} p$$

$$(a[]a[] \rightarrow) \qquad a[t+u] : p \xrightarrow[t]{a} a[u] : p$$

$$(X \rightarrow) \qquad \frac{p \xrightarrow[t]{a} p', q \xrightarrow[t]{b} q'}{pXq \xrightarrow[t]{ab} p'Xq'}$$

Por exemplo, a regra "$(a[] \rightarrow)$" diz que "$a[t] : p$" leva t para ir sob a para p, com $t > 0$, e a regra "$(X \rightarrow)$" dá o significado da coexistência de dois agentes: se p leva t para ir sob a para p' e q leva t para ir sob b para q', então "pXq" leva t (o mesmo t) para ir sob $a.b$ para $p'Xq'$.

Para os agentes não-determinísticos, considere o conjunto de operadores dado anteriormente acrescido dos seguintes operadores:

- Uma constante 0 representando um agente sem ações; quando um sistema alcança o estado 0, uma catástrofe ocorre e o tempo pára de correr, assim 0 é chamado de desastre.
- Um operador unário de prefixação "$a(t)$:" realizando a ação a por um intervalo de tempo positivo de *no máximo* t; "$a(t)$:" indroduz um não-determinismo chamado por Cardelli de horizontal (no sentido de que linhas podem ser traçadas horizontalmente de acordo com a duração de um "$a(t)$:").
- Um operador binário infixo + representando a escolha entre dois comportamentos; + introduz um não-determinismo discreto vertical.

Pode-se imaginar o comportamento de um agente como uma trajetória descontínua no plano, com o tempo no eixo x e o conjunto de ações no eixo y.

A semântica operacional é a seguinte:

$$(a() \rightarrow) \qquad a(t) : p \xrightarrow[v]{a} p \qquad v \leq t$$

$$(a()a() \rightarrow) \qquad a(t+u) : p \xrightarrow[v]{a} p + a(u) : p \qquad v \leq t$$

$$(+ \rightarrow) \qquad \frac{p \xrightarrow[t]{a} p'}{p + q \xrightarrow[t]{a} p'} \qquad \frac{q \xrightarrow[u]{b} q'}{p + q \xrightarrow[u]{b} q'}$$

Não existe axiomas para 0. O agente "$a(t) : p$" leva o tempo $v \leq t$ para ir sob a influência de a até p, e "$a(t+u) : p$" leva o tempo $v \leq t$ para ir sob a para $p+a(u) : p$. Desta maneira, $a(t) : p$ pode escolher, em qualquer evolução, por encurtar sua vida avançando alguma quantidade de tempo; de qualquer maneira em qualquer ponto no tempo ele pode parar sua ação a começar a realização de p.

A comunicação no modelo de Cardelli é realizada de modo semelhante ao CCS [Milner 80], isto é, através do "matching" de duas ações complementares a e $\overline{a}$, resultando $a\overline{a} = 1$. O que significa que a comunicação envolve exatamente dois agentes.

Finalmente, o modelo de Cardelli permite, a exemplo do modelo de Milner [Milner 83], a descrição de um mundo síncrono determinístico através dos primeiros operadores $(ll, a[t] :, X)$, e de outro assíncrono não-determinístico por $(0, a(t) :, +)$. Acrescente-se nisto a introdução do tempo denso de forma explícita.

B.2.3 Temporal CCS [Moller 89]

O "Temporal Calculus of Communicating Systems" (TCCS) [Moller 89] é uma extensão temporizada do CCS de Milner [Milner 89]. TCCS é um modelo assíncrono no qual o tempo passa de maneira independente do aspecto funcional de um processo. O sistema transicional, sob o qual o cálculo é construído, é dividido em duas partes ortogonais: uma descreve os aspectos funcionais do processo; e a outra seu aspecto temporal. Segundo os autores, existe uma motivação física para a imposição desta separação que eles justificam pela afirmação seguinte: "computação envolve trocas de energia, e existe um resultado da mecânica quântica que diz que trocas de energia e tempo não podem ser medidos simultaneamente". Com estes argumentos os autores advogam que em modelos baseados na observação do tempo e da computação não se deve permitir que estas duas atividades sejam observadas simultaneamente.

As principais características do TCCS são:

- a definição de dois conjuntos de regras semânticas formais distintas e ortogonais, uma para representação dos aspectos funcionais dos processos, e outra para a descrição dos aspectos temporais;
- a utilização de um modelo de tempo discreto (mapeado sobre os inteiros positivos) e global (os "tics" do relógio são sentidos por todos os processos do sistema).
- devido a separação ortogonal dos conjuntos de regras semânticas, o TCCS possui uma característica bastante desejável em modelos para especificação de sistemas dependentes do tempo: as partes de uma aplicação dependentes do tempo são descritas utilizando os construtores temporais, enquanto as partes da aplicação que não dependem do tempo podem ser descritas utilizando apenas os construtores funcionais. Desta forma, não é necessário "poluir" partes de uma aplicação não dependente do tempo com declarações temporais desnecessárias.

A Linguagem TCCS

A definição de linguagem é feita a partir dos elementos descritos a seguir. Seja Λ um conjunto de símbolos de ações atômicas não contendo τ nem ε (ações invisíveis), e seja $Act = \Lambda \cup \{\tau\}$.

Considere-se a bijeção "complemento" $\bar{}$ entre os elementos de Act, e tal que $\bar{\bar{a}} = a$. Como em CCS, esta bijeção é a base da comunicação. Seja também $\mathcal{T} = \{1, 2, 3, \cdots\}$ representando o tempo, e considere-se um conjunto Var de variáveis de processos.

A coleção de expressões TCCS é definida pela expressão BNF abaixo. Considere $a \in Act$, $X \in Var$, $t \in \mathcal{T}$, e S representando funções de re-rotulação, tais que $S : \Lambda \to \Lambda$ tal que $\overline{S(a)} = S(\bar{a})$ e $S(\tau) = \tau$.

$$P ::= 0 \mid X \mid a.P \mid (t).P \mid \delta.P \mid P \oplus P \mid P + P \mid \ P|P \ \mid P\backslash a \mid P[S] \mid \mu_i \tilde{x}.\tilde{P}$$

A interpretação informal dos operadores é a seguinte:

- 0 representa o processo *nil*, nenhuma ação ou passagem do tempo é realizada.
- X representa o processo denotado por X.
- $a.P$ representa o processo que realiza a ação a e após isso evolui para o processo P.
- $(t).P$ representa o processo que evolui para o processo P após exatamente t unidades de tempo.
- $\delta.P$ representa o processo que se comporta como o processo P, mas está pronto para esperar qualquer quantidade de tempo para isso. Este processo está esperando para sincronizar ou se comunicar com o ambiente, e ficará esperando até que o ambiente esteja pronto a participar de uma tal comunicação.
- $P + Q$ é chamado de escolha *forte* e representa a escolha entre dois processos P e Q. O processo se comportará ou como P ou como Q, com a escolha sendo feita no tempo da primeira ação. Uma passagem inicial do tempo deve ser permitida por ambos, P e Q.
- $P \oplus Q$ é chamado de escolha *fraca* e também representa a escolha entre P e Q, com a seguinte diferença: o processo se comportará ou como P ou como Q, com a escolha sendo feita no tempo da primeira ação, ou na ocorrência da passagem do tempo onde somente um dos operandos pode permitir que a passagem do tempo ocorra. Neste caso, o segundo processo "parado" deve ser eliminado da computação.
- $P \mid Q$ representa a composição paralela de dois processos, onde cada processo pode realizar ações independentemente, ou podem sincronizar em ações complementares, resultando a ação τ. A passagem do tempo é permitida em P e Q.
- $P\backslash a$ representa o processo P sem a ação a, isto é, a ocorrência de a não é permitida.
- $P[S]$ representa o processo P com as ações re-rotuladas pela função de re-rotulação S.
- $\mu_i \tilde{x}.\tilde{P}$ representa a solução x_i tomada das soluções para as definições mutuamente recursivas de processos $\tilde{x}$ definidas como soluções particulares para as equações $\tilde{x} = \tilde{P}$.

O processo "0" age como um processo "deadlock" não permitindo a realização de nenhuma ação, nem mesmo a passagem do tempo. "0" age como um "aniquilador" para escolha forte + e composição paralela "|" de processo com guardas temporizados, e como uma unidade com respeito ao operador de escolha fraca "$\oplus$". Assim, um "deadlock" temporal local pode implicar em um "deadlock" global. Desta maneira, os autores descrevem um processo *deadlock não-temporal* $\delta.0$ que permite a passagem do tempo mas não realiza nenhuma ação.

São definidos dois operadores derivados dos anteriores:

- a *prefixação de processos*:

$$INIT_t(P) \stackrel{def}{=} P \mid (t).0$$

permite a execução normal de um processo por uma quantidade fixa de tempo, e então termina quando ele deve permitir a passagem do tempo.

- o operador *timeout*:

$$TIMEOUT_t(P,Q) \stackrel{def}{=} \delta.P + (t).Q$$

permite o início da execução do processo P através de sua comunicação com o ambiente em qualquer instante sobre um intervalo fixado de tempo, mas permite também a execução do processo Q após este tempo, com preempção da primeira possibilidade.

A Semântica do TCCS

A semântica do TCCS apresentada pelos autores é do tipo transicional, e representada por dois conjuntos de regras: um para as ações funcionais e outro para as ações temporais.

Para representar nas regras semânticas quando um processo deve parar retardando sua realização dentro de uma quantidade de tempo particular define-se a função $|\ .\ |_{\mathcal{T}}$.

Formalmente, a função $|\ .\ |_{\mathcal{T}} : TCCS \rightarrow \{0, 1, 2, \cdots, \omega\}$ define o tempo máximo de espera que um processo pode permitir antes de forçar a realização de uma ação (ou "deadlock"). Sua definição, para cada operador do cálculo, é dada por:

$$\begin{array}{ll}
|\ 0\ |_{\mathcal{T}} = 0 & |\ P \oplus Q\ |_{\mathcal{T}} = \max(|\ P\ |_{\mathcal{T}}, |\ Q\ |_{\mathcal{T}}) \\
|\ X\ |_{\mathcal{T}} = 0 & |\ P + Q\ |_{\mathcal{T}} = \min(|\ P\ |_{\mathcal{T}}, |\ Q\ |_{\mathcal{T}}) \\
|\ a.P\ |_{\mathcal{T}} = 0 & |\ P \mid Q\ |_{\mathcal{T}} = \min(|\ P\ |_{\mathcal{T}}, |\ Q\ |_{\mathcal{T}}) \\
|\ (s).P\ |_{\mathcal{T}} = s + |\ P\ |_{\mathcal{T}} & |\ P\backslash a\ |_{\mathcal{T}} = |\ P\ |_{\mathcal{T}} \\
|\ \delta.P\ |_{\mathcal{T}} = \omega & |\ P[S]\ |_{\mathcal{T}} = |\ P\ |_{\mathcal{T}} \\
 & |\ \mu_i \tilde{x}.\tilde{P}\ |_{\mathcal{T}} = |\ P_i\{\mu_i \tilde{x}.\tilde{P}/\tilde{x}\}\ |_{\mathcal{T}}
\end{array}$$

A semântica transicional do TCCS é dada pelas relações: $\rightarrow \subseteq$ TCCS $\times$ Act $\times$ TCCS, e $\rightsquigarrow \subseteq$ TCCS $\times$ $\mathcal{T}$ $\times$ TCCS. As regras operacionais apresentadas pelos autores são transcritas em duas partes (temporal e funcional) na tabela B.1.

B.2.4 Temporal Process Algebra (TPA) [Hennessy 90]

No trabalho de Hennessy e Regan [Hennessy 90] são descritos um modelo algébrico e uma linguagem para descrição de sistemas em que o tempo intervém, mas sem ser este o aspecto dominante. O modelo é uma extensão do CCS no qual foi acrescentado uma noção mínima de tempo.

A idéia central do modelo é a introdução de uma ação σ em uma álgebra de processos do tipo CCS. A execução desta ação σ por um processo indica que ele está ocioso (ou fazendo nada) até o próximo ciclo de um *clock*. Esta ação não só indica a ociosidade, mas também permite representar um retardo (*"delay"*). Um aspecto característico do modelo é que a semântica da sincronização entre processos é baseada na abordagem *"must"*, isto é, se existe a possibilidade de dois processos sincronizarem, então eles sincronizam (como nos modelos discutidos em [Quemada 89 e Bolognesi 90]).

Sintaxe e Semântica Comportamental

A álgebra de processos TPL (Temporal Process Language) é descrita a seguir. A sintaxe abstrata de TPL dada pela seguinte definição BNF:

$$t ::= nil \mid x \mid \sigma.t \mid a.t \mid \tau.t \mid t+t \mid t|t \mid t[S] \mid t\backslash a \mid recx.t$$

Todos os operadores foram tomados diretamente do CCS [Milner 89], exceto a ação σ. *nil* representa um processo terminado ou bloqueado ("deadlocked"), $a \in Act$, conjunto de ações que tem a estrutura $\Lambda \cup \bar{\Lambda}$, onde Λ é um conjunto básico de ações, e $\bar{\Lambda} = \{\bar{a}, a \in \Lambda\}$ ($\bar{\bar{a}} = a$). $\tau \notin Act$, denota uma ação interna (invisível), enquanto $+, |, [S]$ e $\backslash a$ representam o não determinismo, o paralelismo, a re-rotulação por uma função S, e a restrição. σ representa a *ação ociosa ou retardo ("delay")*.

É utilizado μ para representar ações do conjunto de ações completo $Act \cup \{\tau, \sigma\}$, α para ações do conjunto $Act \cup \{\tau\}$, e a para representar ações do conjunto de ações externas Act.

A semântica operacional dos processos é dada em dois blocos. O primeiro define as relações $\xrightarrow{\alpha}$, para cada $\alpha \in Act \cup \{\tau\}$; ou seja, a semântica operacional padrão do CCS, da qual a ação σ não participa. O segundo bloco de regras semânticas define a relação $\xrightarrow{\sigma}$. Seguem os dois blocos de regras semânticas:

- **Regras Semânticas do Cálculo Sem as Ações σ:**

$$\frac{}{\alpha.p \xrightarrow{\alpha} p}$$

$$\frac{p \xrightarrow{\alpha} p'}{p+q \xrightarrow{\alpha} p'} \qquad \frac{q \xrightarrow{\alpha} q'}{p+q \xrightarrow{\alpha} q'}$$

$$\frac{p \xrightarrow{a} p'}{p[S] \xrightarrow{S[a]} p'[S]}$$

Parte Funcional	Parte Temporal
$\frac{-}{a.P \xrightarrow{a} P}$	$\frac{-}{\delta.P \overset{t}{\rightsquigarrow} \delta.P}$
$\frac{P \xrightarrow{a} P'}{\delta.P \xrightarrow{a} P'}$	$\frac{-}{(s+t).P \overset{s}{\rightsquigarrow} (t).P}$
$\frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'}$	$\frac{-}{(t).P \overset{t}{\rightsquigarrow} P}$
$\frac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} Q'}$	$\frac{P \overset{s}{\rightsquigarrow} P'}{(t).P \overset{s+t}{\rightsquigarrow} P'}$
$\frac{P \xrightarrow{a} P'}{P \oplus Q \xrightarrow{a} P'}$	$\frac{P \overset{t}{\rightsquigarrow} P' \ , \ Q \overset{t}{\rightsquigarrow} Q'}{P+Q \overset{t}{\rightsquigarrow} Q'+P'}$
$\frac{Q \xrightarrow{a} Q'}{P \oplus Q \xrightarrow{a} Q'}$	$\frac{P \overset{t}{\rightsquigarrow} P'}{P \oplus Q \overset{t}{\rightsquigarrow} Q'} \quad (\mid Q \mid_{\mathcal{T}} < t)$
$\frac{P \xrightarrow{a} P'}{P \mid Q \xrightarrow{a} P' \mid Q}$	$\frac{Q \overset{t}{\rightsquigarrow} Q'}{P \oplus Q \overset{t}{\rightsquigarrow} Q'} \quad (\mid P \mid_{\mathcal{T}} < t)$
$\frac{Q \xrightarrow{a} Q'}{P \mid Q \xrightarrow{a} P \mid Q'}$	$\frac{P \overset{t}{\rightsquigarrow} P' \ , \ Q \overset{t}{\rightsquigarrow} Q'}{P \oplus Q \overset{t}{\rightsquigarrow} Q' \oplus P'}$
$\frac{P \xrightarrow{a} P' \ , \ Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$	$\frac{P \overset{t}{\rightsquigarrow} P' \ , \ Q \overset{t}{\rightsquigarrow} Q'}{P \mid Q \overset{t}{\rightsquigarrow} Q' \mid P'}$
$\frac{P \xrightarrow{a} P'}{P \backslash L \xrightarrow{a} P' \backslash L} \quad (a, \bar{a} \notin L)$	$\frac{P \overset{t}{\rightsquigarrow} P'}{P \backslash L \overset{t}{\rightsquigarrow} P' \backslash L}$
$\frac{P \xrightarrow{a} P'}{P[S] \xrightarrow{S(a)} P'[S]}$	$\frac{P \overset{t}{\rightsquigarrow} P'}{P[S] \overset{t}{\rightsquigarrow} P'[S]}$
$\frac{P_i\{\mu\tilde{x}.\tilde{P}/\tilde{x}\} \xrightarrow{a} P'}{\mu\tilde{x}.\tilde{P} \xrightarrow{S(a)} P'}$	$\frac{P_i\{\mu\tilde{x}.\tilde{P}/\tilde{x}\} \overset{t}{\rightsquigarrow} P'}{\mu\tilde{x}.\tilde{P} \overset{t}{\rightsquigarrow} P'}$

Tabela B.1: Regras da semântica operacional do TCCS [Moller 89]

$$\frac{t[recx.t/x] \xrightarrow{\alpha} p'}{recx.t \xrightarrow{\alpha} p'} \qquad \frac{p \xrightarrow{\alpha} p' \ , \ b \neq \alpha}{p'\backslash b \xrightarrow{\alpha} p'\backslash b}$$

$$\frac{p \xrightarrow{\alpha} p'}{p|q \xrightarrow{\alpha} p'|q} \qquad \frac{q \xrightarrow{\alpha} q'}{p|q \xrightarrow{\alpha} p|q'}$$

$$\frac{p \xrightarrow{a} p' \ , \ q \xrightarrow{\bar{a}} q'}{p|q \xrightarrow{\tau} p'|q'}$$

- **Regras Semânticas do Cálculo Com as Ações σ:**

1. $$\frac{}{a.p \xrightarrow{\sigma} a.p} \qquad \frac{}{nil \xrightarrow{\sigma} nil}$$

   A primeira regra diz que $a.p$ e nil podem ser retardadas. Isto é razoável, visto que se $a.p$ está em um ambiente onde nenhuma comunicação via a é possível, então ela pode sofrer um retardo até o próximo ciclo de tempo. De modo similar, nil pode ser retardado indefinidamente pois ele nunca pode realizar uma comunicação. Entretanto, $\tau.p$ não pode sofrer retardos pois deve realizar a ação τ antes do próximo ciclo de tempo.

2. $$\frac{}{\sigma.p \xrightarrow{\sigma} p}$$

3. $$\frac{p \xrightarrow{\sigma} p' \ , \ q \xrightarrow{\sigma} q'}{p+q \xrightarrow{\sigma} p'+q'}$$

   Esta terceira regra diz que $p+q$ pode ser retardado, se p e q também o puderem. A passagem do tempo, isto é, a realização de σ não decide a escolha em $p+q$.

4. $$\frac{p \xrightarrow{\sigma} p' \ , \ q \xrightarrow{\sigma} q' \ , \ p|q \not\xrightarrow{\tau}}{p|q \xrightarrow{\sigma} p'|q'}$$

   A quarta regra diz que $p|q$ pode sofrer um retardo se ambos, p e q, o puderem, e se nenhuma comunicação entre p e q for possível.

5. $$\frac{p \xrightarrow{\sigma} p'}{p\backslash a \xrightarrow{\sigma} p'\backslash a} \qquad \frac{t[recx.t] \xrightarrow{\sigma} p'}{recx.t \xrightarrow{\sigma} p'}$$

Comentário: Com respeito ao modelo de Hennessy e Regan pode-se comentar que é um modelo assíncrono (como o seu modelo subjacente, o CCS). Inicialmente, tem-se a impressão que a ação σ impõe um certo sincronismo aos processos modelados, mas a presença da ação τ no cálculo de base é dominante. O modelo de tempo do TPL é discreto, implícito e global, tal qual o SCCS de Milner [Milner 83].

B.2.5 CCS com Tempo e Probabilidades [Hansson 90]

Hansson e Jonsson propuseram um trabalho no qual era proposto uma extensão do CCS [Milner 80] que incorpora aspectos do tempo e de probabilidade [Hansson 90]. O modelo, chamado pelos autores de *Timed Probabilistic Calculus of Communicating Systems* (TPCCS), visa a descrição de aspectos do tempo-real, de confiabilidade e de performance em sistemas distribuídos. O tempo e a probabilidade são introduzidos em dois passos: primeiro a probabilidade, e depois o tempo.

A Extensão Probabilística

Em CCS um processo é descrito em termos de sua capacidade de se comunicar com outros processos. A execução paralela é modelada pelo entrelaçamento arbitrário de ações. A escolha entre diferentes alternativas (por exemplo, entre a ou b na expressão $a + b$) é não-determinística, isto é, não se tem a *priori* nenhuma informação sobre qual alternativa deverá ocorrer.

Em TPCCS a probabilidade é introduzida através de um operador de escolha probabilística (*"probabilistic choice"*) que não pode ser atribuído à alternativas que realizam ações de comunicação pois suas ocorrências dependem do ambiente. As escolhas probabilísticas são internas aos processos de modo que são realizadas sem a ação do ambiente. Uma escolha probabilística é definida a partir de uma distribuição de probabilidade sobre um conjunto de potenciais estados sucessores ou processos (ou seja, estados imediatamente alcançáveis com medida de probabilidade não nula).

O nome dado pelos autores ao modelo resultante da incorporação da escolha probabilística é *modelo alternante*. Pois em cada estado existe a possibilidade da escolha ser probabilística ou determinística/ não-determinística, e a realização de uma ou outra é feita de maneira alternada. Considere o exemplo (extraído de [Hansson 90] de um processo TPCCS representando uma máquina de venda da figura B.3, onde"•" denota estados determinística/ não-determinísticos, "o" denota estados probabilísticos, as transições determinística/ não-determinísticas são rotuladas com ações e as transições probabilísticas são rotuladas com probabilidades. Intuitivamente, a máquina de venda aceita uma moeda e então, com probabilidade 0.9 oferece chá e café (dando a possibilidade de escolha ao usuário), e com probabilidade 0.1 a máquina recolherá a moeda e não oferecerá nada.

A Extensão Temporal

O tempo em TPCCS é global e discreto, e a sua incorporação no modelo objetiva capturar os aspectos temporais de um conjunto de processos comunicantes se executando simultaneamente. As ações são instantâneas. A passagem do tempo pode ocorrer de duas maneiras: quando um "delay" é especificado explicitamente; ou quando nenhuma ação é possível. Um processo que não pode realizar nenhuma ação interna ou de comunicação pode sempre permitir a passagem do tempo. Por outro lado, o tempo nunca pode avançar quando uma ação interna está habilitada. Com estas características, o modelo é referido como sendo de *espera arbitrária* e *retardo mínimo*. Ou seja, não existe um limite superior de tempo que um processo possa esperar por uma comunicação (*espera arbitrária*), mas ao mesmo tempo,

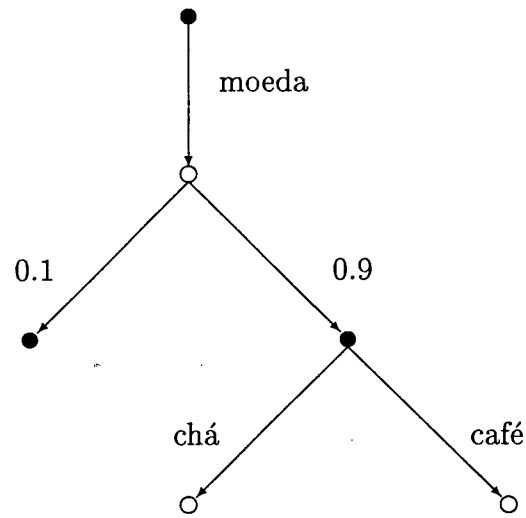


Figura B.3: Uma máquina de venda não confiável

uma vez que uma ação interna seja possível o processo não pode mais ser retardado (*retardo mínimo*).

O modelo utiliza uma ação especial χ para representar a passagem do tempo (os "ticks" do relógio global). Os autores definem um operador especial para tratar características temporais, chamado *timeout* e representado por $\rhd$. Intuitivamente, $N_1 \rhd_i N_2$ denota um processo que após i unidades de tempo torna-se N_2, a menos que N_1 realize qualquer ação antes disso. Se N_1 puder realizar alguma ação (dentro de i unidades de tempo) e tornar-se N_1', então $N_1 \rhd_i N_2$ pode realizar a mesma ação e tornar-se N_1'.

O tratador de *timeout* da figura B.4 ilustra este procedimento. Ele deverá gerar um *timeout* $\overline{to}$ depois de duas unidades de tempo ("ticks") a menos que ele seja abortado antes do segundo "tick".

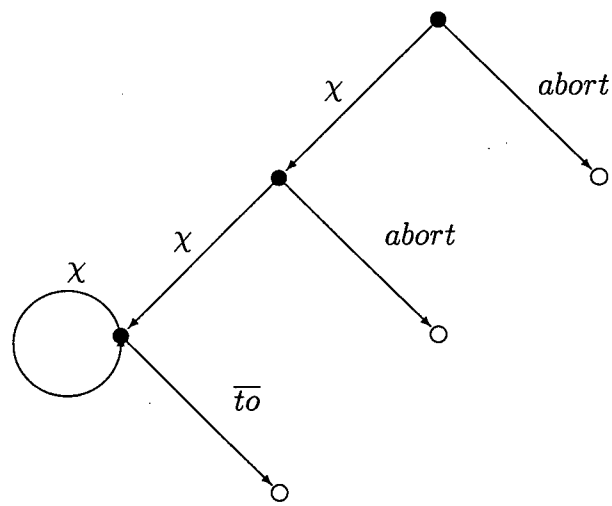


Figura B.4: Comportamento de um tratador de *timeout*: *abort* $\rhd_2 \overline{to}$

A hipótese do retardo mínimo corresponde a um modelo de execução onde os proces-

sadores nunca sofrem restrições de recursos. Desta maneira, os autores evitam problemas de escalonamentos que ocorrem quando é assumido que os sistemas compartilham um número limitado de processadores.

Um outro exemplo da utilização do operador *timeout* ilustra a modelagem de construtores da linguagem de programação Ada através do cálculo TPCCS. O trecho de programa Ada abaixo:

```
select     accept e1;
       or  accept e2;
       or  delay d; accept e3;
end.select;
```

pode ser modelado em TPCCS como:

$$(e_1 + e_2) \rhd_d e_3.$$

O Cálculo

O cálculo TPCCS é definido em duas partes. Inicialmente é definido um cálculo probabilístico pela adição do operador de escolha probabilística, e pela distinção entre as expressões de processos determinísticas/ não-determinísticas e as expressões de processos probabilísticas. Em seguida, os autores extendem este cálculo através da inclusão do operador *timeout*.

Seja Λ um conjunto de símbolos de ação ($\chi, \tau \notin \Lambda$), e seja $\overline{\Lambda}$ o conjunto de símbolos de ações complementares ($\bar{a} \in \overline{\Lambda}$ *sse* $a \in \Lambda, \bar{\bar{a}} = a$). Sejam as letras a, b etc representando os elementos de $Act = \Lambda \cup \overline{\Lambda}$. Sejam as letras gregas α, β etc representando elementos de $Act\tau = Act \cup \{\tau\}$.

Sejam o intervalo fechado $[0,1] \subset \Re$ representando o conjunto de probabilidades e as letras $q, r, q_1, \cdots$ representando valores deste conjunto. Assuma que: X representam processos; f representam funções de re-rotulação; e I representam conjuntos de índices.

A sintaxe é definida para dois tipos de expressões de processos: o conjunto E_N de expressões de processos não-determinísticos cujos elementos são representados pelas letras $N, N_1, \cdots$; e o conjunto E_P de expressões de processos probabilísticos cujos elementos são representados por $P, Q, P_1, \cdots$.

A sintaxe das expressões de processos probabilísticos é dada por:

$$P ::= X \mid \sum_{i \in I} [q_i] N_i \mid P|P \mid P\backslash a \mid P[f] \mid fix.X.P$$

$\sum$ é o operador de soma probabilística, e os outros são os operadores padrões do CCS. Exige-se que na expressão $\sum_{i \in I} [q_i] N_i$ tenha-se $\sum_{i \in I} q_i = 1$

A sintaxe das expressões de processos não-determinísticos é dada por:

$$P ::= X \mid \sum_{i \in I} [\alpha_i] P_i \mid N|N \mid N\backslash a \mid N[f] \mid fix.X.N$$

Semântica Operacional

- **Parte Não-Determinística** $\rightarrow \subseteq E_N \times Act\tau \times E_P$ é a relação de transição não-determinística, e uma tripla em $\rightarrow$ é escrita como $N \xrightarrow{\alpha} P$.

  1. *Choice:*

$$\frac{-}{\sum_{i \in I} \alpha_i P_i \xrightarrow{\alpha_j} P_j} \; j \in I$$

  2. *Parallel Composition:*

$$\frac{N_1 \xrightarrow{a} P_1 \;,\; N_2 \xrightarrow{\bar{a}} P_2}{N_1 | N_2 \xrightarrow{\tau} P_1 | P_2} \qquad \frac{N_1 \xrightarrow{\alpha} P}{N_1 | N_2 \xrightarrow{\alpha} P | [1] N_2} \qquad \frac{N_2 \xrightarrow{\alpha} P}{N_1 | N_2 \xrightarrow{\alpha} [1] N_1 | P}$$

  3. *Restriction:*

$$\frac{N \xrightarrow{\alpha} P}{N \backslash a \xrightarrow{\alpha} P \backslash a} \; \alpha, \bar{\alpha} \neq a$$

  4. *Relabeling:*

$$\frac{N \xrightarrow{\alpha} P}{N[f] \xrightarrow{f(\alpha)} P[f]}$$

  5. *Fix:*

$$\frac{N\{fix.X.N/X\} \xrightarrow{\alpha} P}{fix.X.N \xrightarrow{\alpha} P}$$

- **Parte Probabilística** $\longmapsto \subseteq E_P \times [0,1] \times E_N$ é a relação de transição probabilística, e uma tripla em $\longmapsto$ é escrita como $N \xmapsto{q} P$.

  1. *P-Choice:*

$$\frac{-}{\sum_{i \in I} [q_i] N_i \xmapsto{q_i} N_i}$$

  2. *P-Parallel Composition:*

$$\frac{P_1 \xmapsto{q} N_1 \;,\; P_2 \xmapsto{r} N_2}{P_1 | P_2 \xmapsto{q * r} N_1 | N_2}$$

  3. *P-Restriction:*

$$\frac{P \xmapsto{q} N}{P \backslash a \xmapsto{q} N \backslash a}$$

  4. *P-Relabeling:*

$$\frac{P \xmapsto{q} N}{P[f] \xmapsto{q} N[f]}$$

  5. *P-Fix:*

$$\frac{P\{fix.X.P/X\} \xmapsto{q} N}{fix.X.P \xmapsto{q} P}$$

- **Parte Temporal** O tempo é introduzido no modelo em duas etapas: primeiro, são definidos o operador *timeout*, $\rhd$, e a ação especial de "tick", χ; as 7 regras semânticas que definem estes operadores referem-se à semântica de *espera arbitrária*. Por fim, as regras que caracterizam a semântica do *retardo mínimo* são definidas através da relação $\hookrightarrow$.

  O operador de *timeout*, indroduzido anteriormente, é generalizado da forma seguinte:

$$N \rhd_n N' \equiv N \rhd (N \rhd_{n-1} N') \ (n > 1)$$

$$N \rhd_1 N' \equiv N \rhd N'$$

  Seguem as regras que definem a semântica de *espera arbitrária*:

  1. *Choice-Idle:*

$$\frac{-}{\sum_{i\in I} \alpha_i P_i \xrightarrow{\chi} \sum_{i\in I} \alpha_i P_i}$$

  2. *Parallel Composition-Tick:*

$$\frac{N_1 \xrightarrow{\chi} N_1' \ , \ N_2 \xrightarrow{\chi} N_2'}{N_1 | N_2 \xrightarrow{\chi} N_1' | P_2'}$$

  3. *Restriction-Tick:*

$$\frac{N \xrightarrow{\chi} N'}{N\backslash a \xrightarrow{\chi} N'\backslash a}$$

  4. *Relabeling-Tick:*

$$\frac{N \xrightarrow{\chi} N'}{N[f] \xrightarrow{\chi} N'[f]}$$

  5. *Fix-Tick:*

$$\frac{N_1\{fix.X.N_1/X\} \xrightarrow{\chi} N_2}{fix.X.N_1 \xrightarrow{\chi} N_2}$$

  6. *to-Tick:*

$$\frac{-}{N_1 \rhd N_2 \xrightarrow{\chi} N_2}$$

  7. *to-Action:*

$$\frac{N_1 \xrightarrow{\alpha} P}{N_1 \rhd N_2 \xrightarrow{\alpha} N_2}$$

  Definição da Semântica do Retardo Mínimo As regras semânticas abaixo forçam o retardo mínimo através da eliminação da possibilidade dos processos realizarem "ticks" em estados onde ações τ são possíveis. A relação $\hookrightarrow$ é definida com este objetivo, ela é um subconjunto da relação $\rightarrow$ em que foram excluídas apenas as transições de "tick" originadas de estados oferecendo transições τ. A definição formal de $\hookrightarrow$ é dada abaixo:

$$\frac{N \xrightarrow{\alpha} P}{N \overset{\alpha}{\hookrightarrow} P} \qquad \frac{N \xrightarrow{\alpha} N' \ , \ N \not\xrightarrow{\tau}}{N \overset{\chi}{\hookrightarrow} N'}$$

Comentário sobre a extensão probabilística: Permite associar a ações, valores constantes de probabilidade e tempos mas não densidade de probabilidade a intervalos de tempo, o que é necessário para representar mais precisamente a característica aleatória dos fenômenos físicos. Entretanto, esta extensão probabilística permite a representação de escolhas com probabilidades associadas à alternativas o que possibilita a descrição uma estatística de utilização de recursos necessária para realização de análise de desempenho.

Apêndice C

Uma Introdução Rápida a LOTOS

A linguagem LOTOS, *"Language of Temporal Ordering Specifications* [ISO 88], é constituída de duas compontes [Logrippo 90]: uma componente *tipos de dados*, que é baseada na linguagem ACT ONE, baseada em Tipos de Dados Abstratos Algébricos; e uma componente *controle*, a qual é baseada numa mistura do CCS [Milner 80] e CSP [Hoare 85]. A maior parte da abordagem teórica da componente controle, e especialmente o conceito de ação interna (ou ação invisível) são baseados no modelo CCS. No entanto, a semântica dada à sincronização entre processos é baseada no conceito de sincronização múltipla (*"multi-way rendezvous"*) do modelo CSP, no qual todos os processos que compartilham uma mesma ação (*gate* em LOTOS) devem participar do "*rendezvous*" naquela ação (a notação para representação de interações é similar à utilizada em CSP).

A exemplo do modelo CCS, a semântica formal da parte de controle de LOTOS é dada em termos do modelo semântico transicional (operacional *à la Plotkin*), e os operadores foram escolhidos de maneira que é possível provar que eles satisfazem algumas interessantes propriedades algébricas como em [Milner 80, Milner 89].

Utiliza-se na literatura o termo LOTOS-básico (*basic-LOTOS*) para fazer referência apenas à parte de controle de LOTOS, isto é, sem referência a dados. Como o nosso interesse maior é pela componente de controle de LOTOS, passaremos a apresentar, resumidamente, a sintaxe e o significado informal dos principais operadores que permitem representar as expressões de comportamento do LOTOS-básico [Bolognesi 87]. Os símbolos 'B', 'B1', 'B2' que aparecem a seguir representam expressões de comportamento.

Inação: Representado pela palavra "**stop**", denota um processo completamente inativo.

Prefixação de ações: Existem duas formas:

- "i;B" que representa o oferecimento da ação interna "i", seguido do comportamento B; e
- "g;B" que representa o oferecimento de uma ação ("gate") "g", seguido do comportamento B.

Escolha: Representado por "B1[]B2", denota um processo que se comporta ou como B1 ou como B2 (é similar ao operador de escolha "+" do CCS).

Composição paralela: Existem três casos:

- "B1|[$g_1, \cdots, g_n$|]B2" que representa a sincronização dos processos que oferecem ações em $\{g_1, \cdots, g_n\}$;
- "B1 ||| B2" que representa o entrelaçamento (*"intereaving"*) das ações de B1 e B2; e
- "B1 || B2" que representa a sincronização completa, isto é o conjunto de ações para sincronização é o conjunto de todas as ações possíveis.

Ocultação ("Hiding"): Denotado por "hide $g_1, \cdots, g_n$ in B", este operador tem o efeito de transformar ações observáveis de um processo em ações não observáveis.

Instanciação de processos: Representado por "p[$g_1, \cdots, g_n$]", denota uma instanciação do processo p.

Terminação com sucesso: Representada por "**exit**", esta ação denota o fim bem sucedido de um processo.

Composição seqüencial: A composição seqüencial de dois processos, B1 seguido de B2, é representada por "B1 $\gg$ B2".

Desabilitação: Representado por "B1 [> B2", este operador representa a possibilidade do processo B1 ser desabilitado pelo processo B2.

Como exemplo de uma especificação LOTOS considere o seguinte exemplo [Bochmann 88]: o processo *exemplo* definido a seguir, usa as ações a, b, x, y para suas interações.

```
process exemplo [a,b,x,y]: noexit :=
   y;( a;segue_a[x,y]
     [] b;segue_b[x,y]
     [] i;segue_c[x,y]
     [] i;segue_d[x,y]
     [] i;x;exemplo[a,b,x,y])
 endprocess
```

O corpo deste processo indica que o processo exemplo deve primeiro executar a interação "y" e então pode executar em "rendezvous" com seu ambiente as interações "a" ou "b", em tais casos ele continua com o comportamento definido por segue_a ou segue_b, respectivamente; ou ele faz uma transição interna, indicada por i. No caso dele escolher a última alternativa, o processo deve realizar a ação x e em seguida iniciar novamente com ação y.

A título de informação adicional, cabe aqui dizer que a técnica básica de provas existente em LOTOS é baseada no conceito de *bissimulação*, o qual é similar ao conceito de congruência observacional do CCS.

Isto termina a breve apresentação da linguagem LOTOS.

Bibliografia

[Abadi 88] Abadi, M., *The Power of Temporal Proofs*, DIGITAL Research Report Number 30, august 15, 1988.

[1] x Alur, R. ; Courcoubetis, C. e Dill, D., *Model Checking for Real-Time Systems*, In Proceedings of the Fifth IEEE Symposium on Logic in Computer Science, 1990.

[Alur 92] Alur, R., M.; 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, pp. 74-106.

[Arnold 90] Arnold, A. *Systèmes de Transitions Finis et Sémantique des Processus Communicants*, Technique et Science Informatiques, Vol. 9, No. 3, 1990.

[Azcorra 90] Azcorra, A.S., *Formal Modeling of Synchronous Systems*, Ph.D. Thesis, Dpto. de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Madrid, Espanha, 1990.

[Audsley 90] Audsley, N.; Bhattacharyya, A.; Burns, A.; Fohler, G.; Kantz, H.; Kopetz, H.; McDermid, J.A.; Schütz, W.; Zainlinger, R., *Timeliness - Summary and Conclusions*, Vol. 2 Specification and Design for Dependability, in First Year Report of the Predictably Dependable Computing Systems ESPRIT Project, Toulouse, 1990.

[Auernheimer 86] Auernheimer, B. Kemmerer, R., *RT-ASLAN: A Specification Language for Real-Time Systems*, IEEE Transaction on Software Engineering, Vol. SE-12, No. 9, pp. 879-889, Setembro, 1986.

[Ayache 85] Ayache, J.M.; Courtiat, J.P.; Diaz, M.; Juanole, G., *Utilisation des Réseaux de Petri pour la Modélisation et Validation de Protocoles*, Technique et Science Informatiques, Vol.4, No. 1, 1985, pp. 51-71.

[Baeten 89] Baeten, J.C.M. and Bergstra,J.A., *Real Time Process Algebra*, Research Report P8916, University of Amsterdam, Programming Research Group, december, 1989.

[Baeten 92] Baeten, J.C.M., *The Total Order Assumption*, In Proc of the Workshop "What good is interleaving?", Sheffield, June, 1992.

[Bakker 92] de Bakker, J.W.; Huizing, C.; de Roever, J.P.; Rozemberg, G. (Eds.), *Real-Time: Theory in Practice*, Proceedings of the REX Workshop, Mook, The Netherlands, June 1991, Lecture Notes in Computer Science No. 600, Springer-Verlag, 1992.

[Belmans 88] Belmans, P.F.R.; Kaitouni, O.D., *Specification et Validation d'un Protocole de Communication Adapté au Temps Réel*, Publication Interne No. 394, IRISA - Rennes, France, Fevereiro, 1988.

[Bergstra 86] Bergstra, J.A. ; Klop, J.W., *Process Algebra: Specification and Verification in Bissimulation Semantics*, Mathematics and Computer Science II, Eds.: Hazewinkel, J.K. et all., North Holland, Amsterdan: CWI Monographs 4, pp. 61-94, 1986.

[Berry 87] Berry, G.; Couronne, P.; Gonthier, G., *Programmation Synchrone des Systèmes Réactifs: le Langage ESTEREL*, Technique et Science Informatiques, Vol. 6, No. 4, 1987, pp. 305-316.

[Berry 88] Berry, G.; Gonthier, G., *The ESTEREL Synchronous Programming Language: Design, Semantics, Implementation* , Rapport de Recherche d'INRIA, Valbone, France, 1988.

[Blair 93] Blair, G.; Blair, L.; Bowman, H. e Chetwynd A., *Formal Support for the Specification and Construction of Distributed Multimedia Systems (The Tempo Project) - Final Project Deliverable*, Internal Report MPG-93-23, Lancaster University, 1993.

[Bochmann 88] Bochmann, G.v. e Vaucher, J., *Adding Performance Aspects to Specification Languages*, Proc. of Protocol Specification, Testing, and Verification VIII - IFIP, Aggarwal, S e Sabnani, K. (eds), North-Holland, pp. 19-31, 1988.

[Bolognesi 87] Bolognesi, T.; Brinksma, Ed, *Introduction to the ISO Specification Language LOTOS*, Computer Networks and ISDN Systems (NORTH-HOLLAND), Num. 14, 1987, pp. 25-59.

[Bolognesi 87b] Bolognesi, T.; Smolka, S. A., *Fundamental Results for the Verification of Observational Equivalence: A Survey*, Protocol Specification, Testing, and Verification, VII, H. Rudin and C.H. West (Eds.), (NORTH-HOLLAND), IFIP, 1987.

[Bolognesi 89] Bolognesi, T.; Cremonese, P., *The Weakness of some Timed Models for Concurrent Systems*, Technical Report C89-29, C.R.N.- CNUCE, Italy, 1989.

[Bolognesi 90] Bolognesi, T.; Lucidi, F.; Trigila, S., *From Timed Petri Nets to Timed LOTOS* , Proceedings of IFIP WG 6.1 Tenth International Conference on Protocol Specification, Testing, and Verification, 1990, pp. 377-406.

[Bolognesi 91] Bolognesi, T.; Lucidi, F., *LOTOS-Like Process Algebras with Urgent or Timed Interactions*, in Proceedings of the Formal Description Techniques IV, Parker, K.R. and Rose, G.A. (Eds.), FORTE'91, North-Holland, 1992.

[Bolognesi 92] Bolognesi, T.; Lucidi, F., *Timed Process Algebras with Urgent Interactions and a Unique Powerful Binary Operator*, Proceedings of the REX Workshop, Mook, The Netherlands, June 1991, Lecture Notes in Computer Science No. 600, Springer-Verlag, 1992, pp. 124-148.

[Bolognesi 93] Bolognesi, T.; Lucidi, F.; Trigila, S., *Converging Towards a Timed-LOTOS Standard*, Research Report, CNUCE/C.R.N., Pisa, Itália, 1993.

[Bougé 88] Bougé, L., *Sémantique du Parallélisme: un Tour d'Horizon*, Rapport de Recherche du L.I.E.N.S. - Laboratoire d'Informatique de l'Ecole Normale Supérieure, Julho, 1988.

[Bouldol 88] Bouldol, G.; Castellani, I., *A Non-Interleaving Semantics for CCS Based on Proved Transitions*, Rapport de Recherche 919, INRIA - Sophia Antipolis, France, Outubro, 1988.

[Brams 83] Brams, G.W., *Réseaux de Petri: Théorie et Pratique*, Ed. Masson, Tomos 1 e 2, Paris, 1983.

[Brinksma 88] Brinksma, Ed, *On Design of Extended LOTOS: A Specification Language for Open Distributed Systems*, PhD Thesis, University of Twente, The Netherlands, 1988.

[Brookes 84] Brookes, S.D.; Hoare, C.A.R. ; Roscoe, A.W., *A Theory of Communicating Sequential Processes*, Journal of the Association for Computing Machinery, Vol. 31, No. 3, Julho, 1984, pp. 560-599.

[Broy 86b] Broy, M., *Denotational Semantics of Communicating Sequential Processes*, Information Processing Letters (The Netherlands), Vol. 23, Num. 5, November, 1986, pp. 253-259.

[Broy 89] Broy, M., *Towards a Formal Foundation of the Specification Language SDL*, Research Report MIP-8923, Fakultät für Mathematik und Informatik, Universität Passau, Passau, West Germany, July, 1989.

[Broy 89b] Broy, M. *Formalization of Distributed, Concurrent, Reactive Systems*, em IFIP - State of the Art Seminar on Formal Description of Programming concepts, Petrópolis, 1989.

[Camargo 90] Camargo, M. S. de, *Réseau de Petri Temporel et Lotos Temporisé: Une Etude de Cas*, Rapport de Recherche No. 90189 du LAAS-CNRS, Toulouse, France, juin, 1990.

[Camargo 91] Camargo, M. S. de, *Especificação e Avaliação de Desempenho em Sistemas Dependentes do Tempo: Uma Proposta de Inclusão do Tempo e das Probabilidades em uma Álgebra de Processos*, Monografia apresentada para exame de qualificação de Doutorado, PGEEL-CTC, Laboratório de Controle e Microinformática, Universidade Federal de Santa Catarina, Setembro, 1991, 104 páginas.

[Camargo 92] Camargo, M. S. de; Farines, J.-M., *Uma Variante do Modelo LOTOS Básico com Tempo Estocástico para Especificação e Avaliação de Desempenho em Sistemas Distribuídos Dependentes do Tempo*, anais do 10 Simpósio Brasileiro de Redes de Computadores, Recife, Abril, 1992, pp. 192-207.

[Camargo 93] Camargo, M.S. de; Farines, J.-M., *A Translação de RT_LOTOS para o Modelo de Grafos Temporizados: uma Abordagem para Verificação de Sistemas Tempo Real*, Relatório Técnico RT 93-31, LCMI-UFSC, Florianópolis, Dezembro, 1993.

[Camargo 94] 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.

[Cardelli 82] Cardelli, L., *Real Time Agents*, Proc. ICALP'82, Lecture Notes in Computer Science, no. 140, Springer Verlag, 1982.

[Carmo 92] Carmo, L.F.R.C and de Saqui, P and Courtiat, J.P. *Basic Synchronization Concepts in Multimedia Systems*, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, November 92.

[Caspi 87] Caspi, P.; Halbwachs, N.; Pilaud, D. and Plaice, J., *Le Langage LUSTRE et sa Sémantique Opérationnelle*, Actes du Deuxième Colloque C-Cube, A. Arnold (ed.), pp. 81-98, may, 1987.

[2] x Cavalli, A.R. e Horn, F, *Proof of Specification Properties by Using Finite State Machines and Temporal Logic*, Em IFIP - Protocol Specification, Testing, and Verification Proceedings, H. Rudin e C.H. West (eds), 1987.

[CCITT 88] CCITT, *Recommendation Z.100: Specification and Description Language SDL*, AP IX-35, 1988.

[CNRS 88] Groupe de Réflexion Temps Réel - CNRS - France, *Le Temps Réel*, Rapport Etabli par le Groupe de Réflexion Temps Réel du Département Sciences Physiques pour l'Ingénieur du CNRS, France, Juin, 1988.

[Coelho da Costa 92] Costa, R.J.C. da. and Courtiat, J.P. *A true concurrency semantics for LOTOS*, In Proceedings of the IFIP TC6/WG6.1 5th Int. Conf. on Formal Description Techniques -FORTE'92, North-Holland, 1992.

[Coelho da Costa 93] Costa, R.J.C. da, *Systèmes de Transitions Etiquetés Causaux: Une Nouvelle Approche pour la Description du Comportement Evénémentiel de Systèmes Concurrents*, Thèse de Docteur de l'Université Paul Sabatier de Toulouse, França, maio, 1993.

[Courtiat 93] 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.

[Courtiat 93b] Courtiat, J.P. and de Camargo, M.S. and Saïdouni, D.E, *RT_LOTOS: A Time Extension of LOTOS for the Specification of Real-Time Systems*, LAAS Report 93158, LAAS/CNRS, Toulouse, França, maio, 1993.

[Crubille 88] Crubille, P., *MEC: un Outil de Calcul sur les Systèmes de Transitions*, Actes du Colloque Francophone sur l'Ingénierie des Protocoles, Bordeaux, France, Septembre, 1988, pp. 267-280.

[Dasarathy 85] Dasarathy, B., *Timing Constraints of Real-Time Systems: Constructs for Expressing Them, Methods of Validating Them*, IEEE Transaction on Software Engineering, Vol. SE-11, No.1, pp 80-86, 1985.

[Davidson 89] Davidson, S.; Lee, I.; Wolfe, V., *Language Constructs for Timed Atomic Commitment*, in Proc. IEEE Real-Time Systems Symposium, December, pp. 470-477, 1989.

[Daws 94] Daws, C. et Olivero, A. et Yovine, S., *Verifying ET-LOTOS programs with KRONOS*, In Proceedings of the FORTE'94, Berne, Outubro, 1994.

[DeNicola 83] DeNicola, R.,; Hennessy, M., *Testing Equivalences for Processes*, Theoretical Computer Science the Journal of the EATCS, Vol. 40, 1985, pp. 83-133.

[Diaz 86] Diaz, M., *Nets in Protocols*, Advanced Course on Petri Nets, Bad Honnef, GMD, Germany, Setembro, 1986.

[Diaz 87] Diaz, M., *Applying Petri Net Based Models in the Design of Systems*, Rapport de Recherche du LAAS-CNRS, Toulouse, France, 1987.

[Diaz 89] Diaz, M., *Environnements Logiciels pour la Conception des Protocoles dans les Systèmes Distribués*, Anais do Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos, Florianópolis, Setembro, 1989, pp. 28-35.

[Diaz 91] Berthomieu, B. ; Diaz, M., *Modeling and Verification of Time Dependent Systems Using Time Petri Nets*, IEEE Transaction on Software Engineering, Vol. 17, No. 3, marco 1991, pp. 259-273.

[Dill 89] Dill, D.L., *Timming Assumptions and Verification of Finite-State Concurrent Systems*, in Automatic Verification Methods for Finite State Systems, J. Sifakis (Ed.), LNCS 407, pags. 196-211, Springer-Verlag, 1989.

[Dodhiawala 89] Dodhiawala, R.; Sridharan, N.S.; Raulefs, P.; Pickering, C., *Real-Time AI Systems: A Definition and an Architecture*, in Proc. of the IJCAI-89, 1989.

[Dzierzgowski 90] Dzierzgowski, D., *Quatre Exemples de Langages ou Environnements pour le Développement de Programmes où le Temps Intervient*, Technique et Science Informatiques, Vol. 9, No. 4, pp. 289-312, 1990.

[Elloy 88] Elloy, J.P., *Le Temps Réel*, Technique et Science Informatiques, Vol. 7, No. 5, pp. 493-500, 1988.

[Farines 93] Farines, J.-M., *Questões de Tempo Real nos Sistemas Informáticos e Aspectos de sua Modelização*, Monografia apresentada para fins de concurso público para professor titular da Universidade Federal de Santa Catarina, 1993.

[Faulk 88] Faulk, S.; Parnas, D.L., *On Synchronization in Hard-Real-Time Systems*, Communications of the ACM, Vol. 31, No. 3, março, 1988, pp. 274-287.

[Ferrari 90] Ferrari, D., *Client Requirements for Real-Time Communication Services*, Communications Magazine, novembro, 1990, pp.65-72.

[Fournier 91] Fournier, R.; Bochmann, G. von, *The Equivalence in the DCP Model*, Theoretical Computer Science, Vol. 97, 1991, pp. 97-114.

[Garavel 89] Garavel, H., *Compilation et Vérification de Programmes LOTOS*, Thèse de Docteur de l'Université Joseph Fourier - Grenoble I, França, Novembro, 1989.

[Garavel 89b] Garavel, H.; Najn, E., *TILT: From LOTOS to Labelled Transition Systems* in The Formal Description Technique LOTOS, Eijk, P.H.J.V; Vissers, C.A,; Diaz, M. (Ed.), Elsevier Science Publishers B.V., 1989.

[Ghezzi 89] Ghezzi, C.; Mandrioli, D.; Morasca, S.; Pezze, M., *A General Way to Put Time in Petri Nets*, ACM SIGSOFT Eng. Notes, Vol. 14, No. 3, may, pp. 60-67, 1989.

[Giacalone 90] Giacalone, A. ; Jou, C.-C. ; Smolka, S.A., *Algebraic Reasoning for Probabilistic Concurrent Systems*, Proc. IFIP Conf. on Programming Concepts and Methods, Israel, Abril 1990.

[Glabbeek 89] Glabbeek, R. van; Goltz, U., *Equivalence Notions for Concurrent Systems and Refinement of Actions*, Arbeitspapiere der GDM 366, Gesellschaft für Mathematik und Datenverarbeitung MBH, February, 1989.

[Glabbeek 89] Glabbeek, R.van ; Smolka, S.A. ; Steffen, B. ; Tofts, C.M.N., *Reactive, Generative, and Stratified Models of Probabilistic Processes*, Proc. of 5th IEEE LiCS, Philadelphia, Junho 1990.

[Godskesen 92] Godskesen, J.C. ; Larsen, K. *Real-Time and Expansion Theorems*, in Proc. of NAPAW'92 - First North American Process Algebra Workshop, Stony Brook, 1992.

[Gordon 79] Gordon, M.J.C, *The Denotational Description of Programming Languages*, Springer-Verlag, 1979.

[Goswami 88] Goswami, A. ; Joseph, M., *Semantics of Real-Time Distributed Programs*, LNSC 335, Vogt, F.H.(Ed.) CONCURRENCY 88, International Conference on Concurrency, Hamburg FRG, October, 1988.

[Groote 90] Groote, J.F., *Transition System Specification with Negative Premisses*, in J.C.M. Baeten, J.W. Klop, eds., CONCUR'90, Theories of Concurrency: Unification and Extension, LNCS 458, Springer-Verlag, Berlin, 1990, pp. 332-341.

[Gunter 89] Gunter, C.A.; Mosses, P.D. e Scott, D.S., *Semantic Domains and Denotational Semantics*, Research Report DAIMI PB-276, Computer Science Department, Aarhus University, abril, 1989.

[Halbwachs 89] Halbwachs, N; Pilaud, D. Ouabdesselam, L., *Specifying, Programming and Verifying Real-Time Systems Using a Synchronous Declerative Language*, Automatic Verification Methods for Finite State Systems, J. Sifakis (ed.), LNCS 407, pp. 213-231, june, 1989.

[Hansson 90] Hansson, H.; Jonsson, B., *A Calculus for Communicating Systems with Time and Probabilities*, Proc. of the Real-Time Systems Symposium, December, 1990, pp. 278-287.

[Hansson 90b] Hansson, H.; Jonsson, B., *A Logic for Reasoning about Time and Reliability*, FORTE'90, Madrid, Spain, 1990.

[Hansson 91] Hansson, H., *Time and Probability in Formal Design of Distributed Systems*, Ph.D. Thesis, Department of Computer Systems, Uppsala University, Uppsala, SCIS - Swedish Institute of Computer Science, Kista, Suécia, 1991.

[Hennessy 88] Hennessy, M., *Algebraic Theory of Processes*, MIT press series in foundations of computing, Massachusetts Institute of Technology, 1988.

[Hennessy 88b] Hennessy, M., *Observing Processes*, Thechnical Report 5/88, Computer Science, University of Sussex, Dezembro 1988.

[Hennessy 90] Hennessy, M.; Regan, T., *A Temporal Process Algebra*, FORTE'90, Madrid, Spain, 1990.

[Hoare 85] Hoare, C.A.R., *Communications Sequential Processes*, Prentice-Hall International, 1985

[Holliday 87] Holliday, M.A. ; Vernon, M.K., *A Generalized Timed Petri Net Model for Performance Analysis*, IEEE Transaction on Software Engineering, Vol. SE-13, No. 12, December 1987, pp. 1297-1310.

[Hooman 89] Hooman, J.J.M.; Roever, W.P. de, *Design and Verification in Real-Time Distributed Computing: an Introduction to Compositional Methods*, Ninth International Symposium on Protocol Specification, Testing, and Verification, Enschede, The Netherlands, June, 1989.

[Hulzen 89] Hulzen, W.H.P.; Tilanus, P.A.J.; Zuidweg, H., *LOTOS Extended with Clocks*, in proceedings of the FORTE 89, Vancouver, Canada, 1989.

[ISO 87] ISO, *Estelle - A Formal Description Technique Based on an Extended State Transition Model*, ISO/TC97/WG1/DIS9074, 1987.

[ISO 88] ISO, *LOTOS: A Formal Description Technique Based on the Ordering of Observational Behaviour*, ISO DP 8807, 1988.

[Jaffe 91] Jaffe, M.S. ; Leveson, N.G. ; Heimdahl, M.P.E. ; Melhart, B.E., *Software Requirements Analysis for Real-Time Process-Control Systems* IEEE Transaction on Software Engineering, Vol. 17, No. 3, marco 1991, pp. 241-258.

[Jahanian 86] Jahanian, F. and Mok, A.K., *Safety Analysis of Timing Properties in Real-Time Systems*, IEEE Transactions on Software Engineering, Vol SE-12, N0. 9, pp. 890-904, Setembro, 1986.

[Joseph 88] Joseph, M.; Goswami, A., *Formal Description of Real-Time Systems: a Review*, Research Report 129, Depatment of Computer Science - University of Warwick - UK, Agosto, 1988.

[Joseph 89] Joseph, M., *Time and Real-Time in Programs*, LNSC 405, Springer-Verlag, 1989, pp. 313-324.

[Joseph 89b] Joseph, M.; Goswami, A., *Relating Computation and Time*, in Proceedings of the Joint University of Newcastle Upon Tyne / International Computers Limited Seminar, B. Randell (ed.), Setembro, 1989.

[Joseph 92] Joseph, M., *Problems, Promisses and Performance: Some Questions for Real-Time Systems Specification*, Proceedings of the REX Workshop, Mook, The Netherlands, June 1991, Lecture Notes in Computer Science No. 600, Springer-Verlag, 1992, pp. 315-324.

[Kopetz 89] Kopetz, H., *Real-Time Systems*, Research Repport Nr. 12/89, Institut fur Technische Informatik, Technische Universität Wien, september, 1989.

[Kopetz 92] Kopetz, H., *Sparse Time versus Dense Time in Distributed Real-Time Systems*, In proc. of the 12th International Conference on Distributed Computing Systems, Yokohama, Japão, Junho, 1992, pp. 460-467.

[Koymans 88] Koymans, R.; Shimasundar, R.K.; Roever, W.P. de; Gerth, R.; Arun-Kumar, S., *Compositional Semantics for Real-Time Distributed Computing*, Information and Computation, Vol. 79, Num. 3, December, 1988.

[Koymans 90] Koymans, R., *Specifying real-time propreties with metric temporal logic*, Real-Time Systems, 2(4), pp. 255-299, Novenbro, 1990.

[Lamport 78] Lamport, L., *Time, Clocks, and the Ordering of Events in a Distributed System*, Communications of the ACM, Vol. 21, No. 7, July, 1978, pp. 558-565.

[Lamport 86] Lamport, L., *The Mutual Exclusion Problem: Part I - A Theory of Interprocess Communication*, Journal of the Association for Computing Machinery, Vol. 33, Num. 2, April, 1986, pp. 313-326.

[Leduc 91] Leduc, G. ; Léonard, L., *An Upward Compatible Timed Extension for LOTOS*, In Proceedings of the IFIP TC6/WG6.1 4th Int. Conf. on Formal Description Techniques -FORTE'91, 1991.

[Leduc 92] Leduc, G. ; Léonard, L., *A Timed LOTOS Supporting a Dense Time Domain and Including New Timed Operators*, In Proceedings of the IFIP TC6/WG6.1 5th Int. Conf. on Formal Description Techniques -FORTE'92, 1992.

[Leduc 93] Leduc, G. e Léonard, L. *Comment Rendre LOTOS Apte à Spécifier des Systèmes Temps Réel?*, actes du congrés CFIP'93, Montréal, Canadá, Setembro 1993.

[Leduc 93a] Leduc, G. ; Léonard, L., *An Enhanced Version of Timed LOTOS and its Application to a Case Study*, In Formal Description Techniques VI, (FORTE 93), Teney, R.; Amer, P.; Uyar, Ü. (Editors), North-Holland, 1994.

[Leduc 94] Leduc, G. ; Léonard, L., *A Formal Definition of Time in LOTOS*, Research Report, Université de Liège, Institut d'Electricité Montefiori, Bélgica, 1994.

[Lee 85] Lee, I. and Gehlot, V., *Language Constructs for Distributed Real-Time Programming*, 1985.

[Levi 90] Levi, S.-T.; Agrawala, A.K., *Real-Time System Design*, McGraw-Hill, Inc., 1990.

[Lin 88] Lin, F.J. e Liu, M., *An Integrated Approach to verification and Performance Analysis of Communication Protocols*, Proc. of Protocol Specification, Testing, and Verification VIII - IFIP, Aggarwal, S e Sabnani, K. (eds), North-Holland, pp. 125-140, 1988.

[Logrippo 88] Logrippo, L.; Obaid, A., *Outils Logiciels pour le Langage LOTOS*, Actes du Colloque Francophone sur l'Ingénierie des Protocoles, Bordeaux, France, Septembre, 1988, pp. 225-226.

[Marsan 94] Marsan, M.A. et ali., *A LOTOS Extension for the Performance Analysis of Distributed Systems*, IEEE/ACM Transactions on Networking, Vol.2, No.2, Abril, 1994, pp. 151-165.

[Marshall 89] Marshall, A.K., *Introduction to LOTOS Tools*, The Formal Description Technique LOTOS: Results of the ESPRIT/SEDOS Project, Eijk, P.H.J.V; Vissers, C.A,; Diaz, M. (Ed.), Elsevier Science Publishers B.V., 1989, pp. 339-350.

[Merlin 76] Merlin, P. and Farber, D.J., *Recoverability of Communications Protocols - Implications of a Theoretical Study"*, IEEE Transaction Communication, Vol. COM-24, pp. 1036-1043, Setembro 1976.

[Milne 85] Milne, G.J., *CIRCAL and the Representation of Communication, Concurrency, and Time*, ACM Transactions on Programming Languages and Systems, Vol. 7, No. 2, April, 1985, pp. 270-298.

[Milner 80] Milner, R., *A Calculus of Communicating Systems*, Lecture and Notes in Comp. Science, No. 92, Springer Verlag, 1980.

[Milner 83] Milner, R., *Calculi for Synchrony and Asynchrony*, Theoretical Computer Science, 25, pp. 267-310, 1983.

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

[Milner 91] Milner, R., *The Polyadic π-Calculus: A Tutorial*, Technical Report ECS-LFCS-91-180, LFCS, University of Edinburgh, 1991.

[Moitra 89] Moitra, A. and Joseph, M., *Implementing Real-Time Systems by Transformation*, in Proceedings of the Joint University of Newcastle Upon Tyne / International Computers Limited Seminar, B. Randell (ed.), Setembro, 1989.

[Mok 89] Mok, A.K., *Real-Time Logic, Programming and Scheduling*, in Proceedings of the Joint University of Newcastle Upon Tyne/ International Computers Limited Seminar, B. Randell (ed.), Setembro, 1989.

[Moller 89] Moller, F.; Tofts, C., *A Temporal Calculus of Communicating Systems*, Research Report ECS-LFCS-89-104, University of Edimburgh, 1989.

[Motus 84] Motus, L.; Lorents, P., *Specification of Distributed Rea-Time Systems*, Institute of Cybernetics, Computer Division, Tallinn, Estonia, USRR, 1984.

[Motus 92] Motus, L., *Time Concepts in Real-Time Software*, on Proceedings of International Workshop on Real-Time Programming WRTP'92, Bélgica, Junho, 1992.

[Nicolaou 90] Nicolaou, C., *An Archicteture for Real-Time Multimedia Communications Systems*, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, Abril, 1990, pp. 391-400.

[Nicollin 90] Nicollin, X.; Richier, J.-L.; Sifakis, J.; Voiron, J., *ATP: An Algebra for Timed Processes*, Research Report RT-C16, IMAG, Université Joseph Fourier, Janeiro, 1990.

[Nicollin 92] Nicollin, X.; Sifakis, J., *An Overview and Syntesis on Timed Process Algebras*, Proceedings of the REX Workshop, Mook, The Netherlands, June 1991, Lecture Notes in Computer Science No. 600, Springer-Verlag, 1992, pp. 526-548.

[Nicollin 92b] Nicollin, X.; Sifakis, J.; Yovine, S., *From ATP to Timed Graphs and Hibrid Systems*, Proceedings of the REX Workshop, Mook, The Netherlands, June 1991, Lecture Notes in Computer Science No. 600, Springer-Verlag, 1992, pp. 549-572.

[Nielsen 86] Nielsen, M., *CCS - and its Relationship to the Net Theory*, Petri Nets: Applications and Relationship to the other Models of Concurrency, Lecture and Notes in Computer Science 255 (Springer-Verlag), 1986.

[Olderog 86] Olderog, E.R., *TCSP: Theory of Communicating Sequential Processes*, Petri Nets: Applications and Relationships to the Other Models of Concurrency, LectureEand Notes in Computer Science 255, (Springer-Verlag), 1986.

[Ostroff 89] Ostroff, J.S., *Temporal Logic for Real-Time Systems*, Research Studies Press LTD, 1989.

[Ostroff 89b] Ostroff, J.S., *Automated Verification of Timed Transition Models*, LNCS 407, Proceedings of International Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France, 1989.

[Ostroff 92] Ostroff, J.S., *Formal Methods for the Specification and Design of Real-Time Safety Critical Systems*, Journal Systems Software, Vol. 18, 1992, pp. 33-60.

[Plotkin 81] Plotkin, G.D., *A Structural Approach to Operational Semantics*, Report DAIMI-FN19, Computer Science Dept., Århus University, Dinamarca, 1981.

[Pnueli 81] Pnueli, A., *The Temporal Semantics of Concurrent Programs*, Theoretical Computer Science, Vol. 13, 1981, pp. 45-60.

[Quemada 87] Quemada, J.; Fernandes, A., *Introduction of Quantitative Relative Time into LOTOS*, Em IFIP - Protocol Specification, Testing, and Verification Proceedings, H. Rudin e C.H. West (eds), 1987.

[Quemada 89] Quemada, J.; Azcorra, A. and Frutos, D., *A Timed Calculus for LOTOS*, in proceedings of the FORTE 89, Vancouver, Canada, 1989.

[Ramadge 87] Ramadge, P.J. e W.M. Wonham, *Supervisory control of a class of discrete event processes*, SIAM Journal of Control and Optimization, 25(1), pp. 206-230, Setembro, 1987.

[Raynal 91] Raynal, M., *La Communication et le Temps dans les Réseaux et les Systèmes Répartis*, Ed. Eyrolles, 1991.

[Razouk 89] Razouk, R.R.; Gorlick, M.M., *A Real-Time Interval Logic for Reasoning About Executions of Real-Time Programs*, ACM Software Engineering Notes, Vol. 14, No.8, pp. 10-19, (TAV3), december, 1989.

[Reed 88] Reed G.M.; Roscoe, A.W., *Metric Spaces as Models for Real-Time Concurrency*, Proc. of the Third Workshop on the Mathematical Foundations of Programming Language Semantics, LNCS 298, 1988.

[Reed 89] Reed, G.M., *Mathematical Fundations of Real-time Distributed Computing*, in Proceedings of the Joint University of Newcastle Upon Tyne/ International Computers Limited Seminar, B. Randell (ed.), Setembro, 1989.

[Regan 93] 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), pp. 127-143, 1993.

[Richier 87] Richier, J.L.; Sifakis, J.; Voiron, J., *Une Algèbre de Processus Temporisés*, Actes du Deuxième Colloque C-Cube, A. Arnold (ed.), pp. 359-379, maio, 1987.

[Rico 91] Rico, N. ; Bochmann, G.v. *Performance Description and Analysis for Distributed Systems Using a Variant of LOTOS*, Proc. of Protocol Specification, Testing, and Verification X - IFIP, Suécia, Junho 1991.

[Rudin 85] Rudin, H., *Time in Formal Protocol Specifications*, IBM - Report Number: RZ 1349, 1985.

[Rudin 86b] Rudin, H., *The Dimension of Time in Protocol Specifications*, LNSC 248, Networking in Open Systems - International Seminar Austria, Agosto, 1986.

[Rudin 88] Rudin, H., *Protocol Engineering: A Critical Assessment*, Proc. of Protocol Specification, Testing, and Verification VII - IFIP, Aggarwal, S e Sabnani, K. (eds), North-Holland, 1988, pp. 3-16.

[Scholefield 90] Scholefield, D., *The Formal Development of Real-Time Programs: A Short Review*, Research Report - Department of Computer Science - University of York, England , Setembro, 1990.

[Shankar 87] Shankar, U. and Lam, S.S., *Time-Dependent Distributed Systems: Proving Safety, Liveness and Real-Time Properties*, Distributed Computing, Vol. 2, pp.61-79, 1987.

[Shaw 89] Shaw, A.C., *Reasoning About Time in Higher-Level Language*, IEEE Trans. on Software Engineering, Vol. 15, No. 7, July, 1989.

[Shyamasundar 89] Shyamasundar, R.K.; Hooman, J. and Gerth, R., *Reasoning of Real-Time Distributed Programming Languages*, ACM SIGSOFT Eng. Notes, Vol. 14, No. 3, pp. 91-99, may, 1989.

[Stankovic 88] Stankovic, J.A., *Misconceptions About Real-Time Computing: A Serious Problem for Next-Neneration Systems*, IEEE Computer, October, 1988, pags 10-19.

[Stankovic 88b] Stankovic, J.A., *Real-Time Computing Systems: The Next Generation*, COINS Techninical Report 88-06, University of Massachusets at Amhest, january, 1988.

[Steinmetz 90] Steinmetz, R., *Synchronization Properties in Multimedia Systems*, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, Abril, 1990, pp. 401-412.

[Stoyenko 93] Stoyenko, A.D.; Halang, W.A., *Extending PEARL for Industrial Real-Time Applications*, IEEE Software, Vlo. 10, No. 8, Julho, 1993, pp. 65-74.

[Turski 89] Turski, W.M., *Timing Considerations Will Damage your Programs or How To Trade Clocks for Chips*, in Proceedings of the Joint University of Newcastle Upon Tyne/ International Computers Limited Seminar, B. Randell (ed.), Setembro, 1989.

[Vissers 88b] Vissers, C.A., *FDTs for Distributed Systems*, Research into Networks and Distributed Applications, R. Speth (Editor) ECSC, EEC, EAEC, Elsevier Science Publishers B.V. (NORTH-HOLLAND), 1988.

[Vissers 89] Vissers, C.A., *LOTOS Backgrounds*, The Formal Description Technique LOTOS: Results of the ESPRIT/SEDOS Project, Eijk, P.H.J.V.; Vissers, C.A,; Diaz, M.(Editores), Elsevier Science Publishers B.V., 1989, pp. 15-22.

[Winkowski 87] Winkowski, J.; Maggiolo-Schettini, A., *An Algebra of Processes*, Journal of Computer and Systems Sciences, Vol. 35, No. 2, Outubro, 1987.

[Winkowski 89] Winkowski, J., *An Equivalence of Communicating Processes in Distributed Environments*, Fundamenta Informaticae XII, 1989, pp. 97-128.

[Wirth 77] Wirth, N., *Toward a Discipline of Real Time Programming*, Communications of the ACM, Vol. 20, No. 8, August, 1977, pp. 577-583.

[Yavatkar 92] Yavatkar, R., *MCP: A Protocol for Multimedia Coordination and Temporal Sysnchronization in Multimedia Collaborative Applications*, In proc. of the 12th International Conference on Distributed Computing Systems, Yokohama, Japão, Junho, 1992.

[Yi 91] Yi, Wang, *A Calculus of Real Time Systems*, Ph.D. Thesis, Department of Computer Sciences, Chalmers University of Technology, Göteborg, Sweden, 1991.

[Yodaiken 90] Yodaiken, V.; Ramamritham, K., *Specifying and Verififying a Real-Time Priority Queue with Modal Algebra*, Proc. of the Real-Time Systems Sysmposium, Florida, 1990, pp. 300-310.

[YuHsiang 90] YuHsiang, L.; Shyamasundar, R.K., *Static Analysis of Real-Time Distributed Systems*, IEEE Trans. on Software Engineering, Vol.16, No.4, April, 1990, pp. 373-388.

[Zwarico 85] Zwarico, A.; Lee, I., *Proving a Network of Real-Time Processes Correct*, Proceedings of Real-Time Systems Symposium, Dezembro, 1985, pp. 169-177.