UNIVERSIDADE ESTADUAL DE LONDRINA

GILSON DOI JUNIOR

UM MODELO DE FALHAS PARA SISTEMAS DE TEMPO REAL

Londrina
2013

GILSON DOI JUNIOR

UM MODELO DE FALHAS PARA SISTEMAS DE TEMPO REAL

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação, sob orientação do Prof. Dr. Adilson Luiz Bonifácio.

Orientador: Prof. Dr. Adilson Luiz Bonifácio

Londrina
2013

Catalogação elaborada pela Divisão de Processos Técnicos da Biblioteca Central da Universidade Estadual de Londrina.

Dados Internacionais de Catalogação-na-Publicação (CIP)

D657m Doi Junior, Gilson.
Um modelo de falhas para sistemas de tempo real / Gilson Doi Junior. – Londrina, 2013.
66 f. : il.

Orientador: Adilson Luiz Bonifácio.
Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2013.
Inclui bibliografia.

1. Software – Testes – Teses. 2. Engenharia de software – Teses. 3. Programação em tempo real – Teses. 4. Processamento eletrônico de dados em tempo real – Teses. 5. Localização de falhas – Teses. I. Bonifácio, Adilson Luiz. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 519.681

GILSON DOI JUNIOR

UM MODELO DE FALHAS PARA SISTEMAS DE TEMPO REAL

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação Departamento de Computação da Universidade Estadual de Londrina, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação, sob orientação do Prof. Dr. Adilson Luiz Bonifácio.

BANCA EXAMINADORA

_____

Prof. Orientador Dr. Adilson Luiz Bonifácio
UEL – Londrina - PR

_____

Prof. Dr. Arnaldo Vieira Moura
UNICAMP – Campinas - SP

_____

Prof. Dr. Evandro Bacarin
UEL – Londrina - PR

_____

Prof. Dr. Wesley Attrot
UEL – Londrina – PR

Londrina, 10 de abril de 2013.

À minha família

AGRADECIMENTOS

Agradeço à minha família, responsável pela minha formação como pessoa e por ter me inspirado a buscar uma formação acadêmica e o título de mestre. Agradeço ao meu orientador pela dedicação em me guiar durante o período do desenvolvimento deste trabalho.

Aos professores do Departamento de Computação da UEL, em especial os professores Evandro, Jandira e Wesley, que durante meu período de permanência na instituição partilharam seus conhecimentos e ofereceram apoio e conselhos em diversos momentos.

À fundação CAPES pelo apoio aos alunos de mestrado, possibilitando a dedicação exclusiva dos mesmos, no desenvolvimento de nossos trabalhos de pesquisa.

Agradeço também aos meus amigos pela convivência. Não vou citar nomes para não cometer o erro de esquecer alguém, mas de certo modo, todos partilharam essa experiência comigo.

Finalizo agradecendo à todos que de forma direta ou indireta contribuíram para a conclusão de mais essa etapa na minha jornada acadêmica.

“Eu cheguei a conclusão de que, basicamente, toda nossa visão de mundo é, de algum modo, imperfeita ou distorcida”

George Soros

DOI JUNIOR, Gilson. **Um modelo de falhas para sistemas de tempo real**. 2013. 66 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2013.

RESUMO

Sistemas de tempo real são, em geral, sistemas críticos que interagem com o ambiente externo através de eventos de entrada e saída, regulados por restrições de tempo. A atividade de teste em sistemas dessa natureza, via de regra, exige abordagens rigorosas em seu desenvolvimento devido às suas características críticas. Teste baseado em modelos é uma abordagem que se apoia em formalismos, muitas vezes matemáticos, e que proporciona uma maior confiabilidade à fase de teste. No entanto, sua aplicação no desenvolvimento de sistemas de tempo real depende de técnicas adequadas que possam lidar com a evolução contínua do tempo. Várias abordagens contam com métodos de discretização para representar o comportamento de modelos temporizados. Conjuntos de testes são então extraídos de modelos discretizados, permitindo a verificação de conformidade entre especificações e suas respectivas implementações. Neste cenário, quantificar e qualificar o conjunto de teste são tarefas importantes, porém ainda pouco exploradas na abordagem de teste baseado em modelos para sistemas de tempo real. Este trabalho propõe uma forma sistemática de identificação de falhas a partir da estrutura de autômatos grid. O objetivo é definir um modelo de falhas que apoie as atividades de teste, tais como a análise de cobertura e a extração de conjuntos de teste.

Palavras-chave: Sistemas de tempo real. TIOA. Teste. Modelo de falhas.

DOI JUNIOR, Gilson. **A Fault Model for Real-Time Systems**. 2013. 66 p. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2013.

ABSTRACT

Real-time systems are, in general, critical systems that interact with the environment through input and output events regulated by time constraints. The testing activity on systems of this nature requires rigorous approaches on their development due to their critical aspects. Modelbased testing is an approach that relies on formalisms, often mathematical ones, and that provides more reliability to the testing phase. However, model-based testing for real-time systems depends on appropriate techniques that can deal with continuous time evolution. Several approaches rely on discretization methods to represent the behavior of timed models. Test suites are then extracted from discretized models in order to verify the conformance between specifications and their respective implementations. In this scenario the evaluation task of the test suite is important but rarely addressed on model-based testing approaches for real-time systems. In this work we propose a systematic strategy to identify faults on grid automata. This work aim is to define a fault model that supports model-based testing activities such as coverage analysis and test case generation.

Keywords: Real-Time Systems. TIOA. Testing. Fault model

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

BTDA	Bounded Time Domain Automata
ioco	input-output conformance
MEF	Máquina de Estados Finito
TBM	Teste Baseado em Modelos
TIOA	Timed Input/Output Automata
tioco	timed input-output conformance

SUMÁRIO

1 INTRODUÇÃO

A automação de tarefas de apoio ao desenvolvimento de sistemas tem exigido a inclusão de rotinas cada vez mais complexas neste processo. Um dos motivos tem sido o aumento da demanda por sistemas e o fortalecimento da concorrência nesse mercado, aliado a necessidade de se produzir sistemas de qualidade a custos e prazos menores. Dentre essas atividades, estão as atividades de teste, que visam a melhoria da qualidade dos produtos de software [Bertolino 2007]. Entretanto, o processo de teste não garante a ausência de falhas em sistemas. Por isso as atividades de teste são realizadas visando minimizar o risco da ocorrência de falhas de acordo com a limitação dos custos de projeto e das exigências de qualidade do produto.

Em sistemas críticos as garantias de qualidade exigidas são maiores, já que uma falha pode causar grandes prejuízos e danos irreparáveis. Para atender essas exigências se torna necessária a adoção de métodos rigorosos no processo de desenvolvimento, em especial, nas atividades de teste. Teste baseado em modelos (TBM) [Blackburn R. Busser 2004, Utting e Legeard 2007] é uma abordagem promissora que se apoia em informações das especificações de projeto modeladas por formalismos bem definidos. Os formalismos rigorosos representam os requisitos do sistema de modo a evitar ambiguidades em sua interpretação. A adoção de modelos bem definidos permite a automatização da extração de conjuntos de teste, além de auxiliar em atividades como a análise de cobertura e detecção de falhas.

A abordagem de teste baseado em modelos, apesar de bem estabelecida e com vários estudos já realizados, ainda possui desafios que dificultam sua utilização na prática. Um dos desafios é lidar com sistemas de tempo real, onde o instante de tempo influencia no comportamento do sistema e determina a ocorrência de suas ações. Alguns trabalhos de teste baseado em modelos com foco em sistemas de tempo real [Bonifácio e Moura 2011, En-Nouaary, Dssouli e Khendek 2002, En-Nouaary e Hamou-Lhadj 2008, Springintveld, Vaandrager e D'Argenio 2001] adotam técnicas de discretização para lidar com a evolução contínua de tempo. Em geral, sistemas de tempo real resultam num espaço de estado infinito devido a semântica de tempo contínuo, tornando impossível aplicações práticas. Os modelos discretizados

permitem que os requisitos dos sistemas sejam modelados capturando aspectos de tempo com um espaço de estado finito. No entanto, quando granularidades muito finas são adotadas na discretização, o espaço de estado, ainda que finito, se torna muito grande e recai no problema de explosão combinatória de estados [Alur e Dill 1994, Alur 1999].

Estudos recentes demonstram que granularidades mais grossas de discretização podem manter os requisitos do sistema contínuo preservados através de propriedades da simulação do *Timed Input/Output Automata* (TIOA) [Bonifacio e Moura 2009, Bonifácio e Moura 2011]. Assim, é possível a escolha da granularidade se torna flexível, podendo ser adotada uma granularidade mais fina ou mais grossa de acordo com os requisitos de qualidade e resultando em um conjunto de estados mais gerenciável na prática.

Essas abordagens de discretização para sistemas de tempo real permitem a aplicação de testes baseado em modelos na prática. A partir do modelo TIOA, que representa a especificação do sistema, é obtido o modelo discretizado sobre o qual são aplicadas as atividades de teste como a extração de conjuntos de teste, verificação de conformidade e análise de cobertura de falhas.

De acordo com o objetivo dos testes, diferentes estratégias de análise de cobertura podem ser usadas, sejam qualitativas ou quantitativas. Uma estratégia baseada em propostas de teste é considerada qualitativa [Bonifácio e Moura 2011, Weiglhofer e Wotawa 2009] quando identifica uma falha específica. Uma estratégia baseada num modelo de falhas [Chow 1978, En-Nouaary, Khendek e Dssouli 1999] é considerada quantitativa quando todas as falhas ocasionadas pelas potenciais divergências que uma implementação apresenta em relação a sua especificação são identificadas pelo conjunto de teste.

Este trabalho propõe um modelo de falhas para sistemas de tempo real modelados por autômatos grid, obtidos pela discretização de modelos baseados em TIOA. O objetivo é obter um modelo de falhas capaz de identificar as classes de falha para especificações TIOA, possibilitando atividades de teste voltadas para a análise de cobertura de falhas. A estratégia de identificação de falhas desenvolvida se baseia, em parte, nos modelos de falha propostos para as Máquinas de Estados Finitos (MEFs) [Chow 1978] e para o modelo TIOA [En-Nouaary, Khendek e Dssouli 1999]. Porém, a proposta para TIOA não oferece meios de identificação para os tipos de falhas relacionados ao tempo contínuo. Neste trabalho, a classificação das falhas é precisa e realizada sobre o modelo discretizado. Além disso, o trabalho propõe estratégias de identificação das classes de falhas que ocorrem em sistemas de tempo real.

O trabalho está organizado da seguinte forma. O Capítulo 2 apresenta o conceito de teste de software e introduz a abordagem de teste baseado em modelos e são apresenta-

dos alguns trabalhos relacionados que adotam TBM para sistemas de tempo real. O Capítulo 3 apresenta os modelos adotados pelas técnicas de TBM utilizadas ao longo do trabalho. O Capítulo 4 apresenta os modelos discretizados de autômato de região e autômato grid, adotados em estratégias de discretização para o modelo TIOA. Os conceitos de detecção de falhas, proposta de teste, modelo de falhas, bem como os modelos de falha específicos usados como base para este trabalho são apresentados no Capítulo 5. No Capítulo 6 é descrita a generalização da representação discretizada em autômato grid utilizada no modelo de falhas proposto. O Capítulo 7 define o modelo de falhas proposto para TIOA baseado em grids e apresenta exemplos para a identificação de cada classe de falha. O Capítulo 8 apresenta as considerações finais e direções futuras desse trabalho.

2 TESTE DE SOFTWARE

A atividade de teste de software tem o objetivo de identificar se um sistema atende aos requisitos de sua especificação [Bertolino 2007]. O termo teste de software muitas vezes é utilizado de maneira generalizada para qualquer tipo de sistema, mesmo para componentes de hardware ou entre softwares distintos.

As atividades de teste de software podem ser divididas em fases de planejamento, execução e análise de resultados. Na fase de planejamento é definida a estratégia de teste, a abordagem adotada e os aspectos do sistema a serem observados. Essas definições são utilizadas como critérios para definir o conjunto de testes que será executado. A execução é a fase responsável por exercitar o sistema segundo o conjunto de teste definido. Após a fase de execução, a análise de resultados é responsável por determinar se a implementação testada atende os requisitos da especificação do sistema.

Existem diversos paradigmas e abordagens de teste, onde as informações disponíveis em relação a estrutura da implementação a ser testada e como essas informações são utilizadas durante a atividade de teste os classificam em:

Teste estrutural: também chamado de teste de caixa branca, caracterizado pelo acesso a estrutura da implementação candidata, obtendo informações utilizadas no planejamento e execução dos testes.

Teste funcional: também chamado de teste de caixa preta, caracterizado pela utilização de uma interface para interagir com a implementação, sem conhecer previamente informações a respeito de sua estrutura.

Teste de caixa cinza: utiliza uma abordagem mista das duas anteriores. As informações a respeito da estrutura da implementação são utilizadas para obter o conjunto de teste, no entanto, a execução é realizada como no teste de caixa preta.

O foco deste trabalho está na abordagem de teste de caixa preta, mais especificamente os testes baseado em modelos. Essa abordagem de teste utiliza modelos (semi)

formais para especificar os requisitos do sistema e fornecer maior rigor na definição das rotinas de teste.

2.1 Teste Baseado em Modelos

Abordagens de especificação de requisitos que se apoiam no uso de linguagem natural possibilitam interpretações ambíguas, aumentando a possibilidade de falhas no produto final. Um modelo bem definido evita tais ambiguidades e assim, diminui drasticamente a possibilidade de erros de interpretação dos requisitos [Hierons et al. 2009]. Devido a esses benefícios obtidos pela adoção do modelo de especificação (semi) formal, a utilização de técnicas de teste baseado em modelos fornece uma maior confiabilidade do produto final, quando confrontada com técnicas tradicionais [Blackburn R. Busser 2004].

A seleção de conjuntos de teste é outra atividade muitas vezes realizada de maneira manual e arbitrária, não havendo um planejamento bem definido sobre quais aspectos do sistema os testes devem exercitar durante sua aplicação. A abordagem de teste baseado em modelos define rotinas rigorosas de seleção de conjuntos de teste do sistema utilizando informações do modelo de especificação. Além disso, é possível automatizar a extração do conjunto de teste, uma vez que a rotina é rigorosamente definida e não existe ambiguidade na interpretação do modelo.

Existem diversos trabalhos já realizados com a abordagem de teste baseada em modelos. Cada estudo se diferencia pelo modelo adotado, rotina de extração, rotina de execução dos testes e pelo poder de detecção de falhas. Apesar dessas diferenças, o procedimento de teste baseado em modelos pode ser abstraído para uma rotina padrão como mostra a Figura 2.1. Esse procedimento pode ser definido em etapas de: (i) modelagem do conjunto de requisitos de um modelo de especificação do sistema; (ii) extração dos conjuntos de teste a partir do modelo; (iii) execução do conjunto de teste em uma implementação candidata desenvolvida a partir dos mesmos requisitos do modelo; e (iv) o resultado, chamado de veredito, que define se a implementação candidata passa nos testes, atendendo a especificação do sistema [Utting e Legeard 2007].

A abordagem de teste baseado em modelos também inclui atividades que auxiliam a análise dos resultados do teste. O comportamento esperado da implementação é obtido através da sequência de ações de saída observada no modelo de especificação após a aplicação do teste. Uma relação de conformidade [Krichen e Tripakis 2004, Tretmans 2008] define os aspectos avaliados para verificar a ocorrência de falhas na implementação, confrontando as

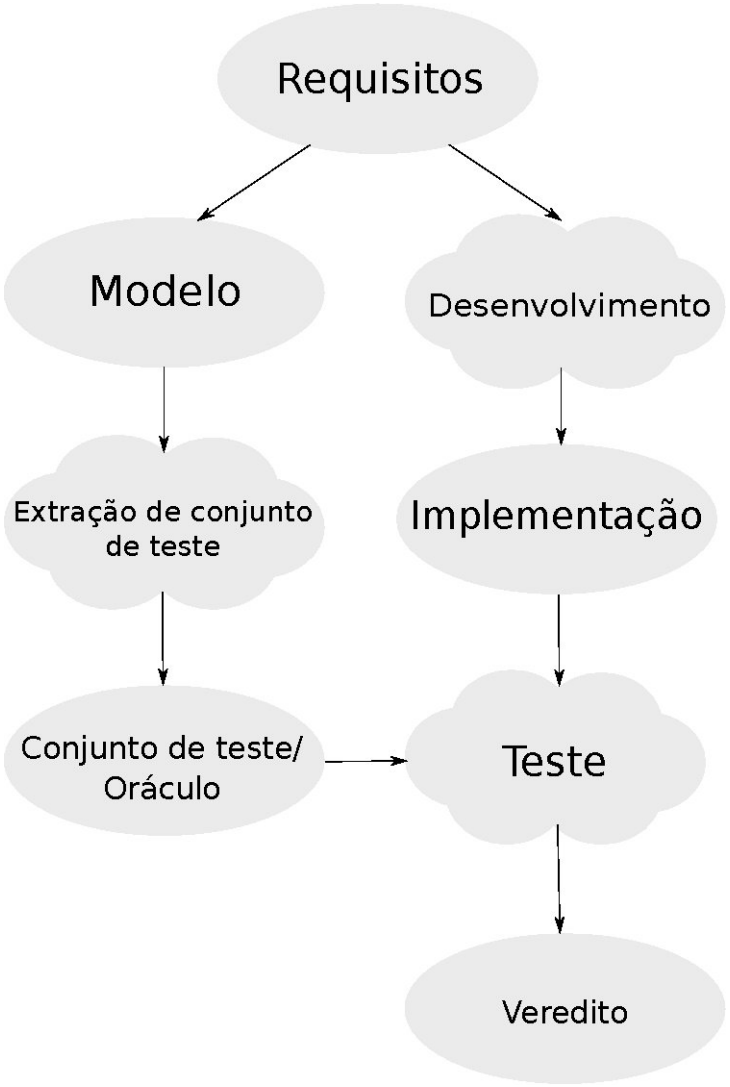

Figura 2.1: Procedimento padrão de Teste Baseado em Modelos

informações do comportamento esperado com o comportamento observado para o conjunto de teste aplicado.

Existem diversas abordagens para definir a relação de conformidade adotada para confrontar a especificação e a implementação candidata. Cada abordagem possui características que se adequam de acordo com o objetivo do teste. O foco desse trabalho é uma abordagem chamada modelo de falhas, que caracteriza classes de falhas através de divergências que a implementação pode apresentar em relação ao modelo de especificação.

2.2 Trabalhos Relacionados

Abordagens de teste baseado em modelos são aplicadas em sistemas de tempo real através da modelagem em TIOA. No entanto, os conjuntos de teste gerados, via de regra,

são muito grandes e inviáveis em aplicações práticas. Visando obter conjuntos de teste mais sucintos, diversos estudos têm sido desenvolvidos adotando técnicas distintas para discretização e caracterização de falhas.

A discretização em [Springintveld, Vaandrager e D'Argenio 2001] adota uma granularidade baseada no número de regiões de relógio. Uma adaptação do método W [Chow 1978] é então aplicado no TIOA discretizado. O problema dessa abordagem está no fato de que o número de regiões de relógio cresce exponencialmente de acordo com o números de relógios. Assim, o espaço de estado resultante da discretização se torna exponencial, inviabilizando a utilização do método na prática.

O método *Timed-Wp* [En-Nouaary, Dssouli e Khendek 2002], uma adaptação do método Wp [Fujiwara et al. 1991] para sistemas de tempo real, também utiliza uma abordagem de discretização baseada no número de regiões de relógio. Logo, o método recai no problema de explosão de estados, se tornando inviável na prática.

Outros trabalhos [En-Nouaary 2008, En-Nouaary e Hamou-Lhadj 2008] adotam uma discretização através de pontos fixos obtidos explorando aa estrutura do TIOA. Em [En-Nouaary 2008] cada transição é explorada para definir a discretização do tempo apenas nos limites do intervalo de cada condição de relógio. Em [En-Nouaary e Hamou-Lhadj 2008] além do limite superior e inferior, o ponto médio do intervalo entre os dois limites de cada condição de relógio também é usado na discretização. Essa abordagem resulta em uma discretização mais controlada e dependente do número de transições do modelo. Porém, continua dependente do número de relógios e o número de pontos observados em cada transição não oferece uma representação razoável do tempo contínuo sobre todo intervalo de habilitação da transição. Logo, não existe nenhuma garantia sobre a ocorrência ou não de falhas nas lacunas de tempo.

Já numa abordagem mais recente [Bonifácio e Moura 2011], um *framework* de teste permite uma discretização mais gerenciável de um TIOA, onde a escolha da granularidade é flexível. O trabalho apresenta a relação de simulação entre um TIOA e sua representação discretizada em autômato grid, quando determinadas propriedades são atendidas. Nesse cenário, a granularidade adotada pode ser mais ou menos grossa, de acordo com a necessidade da aplicação. Assim, o conjunto de estados da representação discretizada se torna mais gerenciável e a aplicação do método se torna viável na prática. No entanto, a eficácia do método é obtida através do uso de propostas de teste, o que restringe o conjunto de falhas detectáveis.

Neste trabalho o objetivo é propor um modelo de falhas baseado em grids, de modo que seja possível caracterizar as potenciais classes de falha de uma implementação.

3 MODELOS DE ESPECIFICAÇÃO

A abordagem de teste baseado em modelos especifica os requisitos do sistema através de formalismos bem definidos. Cada um desses formalismos possui características próprias, que possibilitam descrever particularidades de requisitos de vários tipos de sistema. A máquina de estados finitos [Gill 1962, Wagner 2006], por exemplo, é um formalismo adotado para descrever sistemas com transições que produção de respostas imediatas a ocorrência de estímulos de entrada. Já o modelo TIOA [Alur 1999] é adotado para descrever sistemas de tempo real, onde o instante de ocorrência das ações influenciam o comportamento do sistema e as ações de entrada e saída são dissociadas. Para esses sistemas a adoção de técnicas de teste baseado em modelos é um desafio por conta na modelagem do tempo contínuo e da sua manipulação em aplicações práticas.

Nas seções subsequentes são apresentados os modelos de Máquina de Estados Finitos (MEF) e *Timed Input/Output Automata* (TIOA). Como o foco deste trabalho está nos sistemas de tempo real, a fundamentação do modelo TIOA é apresentada com maiores detalhes para facilitar a compreensão das representações discretizadas apresentadas no Capítulo 4.

3.1 Máquina de Estados Finitos

Um dos modelos mais utilizados no teste baseado em modelos é a Máquina de Estados Finitos (MEF). Uma das razões é sua relativa simplicidade e ampla capacidade de abstração, podendo especificar sistemas pela ocorrência de estímulos de entrada e produção de repostas [Gill 1962, Wagner 2006]. Formalmente, uma MEF é definida como uma tupla $M = (X, Y, S, s_0, \delta, \lambda)$, onde:

- X é o conjunto finito de símbolos do alfabeto de entrada. Como exemplo, $x \in X$ é considerado uma entrada do usuário quando usa um software.
- Y é o conjunto finito de símbolos do alfabeto de saída. Como exemplo, $y \in Y$ é uma saída produzida em resposta a entrada do usuário.

- S é o conjunto finito de estados. O conjunto S representa as configurações possíveis de um sistema.
- s_0 é o estado inicial da máquina. Representar o início da execução de um programa.
- δ é a função de mudança de estado, dada por $\delta : (X \times S) \rightarrow S$. Representa como as configurações de um sistema mudam de acordo com os eventos.
- λ é a função de saída, dada por $\lambda : (X \times S) \rightarrow Y$. Representa a produção saídas em resposta aos eventos de entrada.

A Figura 3.1 apresenta uma MEF com $X = \{press\}$, $Y = \{on, off\}$, $S = \{q_0, q_1\}$, $s_0 = q_0$. Essa MEF pode representar a especificação de um sistema que possui uma lâmpada com dois estados, acesso e apagado, e um botão que comuta o estado da lâmpada ao ser pressionado.

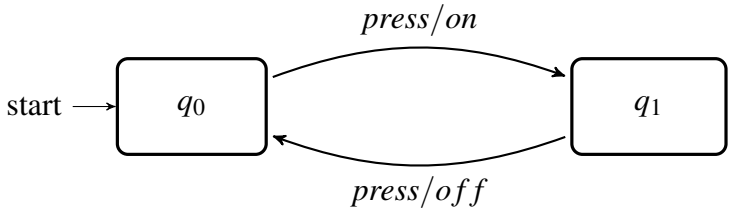

Figura 3.1: Exemplo de uma MEF

3.2 *Timed Input/Output Automata*

Uma das alternativas para se especificar sistemas de tempo real é o modelo TIOA que possibilita a representação de tempo contínuo. Diversos métodos de teste baseado em modelos para sistemas de tempo real adotam o modelo TIOA [Springintveld, Vaandrager e D'Argenio 2001, En-Nouaary, Dssouli e Khendek 2002, Bonifácio e Moura 2011].

A evolução contínua de tempo no TIOA é modelada por variáveis *relógio* c. O conjunto C representa o conjunto de variáveis relógio de um sistema modelado por um TIOA. Os valores dos relógios do conjunto C em determinado instante da execução do sistema é dado por uma função v chamada de *interpretação de relógio*. Assim, uma interpretação de relógio v é uma função parcial de C em $\mathbb{Q}_{\geq}$[1], tal que $v \in [C \rightharpoonup \mathbb{Q}_{\geq}]$ para todo relógio $c_i \in C$, com $1 \leq i \leq n$. Assim $v(c)$ com $c \in C$ indica o valor do relógio c mapeado pela função de interpretação de relógio v em um determinado instante.

[1]Definimos as interpretações de relógio no domínio do racionais não negativos.

Na inicialização do sistema, o conjunto de relógios possui uma *interpretação de relógio inicial* ν_0, onde todo relógio possui valor zero. A partir dessa interpretação os relógios evoluem de maneira síncrona e o valor de um relógio só pode ser alterado de maneira independente aos demais através de uma operação de reinicialização. Uma *operação de reinicialização* é uma atribuição de uma valoração para um relógio, no formato $c := \tau$, onde a reinicialização executada faz o relógio c assumir valor τ. As operações de reinicialização executadas em uma transição de estados do sistema é mapeada pela função $\theta \in [C \curvearrowright \mathbb{Q}_{\geq}]$, que indica quais relógios em C são reinicializados e qual valor $\tau \in \mathbb{Q}_{\geq}$ o relógio deve assumir.

Para representar a evolução do tempo para todos os relógios do sistema a partir de um determinado instante é usada a notação $\nu' = \nu + k$, onde ν é a interpretação de relógio para a instante inicial e $k \in \mathbb{Q}_{\geq}$ é o intervalo de tempo decorrido e ν' a intepretação de tempo após o intervalo, tal que $\nu'(c) = \nu(c) + k$ para todo c no domínicio de ν.

A partir dos valores das variáveis relógios, as restrições de comportamento do sistema são capturadas por *condições de relógio*. Essas condições são definidas pela gramática $\delta := true \mid c \leq \tau \mid \tau \leq c \mid \neg\delta \mid \delta_i \wedge \delta_j$, onde $c \in C$ é um relógio, $\tau \in \mathbb{Q}_{\geq}$ é um valor racional não negativo que indica um instante de tempo, $\leq$ é o operador para menor ou igual, $\neg$ é o operador de negação e $\wedge$ é o operador de conjunção, e $\delta, \delta_i, \delta_j$ são condições de relógio.

Uma condição de relógio é satisfeita em um determinado instante, se as interpretações de cada relógio satisfaz a condição. Formalmente, a notação $\nu \vDash \delta$ diz que uma interpretação de relógio ν satisfaz uma condição de relógio δ. A notação Φ_C é utilizada para representar o conjunto de todas as condições de relógio do sistema. A condição $c_i \leq \tau_i \wedge \neg(c_j \leq \tau_j)$ é um exemplo de condição de relógio que significa: a interpretação de cada relógio, o valor do relógio c_i deve ser menor ou igual a τ_i e do relógio c_j não deve ser menor ou igual a τ_j.

As condições de relógio obtidas através da gramática, eventualmente, podem ser representadas de maneira mais sucinta, simplificando os operadores da expressão obtida seguindo regras matemáticas convencionais. Por exemplo, dizer que o valor do relógio c_j não deve ser menor ou igual a τ_j é o mesmo que dizer que o valor do relógio c_j deve ser maior que τ_j. Logo a condição $\neg(c_j \leq \tau_j)$ pode ser simplificada para $c_j > \tau_j$.

O modelo TIOA (*Timed Input/Output Automata*) [Alur e Dill 1994, Alur 1999] é um modelo de estados finitos que utiliza relógios e condições de relógio para especificar sistemas de tempo real. Esse modelo, é uma extensão do modelo BTDA (*Bounded Time Domain Automata*) capaz de representar sistemas de tempo reais através de relógios que descrevem a semântica de tempo contínuo, condições de relógio atribuídas a estados e transições que descrevem as restrições de tempo do sistema e operações de reinicialização de relógio [Alur e Dill

1994].

Definição 1. *Um BTDA é uma tupla $M = (S, s_0, \Sigma, C, \nu_0, Inv, T)$, onde S é um conjunto finito de estados; s_0 é um estado inicial; Σ é um conjunto finito de ações; C é um conjunto de relógios, onde cada variável relógio $c \in C$ descreve a evolução contínua do tempo; ν_0 é uma interpretação inicial de relógios; $Inv : S \to \Phi_C$ é uma função que mapeia um estado a uma condição de relógio, de modo que, $Inv(s)$ indica uma invariante de evolução de tempo no estado s; e T é um conjunto de transições de estado tal que $T \subseteq (S \times \Sigma \times \Phi_C \times [C \curvearrowright \mathbb{Q}_{\geq}] \times S)$. Uma transição é uma tupla $t = (s, \sigma, \delta, \theta, r)$, onde s é o estado atual, σ uma ação, δ uma condição de relógio, θ uma operação de reinicialização e r um estado alvo.*

Definição 2. *Seja B um BTDA e $\gamma_i = (s_i, \nu_i)$ e $\gamma_j = (s_j, \nu_j)$ duas configurações de B. Um movimento contínuo de γ_i até γ_j para $\tau \in \mathbb{Q}_{\geq}$, denotado por $\gamma_i \xrightarrow[\tau]{} \gamma_j$, existe se $s_i = s_j$, $\nu_j = \nu_i + \tau$ e $\nu_i + \eta \vDash Inv(s_i)$ para todo $0 < \eta \leq \tau$. E um movimento discreto de γ_i até γ_j para $\sigma \in \Sigma$, denotado por $\gamma_i \xrightarrow{\sigma} \gamma_j$, existe se e somente se existe uma transição $(s_i, \sigma, \delta, \theta, s_j) \in T$ tal que $\nu_i \vDash \delta$, $\nu_j = \nu_i \oplus \theta$ e $\nu_j \vDash Inv(s_j)$.*

A transição deve ser interpretada da seguinte forma: se o sistema encontra-se no estado *s* e ocorre a ação σ em um instante em que a interpretação de relógio ν satisfaz a condição de relógio δ, então os relógios são reinicializados de acordo com θ e o sistema passa para o estado alvo *r*. No entanto, o BTDA não possui distinção entre ações, enquanto o TIOA possui um conjunto de ações de entrada e ações de saída. Uma *ação de entrada* é fornecida ao sistema pelo ambiente externo e dessa forma pode ser controlada. Já *ações de saída* são executadas pelo sistema e disparadas internamente por sua implementação. Como a implementação é uma caixa preta, não é possível determinar quando e como essas ações ocorrem e dessa forma não são controladas.

Definição 3. *O modelo TIOA é um BTDA com particionamento do conjunto de ações Σ em subconjuntos de ações de entrada e saída, formalmente descrito pela tupla $M = (B, X, Y)$, onde: $B = (S, s_0, \Sigma, C, \nu_0, Inv, T)$ é um BTDA; $X \subseteq \Sigma$ é um conjunto de ações de entrada ; e $Y \subseteq \Sigma$ é um conjunto de ações de saída, tal que $X \cap Y = \emptyset$;*

O exemplo de sistema apresentado na Figura 3.1 não possui tempo, logo não possui características de um sistema de tempo real. Essa especificação é modificada de forma que a lâmpada alterna do estado apagado para aceso em até uma unidade de tempo após o botão ser pressionado duas vezes em um intervalo de 2 unidades de tempo entre a primeira e a segunda vez; uma vez no estado aceso o sistema alterna automaticamente para o estado apagado após 3 unidades de tempo. Em decorrência dessa alteração, o modelo MEF perde a capacidade de

representar a especificação do sistema e outro modelo capaz de representar sistemas de tempo real, como o modelo TIOA, deve ser adotado. Além disso, as ações de entrada são dissociadas das ações de saída neste novo sistema.

A Figura 3.2 apresenta um TIOA com $X = \{press\}$, $Y = \{on, off\}$, $S = \{q_0, q_1, q_2\}$ e $C = \{c\}$, que representa a especificação do sistema. O estado q_0 representa o estado inicial apagado da lâmpada. Na transição de q_0 para q_1 a ação *press* indica que o botão foi pressionado, sem nenhuma restrição de tempo para que possa ocorrer, e reinicia o relógio c para 0. O estado q_1 ainda representa a lâmpada apagada até que o botão seja pressionado novamente. Se decorrer mais de 2 unidades de tempo sem que o botão seja pressionado, a transição de q_1 para q_0 é habilitada e reinicia o sistema para o estado inicial. Caso contrário, o botão é pressionado dentro do intervalo de 2 unidades de tempo e o relógio é reinicializado para 3. A reinicialização para o valor 3 é um artifício para permitir um intervalo de 1 unidade de tempo, satisfazendo a invariante do estado e não permitir que a transição de q_1 para q_0 seja habilitada. Outra forma de representar essa condição sem que seja utilizada a reinicialização para 3 é utilizar um segundo relógio. Na transição q_1 para q_2 a ação *on* indica que o sistema acende a lâmpada em até 1 unidade de tempo. O estado q_2 indica que a lâmpada está acesa. Após ficar acesa por 3 unidades de tempo, a condição transição de q_2 para q_0 é habilitada, e a ação *off* indica que a lâmpada é apagada, reinicializando o sistema.

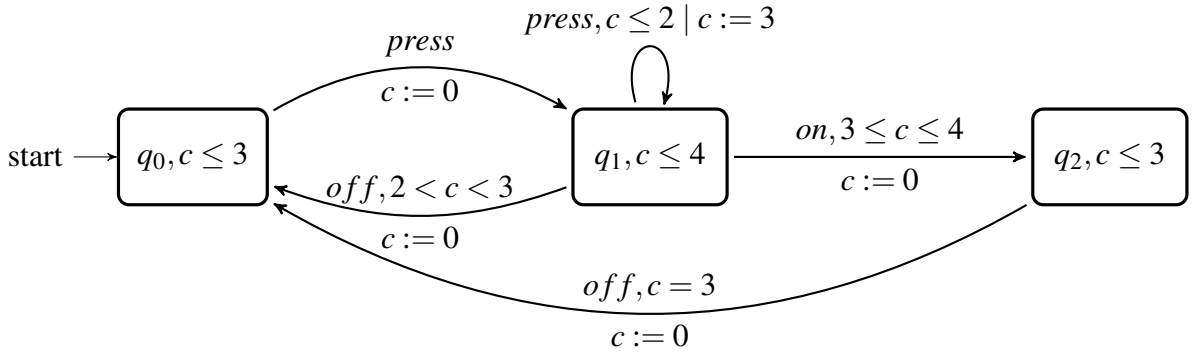

Figura 3.2: Exemplo de TIOA

Para descrever uma sequência de ações de um sistema representado em TIOA, utilizamos palavras temporizadas. Uma palavra temporizada é uma sequência finita de símbolos $(\sigma_1, \sigma_2, ...\sigma_n)$, onde cada símbolo σ_i, para $1 \leq i \leq n$, representa uma ação do sistema ou um valor que indica o intervalo de tempo entre duas ações.

Definição 4. *Seja $M = (S, s_0, \Sigma, C, \nu_0, Inv, T)$ um TIOA, uma palavra temporizada w é uma sequência finita $(\sigma_1, \sigma_2, ...\sigma_n)$, com $n \geq 0$, onde $\sigma_i \in \Sigma \cup \mathbb{Q}_{\geq}$, para $0 \leq i \leq n$.*

Tome o alfabeto $\Sigma = \{press, on, off\}$. A palavra temporizada $(press, 2, press,$

$1, on, 3, off)$ deve ser interpretada como a execução imediata da ação *press*, seguida por 2 unidades de tempo antes da execução de *press* novamente. A ação de saída *on* é executada após $2+1=3$ unidades de tempo e a ação de saída off é executada após $2+1+3=6$ unidades de tempo.

Uma palavra temporizada $w=(\sigma_1,\sigma_2,...\sigma_n)$ descreve uma sequência de ações válida para um TIOA M quando satisfaz a relação de movimentos do TIOA. Para definir a noção de movimento em TIOA utilizamos o conceito de configuração. Uma configuração do TIOA é um par (s,ν) que descreve o estado e as interpretações de relógio do sistema para um determinado instante. A relação de movimento para uma configuração (s,ν) e um símbolo σ é satisfeita quando $\sigma \in \mathbb{Q}_{\geq}$ e $\nu+\sigma$ satisfaz a invariante de s, alterando a configuração do sistema para $(s,\nu+\sigma)$, ou quando $\sigma \in \Sigma$ e existe uma transição $t=(s,\sigma,\delta,\theta,r)$ tal que ν satisfaz a condição de relógio δ, alterando a configuração do sistema para $(r,\nu\oplus\theta)$. A interpretação de relógio $\nu\oplus\theta$ retorna o valor $\theta(c)$ para todo relógio $c\in dom(\theta)$, onde $dom(\theta)$ indica o domínio dos relógio com reinicialização definida em θ; e $\nu(c)$ para todo relógio $c\in dom(\nu)-dom(\theta)$, onde $dom(\nu)-dom(\theta)$ indica o domínio dos relógios definidos na interpretação ν e que não possuí reinicialização definida em θ.

Definição 5. *Seja $M=(S,s_0,\Sigma,C,\nu_0,Inv,T)$ um TIOA, $\gamma_i=(s_i,\nu_i)$ e $\gamma_j=(s_j,\nu_j)$ duas configurações de M com $s_i,s_j\in S$ e $\psi=(\sigma_1,...,\sigma_n)$ uma palavra temporizada. A relação de movimento, denotada por $(\sigma_1\cdot\psi,\gamma_i)\vdash(\psi,\gamma_j)$, é satisfeita quando: (i) $\sigma_1\in\mathbb{Q}_{\geq}$ e $\nu_i+\sigma_1=\nu_j$ e $s_j=s_i$ ou (ii) quando $\sigma_1\in\Sigma$, existe uma transição $t\in T$ com $t=(s_i,\sigma_1,\delta,\theta,s_j)$ tal que $\gamma_j=(s_j,\nu_i\oplus\theta)$, $\nu\vDash\delta$ e $\nu\oplus\theta\vDash Inv(s_j)$.*

A notação $(\psi_1,\gamma_1)\overset{k}{\vdash}(\psi_2,\gamma_2)$ é utilizada para uma sequência de k movimentos, com $k\geq 0$. O fechamento recursivo transitivo $(\psi_1,\gamma_1)\overset{*}{\vdash}(\psi_2,\gamma_2)$ também é usado para representar uma sequência de zero ou mais movimentos. Alternativamente a notação $\gamma_1\overset{\psi}{\Vdash}\gamma_2$ é utilizada para representar uma sequência de movimentos para cada símbolo da palavra.

4 DISCRETIZAÇÃO DE MODELOS TEMPORIZADOS

A modelagem de tempo contínuo em sistemas de tempo real é suprida pela adoção de modelos temporizados. No entanto, a representação de todo o espaço de estado resultante do comportamento desses modelos pode resultar em infinitos estados. Por isso a dificuldade de métodos de teste baseado em modelos ao lidar com sistemas de tempo real, já que visitar cada estado do modelo se torna impossível.

A discretização do tempo é uma das estratégias para lidar com os modelos temporizados, resultando num conjunto finito de configurações do sistema a ser testado. O autômato de região [Alur e Dill 1994, En-Nouaary, Khendek e Dssouli 1999] e o autômato grid [Bonifácio e Moura 2011] são representações discretizadas para o modelo TIOA. Essas representações modelam o comportamento do sistema após a discretização, permitindo verificar se todos os requisitos são mantidos, além de auxiliar em outras atividades de teste [Bonifácio e Moura 2011, En-Nouaary, Khendek e Dssouli 1999].

4.1 Autômato de região

O autômato de região é uma representação do TIOA obtida através de classes de equivalência entre interpretações de relógio. Uma classe de equivalência define para quais interpretações o modelo deve ter o mesmo comportamento. A Figura 4.1 apresenta as regiões obtidas para a evolução de tempo em um TIOA que possui dois relógios: o eixo horizontal representa a evolução do relógio x e o eixo vertical representa a evolução do relógio y. Cada seta rotulada com Ri, com $1 \leq i \leq 44$, representa uma região de relógio. Por exemplo, a região $R16$ representa o conjunto de interpretações em que x e y evoluem de maneira síncrona, para valores entre 0 e 1. Já a região $R10$ representa o conjunto de interpretações de relógio alcançadas quando uma transição habilitada entre os valores 0 e 1, para os relógios x e y, reinicializa o relógio y para 0 e mantém o valor de x inalterado.

Definição 6. *Uma região de relógio* $[v]$ *é uma classe de equivalência de interpretações de relógio tal que para todo par* $v_i, v_j \in [v]$, v_i *é equivalente a* v_j, *denotado por* $v_i \sim v_j$, *se:*

- $\forall c \in C$, $\lfloor v_i(c) \rfloor = \lfloor v_j(c) \rfloor$;
- $\forall c_i, c_j \in C$, $fract(v_i(c_i)) \leq fract(v_i(c_j))$ *se e somente se* $fract(v_j(c_i)) \leq fract(v_j(c_j))$, *onde* $fract(v(c)) = v(c) - \lfloor v(c) \rfloor$ *para* $c \in C$;
- $\forall c \in C$, $fract(v_i(c)) = 0$ *se e somente se* $fract(v_j(c)) = 0$.

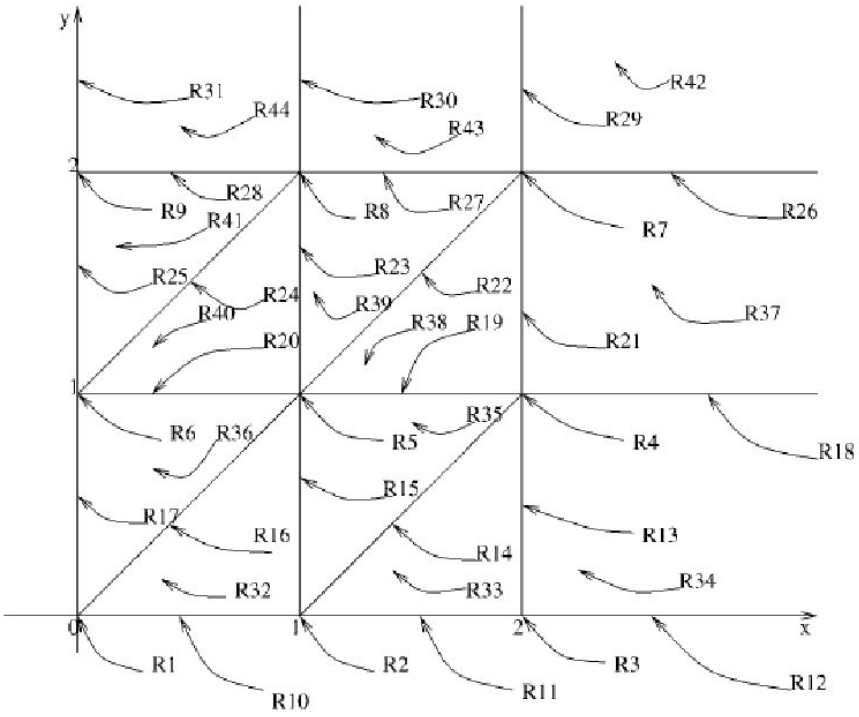

Figura 4.1: Gráfico de regiões com dois relógios

Um autômato de região é uma representação que discretiza o número de configurações do sistema modelado pelo TIOA, representando a evolução do tempo através das classes de equivalência. Assim, cada estado do autômato é uma tupla $(s, [v])$ que representa uma classe de equivalência de configurações, onde s é um estado do TIOA e $[v]$ uma região de relógio. Como o número de regiões de relógio encontradas em um TIOA é sempre finito, é possível representar todas as configurações do sistema. Uma transição $((s, [v]), \sigma, (r, [v']))$, com $\sigma \in \Sigma$, existe no autômato de região se o TIOA possui uma transição $(s, \sigma, \delta, \theta, r)$ tal que $v \vDash \delta$, $\theta(v) = v'$ e $v' \vDash Inv(r)$.

A Figura 4.2 apresenta uma versão reduzida do autômato de região do TIOA da Figura 3.2. Nesse autômato de região são representadas apenas as transições $(q_1, press, c \leq 2, c := 3, q_1)$ e $(q_1, on, 3 \leq c \leq 4, c := 0, q_2)$. Os estados $(q_1, c = 0)$, $(q_1, 0 < c < 1)$, $(q_1, c = 1)$, $(q_1, 1 < c < 2)$ e $(q_1, c = 2)$ representam as configurações em que as interpretações das regiões de relógio satisfazem a condição da transição $(q_1, press, c \leq 2, c := 3, q_1)$. Dessa forma, existe

uma transição rotulada com a ação de entrada *press* em cada um desses estados. Como a transição $(q_1, press, c \leq 2, c := 3, q_1)$ possui a operação de reinicialização $c := 3$, o estado alvo de cada transição rotulada *press* é o estado com configuração $(q_1, c = 3)$. Já os estados $(q_1, c = 3)$, $(q_1, 3 < c < 4)$ e $(q_1, c = 4)$ representam as configurações em que as interpretações das regiões de relógio satisfazem a condição de relógio da transição $(q_1, on, 3 \leq c \leq 4, c := 0, q_2)$. Logo, existe uma transição rotulada com a ação de saída *on* com estado alvo $(q_2, c = 0)$ em cada um desses estados.

Definição 7. *Um autômato de região obtido de um TIOA* $M = (S, s, \Sigma, C, \nu, Inv, T)$ *é dado por* $M_R = (S_R, s_R, \Sigma_R, T_R)$*, onde:*

- S_R *é um conjunto finito de estados, onde cada estado é uma tupla na forma* $(s, [\nu])$ *e,* s *é um estado do tioa e* $[\nu]$ *é uma região de relógio;*
- s_R *é o estado inicial, dado como* $(s, [\nu_0])$*;*
- $\Sigma_R = \Sigma \cup d$ *é um alfabeto, onde d representa um intervalo indefinido de evolução de tempo chamado delay;*
- T_R *é um conjunto finito de transições. Existe uma transição* (p, σ, q) *entre dois estados* $p, q \in S_R$ *e* $\sigma \in \Sigma_R$ *quando* $\sigma = d$*,* $p = (s, [\nu])$ *e* $q = (s, [\nu]')$ *tal que* $[\nu]' = [\nu + \sigma]$ *e* $[\nu'] \vDash Inv(s)$*; ou quando* $\sigma \in \Sigma$*,* $p = (s, \nu)$ *e* $q = (r, \nu')$ *tal que existe uma transição* $(s, \sigma, \delta, \theta, r)$ *em* T*,* $\nu \vDash \delta$*,* $\nu'(c) = \nu(c) \oplus \theta$ *e* $\nu' \vDash Inv(r)$*.*

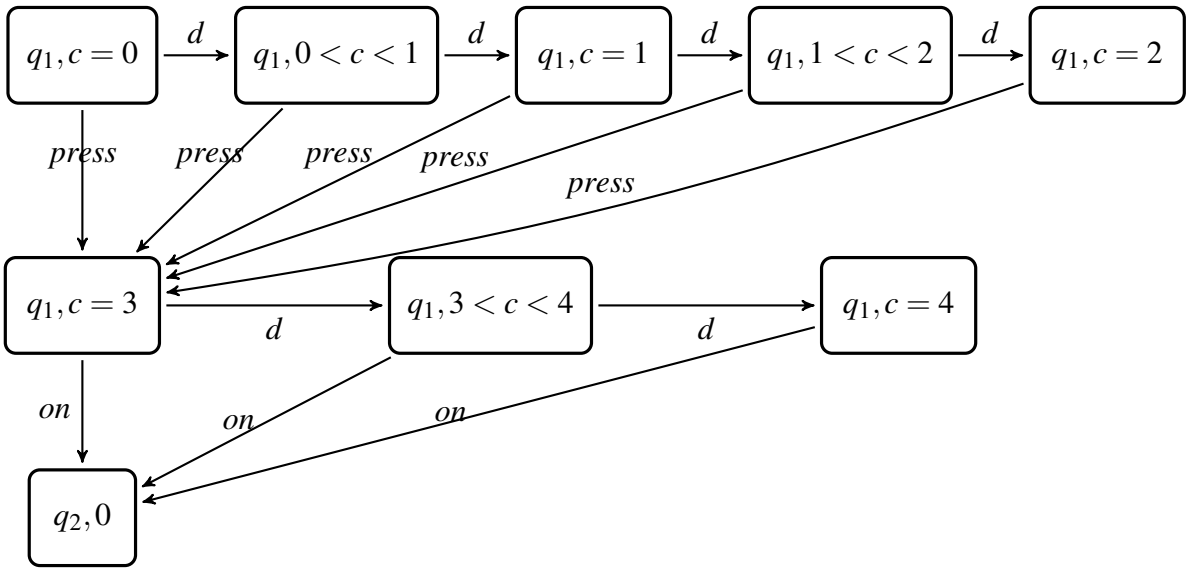

Figura 4.2: Autômato de região parcial do TIOA da Figura 3.2

O modelo discretizado de autômato de região é utilizado em algumas estratégias de teste baseadas no modelo TIOA [En-Nouaary, Dssouli e Khendek 2002]. O autômato de

região é usado para caracterizar as falhas no modelo TIOA e realizar a análise de cobertura de conjuntos de teste [En-Nouaary, Khendek e Dssouli 1999]. No entanto, discretizações baseadas em autômatos de região dependem do número de regiões de relógio. Assim, essas abordagens recaem no problema da explosão combinatória de estados conforme o número de relógios do modelo cresce.

4.2 Autômato grid

O autômato grid também resulta da discretização de modelos TIOA. Estudos recentes [Bonifácio e Moura 2011] demonstram que a obtenção de um grid a partir de um TIOA pode assumir granularidades mais adequadas. Logo, abordagens nessa direção permitem a aplicação de estratégias de teste em espaços de estado mais gerenciáveis.

Definição 8. *Um grid obtido de um TIOA* $M = (S, s, \Sigma, C, \nu, Inv, T)$ *é definido por* $M_G = (S_G, s_G, \Sigma_G, T_G)$*, onde:*

- S_G *é um conjunto finito de estados, onde cada estado* $p \in S_G$ *é uma configuração de M;*
- s_G *é o estado inicial que representa a configuração* (s, ν_0) *do sistema ;*
- $\Sigma_G = \Sigma \cup \{g\}$ *é um alfabeto finito;*
- T_G *é um conjunto finito de transições. Existe uma transição* (p, σ, q) *entre dois estados* $p = (s, \nu_i), q = (r, \nu_j) \in S_G$ *com* $\sigma \in \Sigma_G$ *quando* $\sigma \in \Sigma$ *e existe uma transição* $(s, \sigma, \delta, \theta, r) \in T$ *tal que* $\nu_i \vDash \delta$*,* $\nu_j = \nu_i \oplus \theta$ *e* $\nu_j \vDash Inv(r)$*; ou quando* $\sigma = g$ *tal que* $r = s$*,* $\nu_j = \nu_i \oplus g$ *e* $nu_j \vDash Inv(s)$*.*

O Algoritmo 1 mostra o algoritmo de obtenção de um autômato grid a partir de um TIOA [Bonifácio e Moura 2011]. Esse algoritmo se baseia nas propriedades de um TIOA g-ajustado e L-limitado que garante a simulação do comportamento do TIOA pelo autômato grid obtido. A relação de simulação do TIOA pelo grid, assim como, as propriedades de TIOA g-ajustado e L-limitado são apresentadas em [Bonifacio, Moura e Simao 2008].

A Figura 4.3 apresenta uma versão reduzida do autômato grid obtido com uma granularidade $\frac{1}{2}$ para o TIOA da Figura 3.2. Nesse grid são representadas as transições $(q_1, press, c \leq 2, c := 3, q_1)$ e $(q_1, on, 3 \leq c \leq 4, c := 0, q_2)$.

Uma sequência de ações num grid descreve o comportamento do sistema através da noção de palavras de grid. Uma palavra de grid é uma sequência finita $(\sigma_1, \sigma_2, ...\sigma_n)$,

Algorithm 1: algoritmo de discretização de TIOA

Entrada: Uma granularidade $g = \frac{1}{k}$, onde k é um inteiro positivo, um TIOA $M = (S, s, \Sigma, C, \nu, Inv, T)$ g-ajustado e um limite superior $L \in \mathbb{Q}_{\geq}$ G-ajustado
Saída: Um L,g-grid $M_{L,g} = (S_G, s_G, \Sigma_G, T_G)$

início
  $T_G \leftarrow \emptyset$ // o conjunto de transições do grid é iniciado como vazio;
  $RS \leftarrow s_G = (s, \nu_0)$ // conjunto de estados do grid inicializado com o estado inicial do TIOA ;
  $HS \leftarrow \emptyset$ // conjunto de estados visitados ;
  enquanto $RS \setminus HS \neq \emptyset$ **faça**
    selecione um estado (s, ν) de $RS \setminus HS$;
    mova (s, ν) de RS para HS; **para cada** $(s, z, \delta, \theta, r) \in T$ **faça**
      se $\nu \vDash \delta$ *e* $\nu \oplus \theta \vDash Inv(r)$ **então**
        seja $\eta = (\nu \oplus \theta)_L$;
        adicione a transição $((s, \nu), z, (r, \eta))$ à T_G ;
        adicione o estado (r, η) ao conjunto RS, se $(r, \eta) \notin HS$;
      fim
      se $\nu + h \vDash Inv(s)$ *para todo* $0 < h \leq g$ **então**
        seja $\eta = (\nu + g)_L$;
        adicione a transição $((s, \nu), z, (s, \eta))$ à T_G ;
        adicione o estado (s, η) ao conjunto RS, se $(s, \eta) \notin HS$;
      fim
    fim
  fim
  $S_G \leftarrow HS$;
  $\Sigma_G \leftarrow \Sigma \cup \{g\}$;
fim

onde cada símbolo da sequência é um símbolo do alfabeto do grid e representa uma ação do sistema ou uma granularidade de tempo. Como a representação de tempo contínuo é abstraída na discretização, a evolução contínua do tempo evolui em unidades de g.

Para uma palavra de grid $w = (\sigma_1 \cdot \sigma_2 \cdot \ldots \cdot \sigma_n)$ com $n \geq 1$, se $w' = (\sigma_i \cdot \ldots \cdot \sigma_j)$ é uma subsequência de w, com $i < j$, e para todo k, com $i \leq k \leq j$, temos $\sigma_k = g$ então w' pode ser denotado por $g^{(j-i+1)}$ onde $(j-i+1)g$ indica a passagem de tempo com relação a w'.

Assim como no modelo TIOA, existe uma relação de movimento que determina se a palavra de grid descreve uma sequência de movimentos válidos no autômato grid. No entanto, a noção de tempo é abstraída durante a discretização, de modo que as configurações do TIOA são representadas pelos estados do grid e a evolução do tempo é representada por transições rotuladas. A relação de movimento para grid é definida quando existe uma transição habilitada entre dois estados rotulada com o primeiro símbolo da palavra de grid.

Definição 9. *A relação de movimento* $(\sigma \cdot \psi, p) \vdash (\psi, q)$ *é definida sobre um autômato grid*

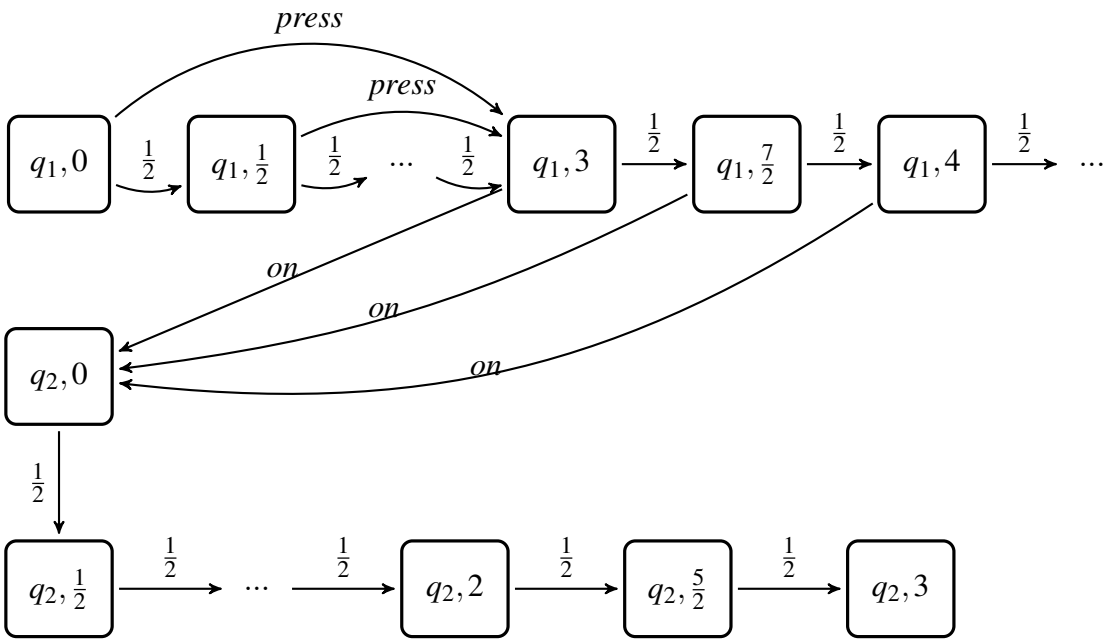

Figura 4.3: Autômato grid parcial do TIOA da Figura 3.2

$M_G = (S_G, s_G, \Sigma_G, T_G)$ *quando existe uma transição* $(p, \sigma, q) \in T_G$*, com* $p, q \in S_G$ *e* $\sigma \in \Sigma_G$.

Assim como na relação de movimento para TIOA, a noção de sequência de k movimentos, $\overset{k}{\vdash}$, fechamento recursivo transitivo, $\overset{*}{\vdash}$, e movimento sobre palavras, $\overset{\psi}{\Vdash}$, são adotadas para autômatos grid.

Note que devido a alta densidade de estados, a escolha da granularidade assume um papel importante na discretização do TIOA. Por isso o tema é um dos principais alvo de estudos em métodos que lidam com uma representação discretizada de sistemas de tempo real, como por exemplo, os métodos de teste baseado em modelos. Diferente dos autômatos região, abordagens recentes de discretização [Bonifácio e Moura 2011] permitem uma flexibilidade na escolha da granularidade. A obtenção de grids com espaço de estado mais compactos permite uma análise de cobertura de falhas, como proposto neste trabalho, para sistemas de tempo real em aplicações práticas.

5 DETECÇÃO DE FALHAS BASEADA EM MODELOS

A detecção de falhas em sistemas computacionais ocorre durante a análise dos resultados dos testes. Em geral, a análise em abordagens de teste baseado em modelos se dá através da comparação do comportamento esperado das implementações e o comportamento extraído do modelo de especificação. Contudo, para proceder a análise dos resultados é preciso caracterizar as potenciais falhas dentro de um escopo delimitado. A simples divergência entre os comportamentos especificados e comportamentos observados nem sempre resulta em uma falha. Por isso, o conceito de falha deve ser bem definido, principalmente em abordagens que permitem a automatização, como é o caso dos testes baseado em modelos.

Uma forma de caracterizar falhas e permitir sua detecção é através de relações de conformidade como o *ioco* [Tretmans 2008] e o *tioco* [Krichen e Tripakis 2004]. Essas relações caracterizam falhas entre o conjunto de comportamentos permitidos por uma especificação e aqueles comportamentos apresentados pelas implementações.

A relação *ioco* (*input-output conformance*) é definida pela comparação do conjunto de saídas observadas na especificação e na implementação candidata. A relação considera que uma implementação está em conformidade com a especificação se para qualquer sequência de entrada, a sequência de saída observada na implementação é a mesma da especificação. A relação *tioco* (*timed input-output conformance*) é uma extensão da relação *ioco* que tem como única particularidade a inclusão de passagens de tempo entre as ações de entrada e saída para determinar o instante em que cada ação ocorre. Essas relações de conformidade, *ioco* e *tioco*, caracterizam falhas em modelos *IOTS* [Tretmans 2008] e *TIOTS* [Krichen e Tripakis 2004], respectivamente, e não para TIOA. Além disso, não existe uma estratégia prática que possibilite a identificação dessas falhas ou a análise de cobertura para estes modelos.

Uma outra abordagem de detecção de falhas é usando propostas de teste [Bonifácio e Moura 2011, Weiglhofer e Wotawa 2009], onde falhas são caracterizadas pela descrição de um comportamento específico. Uma abordagem que utiliza propostas de teste é considerada qualitativa, pois não se preocupa com a quantidade de falhas detectadas e sim com a

capacidade de detectar uma falha singular.

Já uma abordagem usando um modelo de falhas [Chow 1978, En-Nouaary, Khendek e Dssouli 1999] é considerada quantitativa, pois foca na caracterização de classes de falhas. Falhas de uma mesma classe são ocasionadas em decorrência de um mesmo tipo de divergência entre a especificação e uma implementação. Por isso, um modelo de falhas caracteriza as potenciais divergências que uma implementação apresenta.

A seguir são apresentados modelos de falha para MEF e TIOA.

5.1 Modelo de falhas para MEF

A análise de cobertura de falhas do método W [Chow 1978] de geração de casos de teste para modelos MEF é realizada com base num modelo de falhas, dividindo as falhas em três classes: falhas de operação, falhas de transferência e falhas de estados extra/ausentes.

Falhas de operação ocorrem quando uma transição apresenta uma saída distinta da especificada, caracterizada por sua função de operação. Falhas de transferência ocorrem quando uma função de transferência leva a execução de uma transição a um estado distinto do especificado. Falhas de estados extras/ausente ocorrem quando a implementação não está em conformidade com a implementação, por conta da inclusão ou retirada de estados.

Para ilustrar os três tipos de falha adotamos o modelo da Figura 3.1 como especificação. A Figura 5.1 representa uma implementação candidata com falha de operação na transição do estado q_0 para q_1, pois a saída especificada para a transição é *on* e na implementação é *off*. A detecção desse tipo de falha se dá com um conjunto de testes que exercita cada transição do modelo de especificação ao menos uma vez. Um conjunto de teste que detecta falhas de operação para o exemplo contém a sequência {(*press, press*)}. Através dessa sequência a saída esperada segundo a especificação é a sequência *(on, off)* e a saída observada na implementação candidata da Figura 5.1 é *(off, off)*.

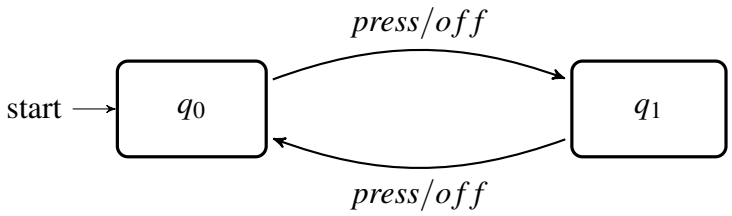

Figura 5.1: Implementação com falha de operação para a especificação da Figura 3.1

Já a Figura 5.2 apresenta o modelo de uma implementação candidata com

falha de transferência na transição entre os estados q_0 e q_1. Essa classe de falha é detectada pela utilização de um conjunto de sequências de ações de entrada, também chamado de conjunto caracterização [Chow 1978, Fujiwara et al. 1991, Bonifacio, Moura e Simao 2008]. Um conjunto caracterização é capaz de distinguir cada par de estados do modelo de especificação. Um conjunto caracterização para a especificação na Figura 3.1 contém a sequência *{(press)}*. Quando o modelo de especificação é minimal, o conjunto caracterização é aplicado após cada sequência de teste para identificar o estado alvo. Por exemplo, no conjunto de teste {(*press*); (*press, press*)} cada sequência termina em um estado distinto do modelo. Quando este conjunto é concatenado com o conjunto caracterização é obtido o conjunto de teste {(*press, press*); (*press, press, press*)} capaz de identificar a falha de transferência na implementação candidata da Figura 5.2 através da sequência de saída (*on, off, off*).

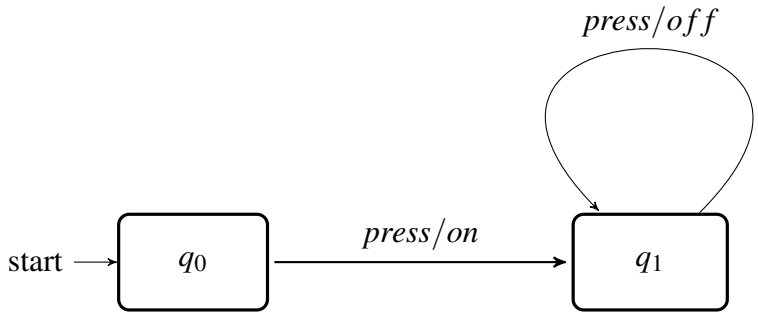

Figura 5.2: Implementação com falha de transferência para a especificação da Figura 3.1

A Figura 5.3 apresenta uma implementação com falha ocasionada por um estado extra. Essa classe de falha é detectada através de uma cobertura de estados considerando estados extras. O conjunto de teste { (*press, press*); (*press, press, press*) } do exemplo anterior não aponta nenhuma divergência de comportamento entre as saídas da especificação e o exemplo da Figura 5.3.

Como a estrutura da implementação candidata não é conhecida, a existência de estados extras é uma hipótese. Como os estados extras não tem um comportamento especificado, a detecção se dá prevendo cada possibilidade de entrada. O conjunto de teste deve então considerar a combinação dos símbolos do alfabeto de entrada com comprimento proporcional ao número de estados extras assumido na hipótese. Assim, um estado extra é detectado, no exemplo, pela sequência de teste (*press, press, press, press*).

No caso de falhas de estados ausentes a detecção é realizada por um conjunto que possui ao menos uma sequência que leve o controle da MEF a cada estado da especificação. Essas sequências são concatenadas ao conjunto caracterização para verificar se os comportamentos são conformes. Caso um dos estados esteja ausente, ao menos uma sequência que

alcance este estado na especificação deve detectar a falha.

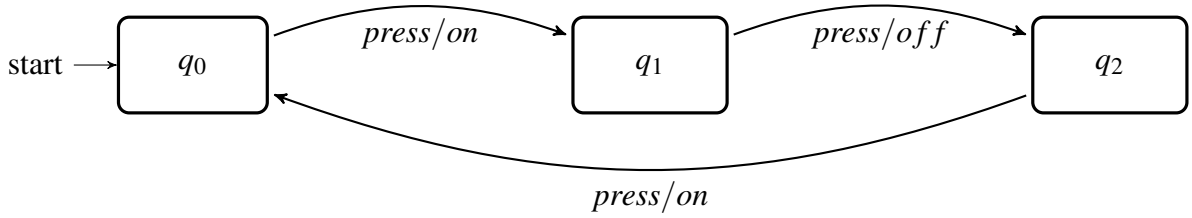

Figura 5.3: Implementação com falha de estado extra para a especificação da Figura 3.1

O modelo de falhas apresentado considera MEFs conexas, determinísticas e completamente especificadas. Numa MEF inicialmente conexa cada estado pode ser alcançado a partir do estado inicial, característica necessária para possibilitar a cobertura de estados. MEFs determinísticas são aquelas que para dado símbolo de entrada no máximo uma transição é habilitada partindo de um determinado estado. MEFs completamente especificadas possuem uma transição para cada símbolo do alfabeto de entrada partindo de cada estado da máquina. Este fato elimina a possibilidade de falhas de transições ausentes.

5.2 Modelo de falhas para TIOA

A análise de cobertura de falhas para o método *Timed Wp* [En-Nouaary, Khendek e Dssouli 1999, En-Nouaary, Dssouli e Khendek 2002] sobre o modelo TIOA é dividida em: falhas de ação, falhas de transferência, falhas de restrição de condição de relógio, falhas de relaxamento de condição de relógio e falhas de reinicialização. Essa análise se baseia na representação de autômatos região [Alur e Dill 1994]. A ocorrência de falhas no TIOA é detectada pela obtenção de autômatos de região distintos daquele que modela a especificação.

A análise de cobertura do método *Timed-Wp* exercita a estrutura do modelo revelando qualquer alteração no autômato de região pela aplicação do conjunto de teste. Este modelo de falha classifica as potenciais falhas em: falhas de tempo, falhas de ação e falhas de transferência de estados. Para ilustrar as classes de falha deste modelo tomamos o TIOA da Figura 3.2 como especificação.

A falha de ação ocorre quando uma transição é implementada com uma ação diferente da especificada. A Figura 5.4 mostra uma implementação candidata com falha de ação na transição do estado q_2 para q_0. Essa falha é detectada por um conjunto de teste que cobre todas as transições do autômato de região, similar a falha de operação do modelo MEF. A única distinção é que, no modelo MEF cada transição é exercitada por uma entrada, produzindo uma

saída. Já no modelo TIOA a transição indica a execução de uma ação, que pode ser de entrada, fornecida pelo ambiente externo, ou de saída, produzida de maneira autônoma pelo sistema.

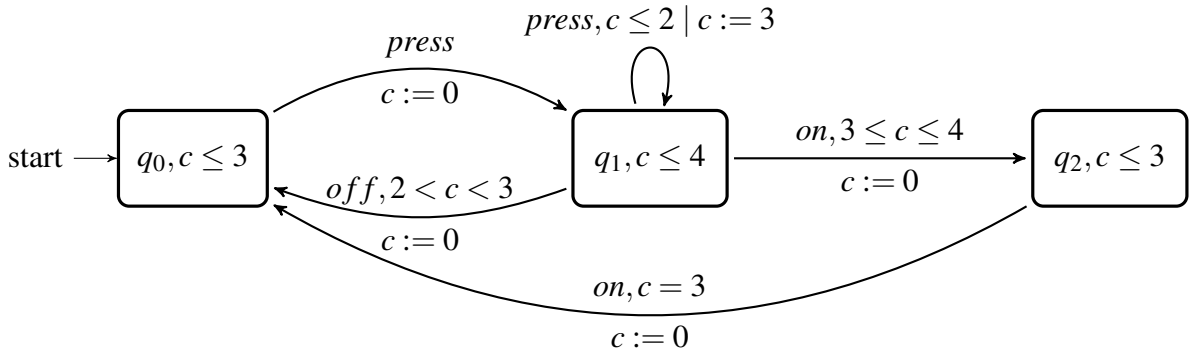

Figura 5.4: Implementação com falha de ação para a especificação da Figura 3.2

A falha de transferência ocorre quando uma transição é implementada com o estado destino diferente do especificado. A Figura 5.5 mostra uma implementação candidata com falha de transferência na transição do estado q_2 para q_2. De maneira similar ao modelo MEF essa falha é identificada com o auxílio de conjuntos caracterização. O conjunto caracterização é capaz de identificar que o comportamento do estado q_2 na implementação candidata apresenta falha, pois o sistema não produz saídas.

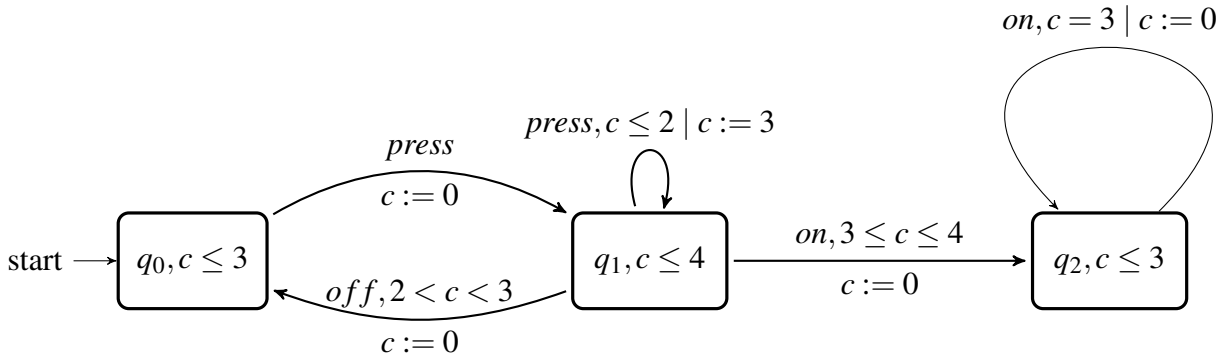

Figura 5.5: Implementação com falha de transferência para a especificação da Figura 3.2

As falhas de tempo são decorrentes de alterações relacionadas a semântica de tempo, como nas condições e reinicializações de relógios. Uma falha de restrição da condição de relógio ocorre quando uma transição com ação de entrada é implementada com uma condição de relógio mais restrita, limitando o intervalo de tempo para a execução da ação. No caso de ações de saída a restrição da condição é considerada válida uma vez que o intervalo restrito está contido no intervalo especificado. A Figura 5.6 apresenta uma implementação com falha de restrição de condição de relógio na transição de q_1 para q_1. A ocorrência de uma falha de

restrição de condição de relógio altera as regiões de relógio obtidas para o TIOA. Uma vez que os estados do autômato de região estão associados às regiões de relógio, essa classe de falha é detectada por um conjunto de teste que cobre todos os estados do autômato de região.

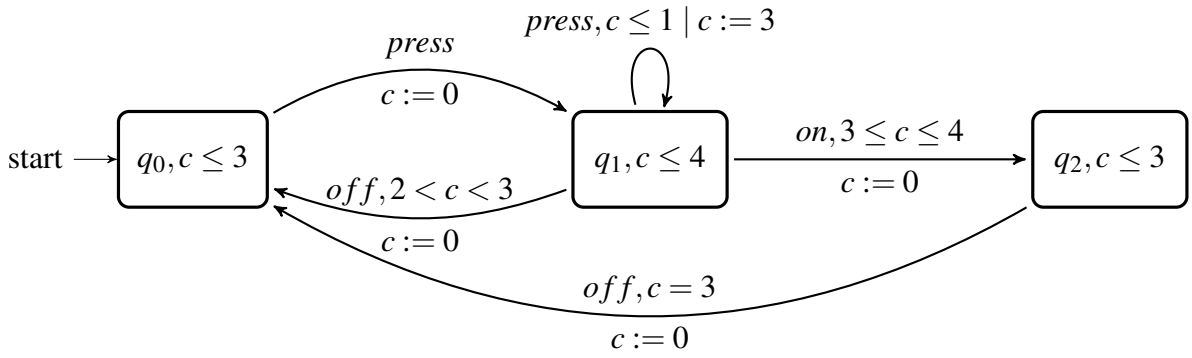

Figura 5.6: Implementação com falha de restrição de condição de relógio para a especificação da Figura 3.2

Uma falha de relaxamento da condição de relógio ocorre quando uma transição é implementada com uma condição de relógio menos restrita, permitindo a ocorrência da ação num intervalo maior que o especificado. Assim como na falha de restrição, a falha de relaxamento na condição de relógio também altera as regiões obtidas para o TIOA. Logo, essa classe de falha é detectada por um conjunto de teste que cobre todos os estados do autômato de região. No caso de ações de saída, essa classe de falha resulta em transições extras no autômato de região da implementação. Essas transições são detectadas, pois ações de saída são não controláveis. A Figura 5.7 apresenta uma implementação com falha de relaxamento de condição de relógio na transição de q_1 para q_1. Assim como na restrição de relógio, essa falha altera as regiões de relógio alcançadas em cada estado do TIOA. Assim, essa classe de falha é detectada por um conjunto de teste que cobre todos os estados do autômato de região.

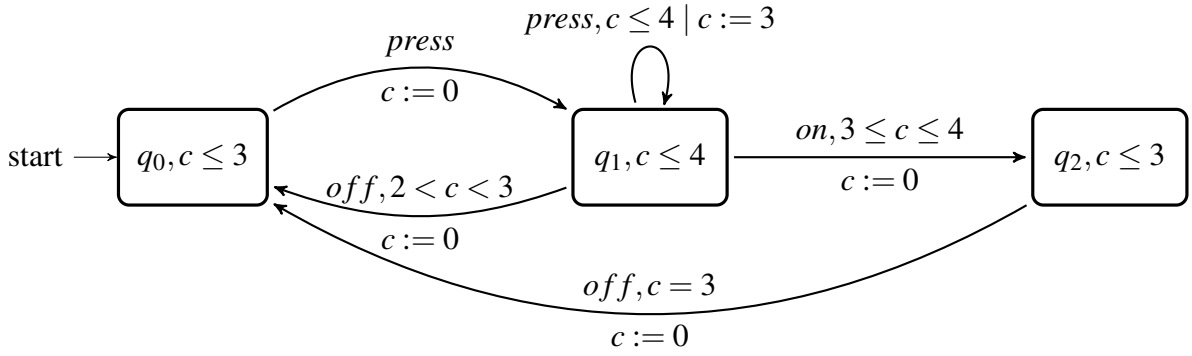

Figura 5.7: Implementação com falha de relaxamento de condição de relógio para a especificação da Figura 3.2

Uma falha de reinicialização de relógio ocorre quando o conjunto de relógios

a ser reinicializado não é implementado corretamente. Esse tipo de falha se divide em dois sub-casos:

- na ausência da reinicialização dos relógios o tempo continua a evoluir alcançando valores não previstos na especificação.
- na implementação de uma reinicialização inexistente o relógio passa a não explorar valores limites do sistema.

A Figura 5.8 apresenta uma implementação candidata que apresenta uma falha de ausência de reinicialização de relógio na transição entre q_2 e q_0.

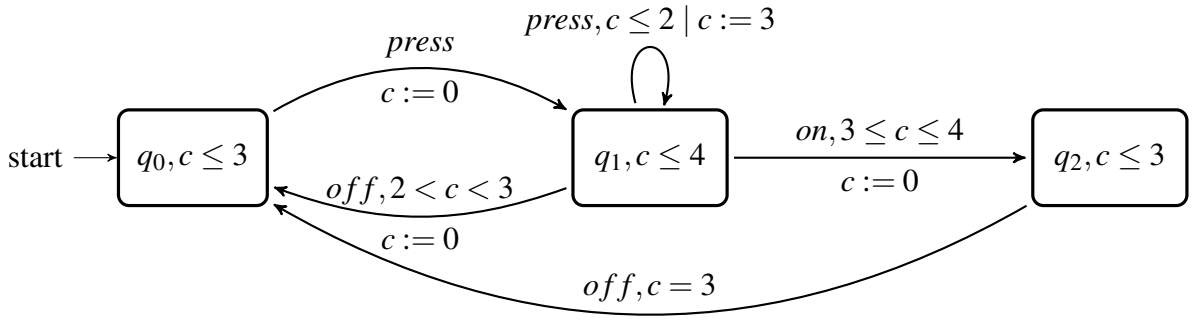

Figura 5.8: Implementação com falha de reinicialização de relógio para a especificação da Figura 3.2

Em outro cenário, com o TIOA da Figura 5.8 representando a especificação do sistema e a Figura 5.9 uma implementação candidata, a transição entre q_2 e q_0 apresenta falha através de uma reinicialização inexistente na especificação.

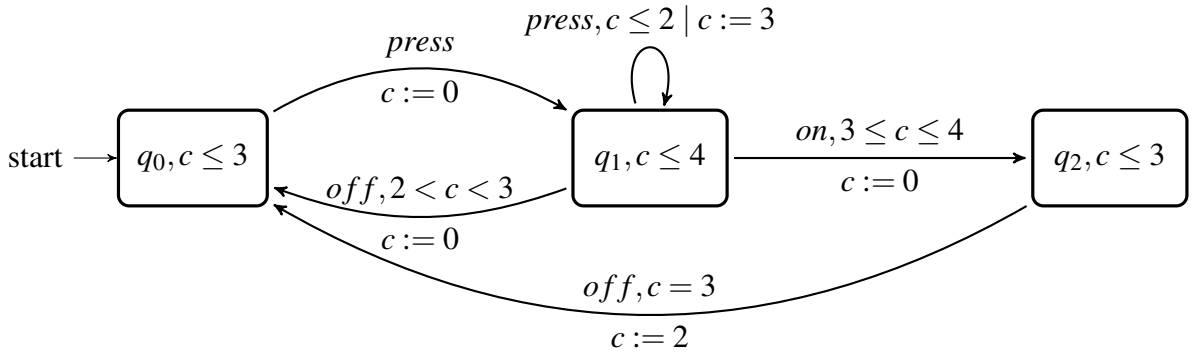

Figura 5.9: Implementação com falha de reinicialização de relógio para a especificação da Figura 5.8

O método *Timed-Wp* assume uma ação especial chamada *ResetClock* para a reinicialização dos relógios. O *ResetClock* identifica a ocorrência de uma reinicialização na

implementação e indica qual o relógio reinicializado. Dessa forma, as falhas de reinicialização de relógio podem ser detectadas pela ocorrência ou não do sinal *ResetClock*. O problema dessa abordagem é que, na prática, a implementação candidata é uma caixa preta, logo as reinicializações de relógio não são ações observáveis. Assim, a detecção da falha de reinicialização fica limitada a um escopo de implementações muito específico, no qual o sistema desenvolvido, possui ações observáveis que indicam quais relógios são reinicializados.

Além da limitação da detecção da falha de reinicialização, esse modelo não se preocupa em identificar um conjunto de características que distingua cada classe de falha. O modelo permite apenas a detecção da ocorrência de falhas através da divergência entre o autômato de região da implementação e da especificação. Assim, a partir do veredito da falha ocorrida na implementação não é possível identificar qual o tipo de falha apresentada.

6 GENERALIZAÇÂO DA DISCRETIZAÇÃO DO TEMPO

Um modelo de falhas é definido obtendo a caracterização das classes de falha que apontam potenciais divergências entre o modelo de especificação e suas respectivas implementações candidatas. O principal objetivo do modelo de falhas é possibilitar, que a partir da especificação, sejam identificadas as falhas em qualquer implementação candidata que atendam as hipóteses de teste. Essas hipóteses delimitam o escopo das implementações, mas ainda englobam uma infinidade de implementações candidatas que as satisfazem. Assim, as classes de falha devem ser caracterizadas de maneira generalizada, de modo que, seja possível identificá-las em qualquer implementação candidata que atenda as propriedades estabelecidas.

Um autômato grid, após a discretização de um TIOA, possui apenas estados e transições, sem os elementos relacionados ao tempo contínuo. No entanto, o autômato grid simula o modelo TIOA, representando a evolução do tempo através de sequências de movimentos grid. Por isso, na análise de um autômato grid é preciso considerar algumas generalizações sobre a representação dos estados e transições que capturam a evolução contínua do tempo. Essa generalização está relacionada à semântica de tempo representada no modelo de autômato grid e serve de base para a caracterização das classes de falha do modelo proposto neste trabalho.

6.1 Representação da Evolução Contínua de Tempo

Na evolução de um sistema modelado por um TIOA um único estado s representa um conjunto de configurações na forma $(s, v_1, v_2, .., v_n)$, onde cada v_i, $1 \leq i \leq n$, é uma interpretação de relógio e v_i satisfaz a invariante de s. Na discretização do TIOA, cada estado do grid representa uma configuração do TIOA.

Com o intuito de generalizar a representação de um estado do TIOA em grid algumas convenções são assumidas. Com relação aos rótulos dos estados, adotaremos p e q para indicar estados do grid, r e s para indicar estados do TIOA. Também é adotado o rótulo de

um estado do TIOA em maiúsculo para indicar o conjunto generalizado de todos os estados do grid que representam configurações oriundas desse estado do TIOA. Por exemplo, o conjunto S_i indica um conjunto $\{q_1, q_2, ..., q_n\}$ de estados do grid, onde cada $q_i, 1 \leq i \leq n$, representa uma configuração $(s, v_1, v_2, .., v_n)$ do sistema.

Na representação em TIOA, o tempo evolui e os valores dos relógios são incrementados continuamente enquanto a invariante de um estado é satisfeita e não ocorre nenhuma transição. Denominamos esse intervalo de evolução de tempo de *linha de tempo*. A linha de tempo é representada no grid pela ocorrência de uma sequência de transições $(p_0, g, p_1), (p_1, gp_2), ..., (p_n, g, p_{n+1})$ e representa a evolução de um intervalo de ng unidades de tempo, onde $g \in \mathbb{Q}_{\geq}$ é a granularidade de discretização. Como todas as transições na sequência são rotuladas com g, os estados p_x, para $1 \leq x \leq n+1$ representam um único estado no qual o sistema modelado pelo grid permanece enquanto o tempo evolui por um intervalo de ng unidades de tempo. A Figura 6.1 mostra a estrutura generalizada de uma linha de tempo no autômato grid para um intervalo ng. Para generalizar os estados do grid que representam a sequência de transições de uma linha de tempo adotamos S_i', onde S_i é o rótulo em maiúsculo do estado do TIOA representado.

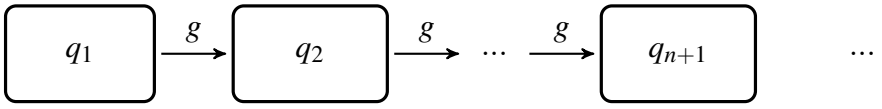

Figura 6.1: Representação de linha de tempo no autômato grid

Quando o TIOA M possui apenas um relógio, o conjunto de estados do grid derivado a partir de um estado do TIOA também possui somente uma única linha de tempo maximal para o intervalo de sua invariante. Quando o TIOA M possui mais de um relógio, o conjunto de estados S_i derivado para cada $s_i \in S$ é uma união dos subconjuntos que representam as linhas de tempo maximal do estado. A linha de tempo de cada subconjunto é derivada a partir de uma combinação da interpretações de relógio do TIOA que representa um instante de tempo em que esse estado é alcançado. Por exemplo, assuma $C = \{c_1, c_2\}$. Existe uma linha de tempo para cada combinação de valores para os quais os relógios c_1 e c_2 podem assumir após uma transição, *e.g.* quando uma transição reinicia o relógio c_1 para o valor τ, o estado alvo dessa transição possui uma linha de tempo combinando τ com cada valor que o relógio c_2 pode assumir satisfazendo δ da transição do TIOA, de acordo com a granularidade g. A Figura 6.2 mostra a representação de duas linhas de tempo $S_i^1 = \{q_1, ..., q_n\}$ e $S_i^2 = \{p_1, ..., p_n\}$, representando $S_i = S_i^1 \cup S_i^2$.

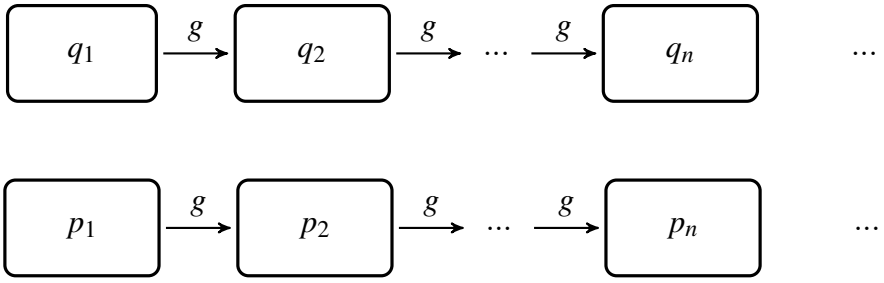

Figura 6.2: Representação de múltiplas linhas de tempo

6.2 Representação da Condição de Relógio

Uma única transição no TIOA é representada por uma ação habilitada num intervalo de tempo definido pela condição de relógio. Como o autômato grid não possui condições de relógio a ação da transição do TIOA é representada com uma transição em cada estado resultante de uma configuração com a condição de relógio habilitada.

As condições de relógio obtidas através das regras apresentada na Seção 3.2 podem ser reduzidas a três casos quando a transição possui restrições somente sobre um relógio: $c = \tau$, $\tau \leq c \leq \tau'$ e $\tau < c < \tau'$. Note que, mesmo quando a condição é escrita na forma $c < \tau'$ o limite inferior considerado é $\tau = 0$; e quando a condição é escrita na forma $\tau < c$ o limite superior é $\tau' = L$, onde L é o limite da invariante do estado ou, quando a invariante é *true*, L é o limite da discretização. Quando a condição é na forma $c = \tau$, t é representada no grid por apenas uma transição para a configuração (s, τ). Quando a condição é na forma $\tau \leq c \leq \tau'$, t é representada no grid por uma transição para cada estado resultante de uma configuração com um valor discretizado para c entre τ e τ'. Quando a condição é na forma $\tau < c < \tau'$, t é representada como uma transição com condição na forma $\tau + g \leq c \leq \tau' - g$.

A gramática de derivação das condições de relógio da Seção 3.2 também inclui regras como $\delta_i \wedge \delta_j$ e $\delta_i \vee \delta_j$, onde a condição de relógio é obtida pela composição de duas condições. Para condições obtidas pela regra $\delta = \delta_i \wedge \delta_j$ a condição δ é habilitada quando as condições δ_i e δ_j são satisfeitas simultâneamente. Assim, a transição é simulada no grid em todos os estados que representam uma configuração para o estado atual do sistema e uma interpretação de relógio que satisfaz δ_i e δ_j. Para condições obtidas pela regra $\delta = \delta_i \vee \delta_j$ a condição δ é habilitada quando qualquer uma ou ambas as condições δ_i e δ_j são satisfeitas. Assim, a transição é simulada no grid em todos os estados que representam uma configuração para o estado atual do sistema e uma interpretação de relógio que satisfaz δ_i ou δ_j. Note que tanto δ_i quanto δ_j também podem ser condições de relógio compostas.

Outra particularidade é o estado do qual parte uma transição num TIOA com

mais de um relógio possuir mais de uma linha de tempo. Nessa caso, a transição t é representada em cada linha de tempo do estado e o conjunto T' é obtido pela união dessas representações de diferentes linhas de tempo.

6.3 Representação de Reinicialização de Relógio

A ocorrência de uma ação num TIOA pode resultar na alteração dos valores de alguns de seus relógios através do conjunto de reinicialização definido na transição. Como a reinicialização altera a configuração no estado alvo da transição, o conjunto de reinicialização θ influencia a representação do estado alvo das transições do grid que representam essa transição.

Para fins de generalização, o conjunto de reinicialização de uma transição é dividido em três casos: quando nenhum relógio é reinicializado; quando um subconjunto estritamente contido do conjunto de relógios C é reinicializado; e quando todos os relógios em C são reinicializados.

Quando θ não reinicializa nenhum relógio, a interpretação de relógio no estado alvo do TIOA tem a mesma interpretação de relógio do instante em que a ação ocorre. Sejam, por exemplo, os estados $p, p' \in T'_G$ a representação de duas configurações do sistema com $p = (s, v_1, v_2, ..., v_n)$ e $p' = (s, v'_1, v'_2, ..., v'_n)$, onde v_i, v'_i, $1 \leq i \leq n$, são interpretações de relógio. Assuma que as configurações p e p' habilitem a transição t e $v'_i = v_i + g$, de modo que existam as transições (p, σ, q) e (p', σ, q') no grid e uma transição (p, g, p'). Os estados alvo q e q' representam o estado alvo de t com a mesma configuração das interpretações de relógio de p e p', respectivamente, tal que $q = (r, v_1, v_2, ..., v_n)$ e $q' = (r, v'_1, v'_2, ..., v'_n)$ e existe uma transição (q, g, q'). A Figura 6.3 mostra como esse caso é representado no autômato grid.

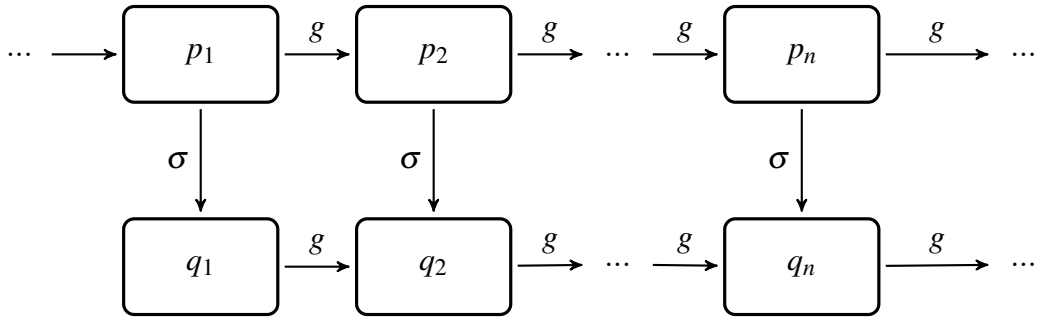

Figura 6.3: Transição discretizada sem reinicialização de relógios

Quando o conjunto θ reinicializa um subconjunto estritamente contido de C, cada instante de tempo em que a ação ocorre resulta em uma combinação distinta entre as interpretações de relógio. Essas combinações encontram-se em linhas de tempo distintas. Nesse

caso, o estado alvo de cada transição que representa t no autômato grid está em uma linha de tempo distinta. A Figura 6.4 mostra como esse caso é representado no autômato grid.

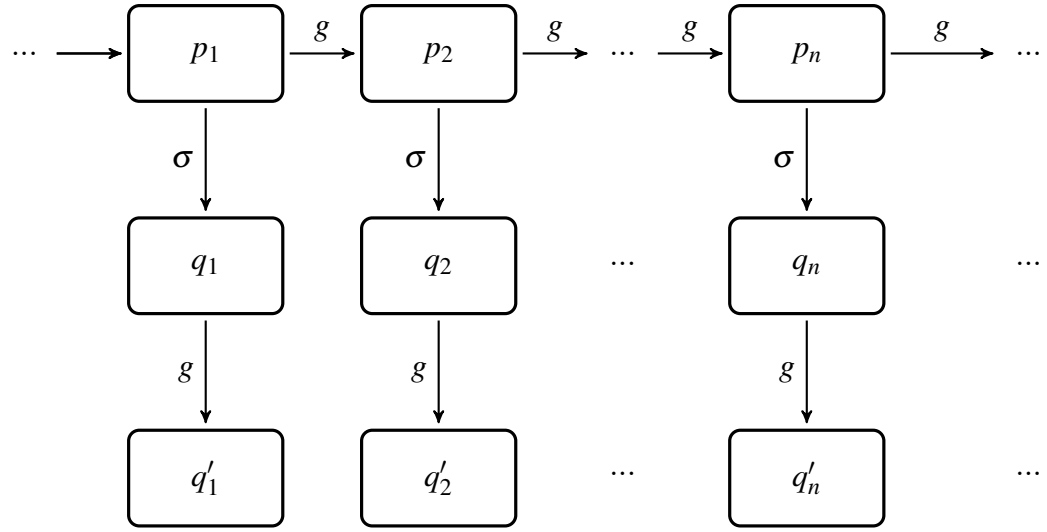

Figura 6.4: Transição discretizada com um subconjunto de reinicialização de relógios

Quando o conjunto θ reinicializa todos os relógios em C a configuração do estado alvo da transição é a mesma para qualquer instante de tempo em que a ação ocorre. No autômato grid esse caso é identificado quando toda transição que representa t leva a um único estado alvo em comum. A Figura 6.5 mostra como esse caso é representado no autômato grid.

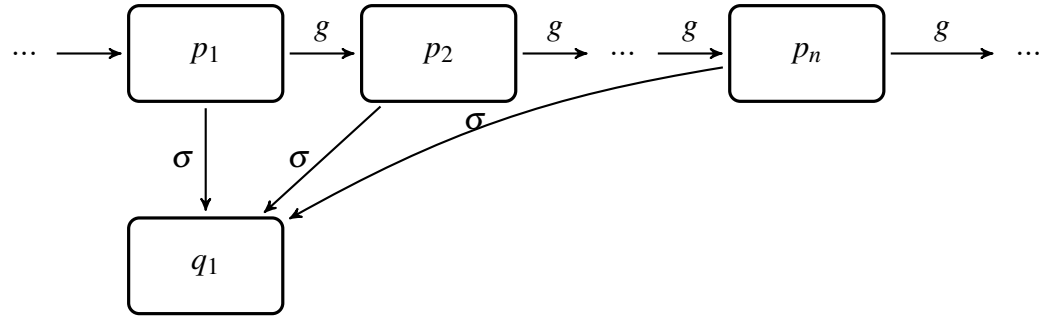

Figura 6.5: Transição discretizada com reinicialização de todos os relógios

7 MODELO DE FALHAS BASEADO EM AUTÔMATO GRID

Um modelo de falhas permite uma análise precisa na identificação das falhas que uma implementação candidata apresenta com relação a sua especificação. O modelo de falhas introduzido em [En-Nouaary, Khendek e Dssouli 1999] caracteriza as classes de falha de um sistema de tempo real modelado por um TIOA. A análise de cobertura do conjunto de teste do método *Timed-Wp* é baseada na utilização desse modelo de falha e da representação em autômato de região. Uma implementação candidata que apresenta uma falha é representada por um autômato de região distinto da especificação.

O modelo de falha baseado em autômato de região não permite a identificação das classes em que cada falha ocorre. Cada estado do autômato de região representa um estado do TIOA e uma região de relógio associada. A identificação de qualquer falha não é possível, pois uma região de relógio representa um número infinito de interpretações equivalentes. Assim, não é possível apontar o instante de tempo preciso da configuração em que ocorre uma falha.

No autômato grid, cada configuração do sistema é indicada por cada estado que representa um estado do TIOA e a interpretação de relógio em um instante. Dessa forma, ao detectar uma falha, a classe a qual ela pertence pode ser identificada e o instante de tempo em que ela ocorre pode ser apontado. Essa proposta utilizando autômatos grid possibilita discretizações mais flexíveis, desviando da abordagem tradicional baseada em regiões de relógio. Os grids baseados em discretizações usando regiões de relógio podem ser submetidos à abordagem clássica de detecção. Já o modelo de falhas baseado em autômato de região não pode ser aplicado diretamente a modelos discretizados com granularidades flexíveis.

Uma análise preliminar apresentada em [Doi Jr e Bonifácio 2012] aponta as classes do modelo de falhas para TIOA baseado em autômato grid. No entanto, a caracterização de falhas se baseia em exemplos específicos, dando apenas indícios de uma generalização da identificação das falhas para grids. Através da generalização do tempo contínuo é possível formalizar a análise preliminar dessas falhas de maneira mais precisa, estabelecendo assim um modelo de falhas proposto neste trabalho.

Como o número de implementações candidatas para um modelo de especificação é infinito, propor um modelo de falhas para qualquer tipo de implementação candidata se torna impossível. Assim, algumas hipóteses de teste são assumidas para delimitar o escopo do conjunto de implementações e garantir a detecção das classes de falhas através do modelo de falhas proposto.

Hipótese 1. *O modelo de falhas para TIOA baseado em autômato grid assume que:*

1. *O comportamento da implementação candidata pode ser modelado por um TIOA que possibilita a discretização em grid para a granularidade adotada [Bonifacio, Moura e Simao 2008];*

2. *A implementação candidata possui o mesmo alfabeto da especificação, possibilitando que toda ação da especificação também seja ação na implementação candidata;*

3. *A especificação e a implementação candidata são grids completos, onde cada estado possui uma transição para cada ação de entrada. Quando o grid não é completo é possível completá-lo adicionando transições extras que levam para um estado de falha;*

4. *A especificação é um grid determinístico, onde cada estado possui apenas uma transição para cada rótulo.*

5. *A especificação é um grid com saídas isoladas, onde para cada estado somente uma transição é rotulada com ação de saída.*

O modelo de falha proposto divide as falhas do TIOA em falhas dependentes de tempo e falhas não dependentes de tempo. Cada classe de falha do modelo TIOA é representada no modelo de autômato grid, apontando se as divergências entre a especificação e a implementação candidata.

Para exemplificar a identificação das falhas são apresentados exemplos para exercitar cada uma das classes de falha. Os exemplos são desenvolvidos sobre uma versão modificada do protocolo multimídia descrito em [En-Nouaary 2008]. A Figura 7.1 apresenta o TIOA que especifica o protocolo multimídia. Esse TIOA possui um conjunto de estados $S = \{s_0, s_1, s_2, F\}$, com conjunto de ações de entrada $X = \{image, sound\}$, conjunto de ações de saída $Y = \{ackAll, error\}$ e um conjunto de relógios $\{c_1, c_2\}$. O estado s_0 é estado inicial do sistema. A transição entre do estado s_0 até s_1 indica que o sistema recebeu uma imagem, indicada pela ação *image*, em um intervalo de até 2 unidades de tempo. Ao executar a transição o sistema reinicia seus dois relógios. O estado s_1 representa um estado de espera pelo som

da imagem. Se após 2 unidades de tempo não ocorrer a ação *sound* indicando o recebimento do som da imagem, uma ação de saída *error* é emitida e o sistema reinicia seus dois relógios, retornando ao estado inicial. Caso contrário, a ação *sound* na transição entre s_1 e s_2 indica que o sistema recebeu o som da imagem dentro do intervalo de 2 unidades de tempo, passando o controle do sistema para o estado s_2 com a reinicialização do relógio c_1.

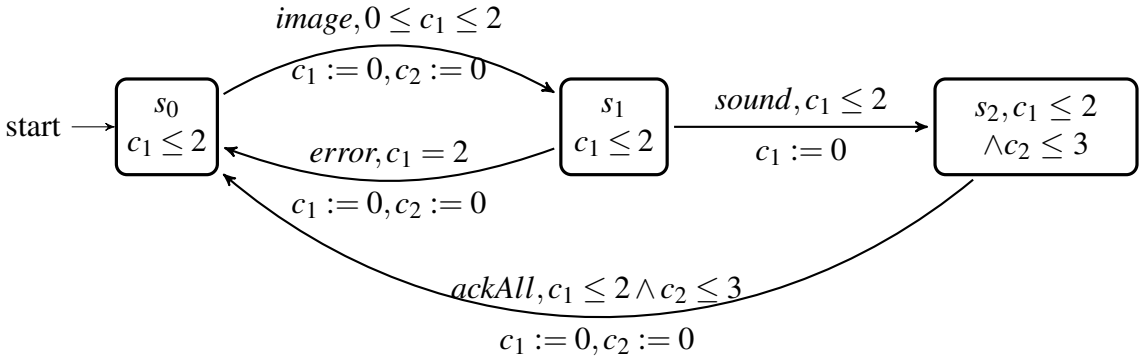

Figura 7.1: O TIOA modificado do sistema multimídia

O modelo de falhas apresentado assume que o TIOA seja completo. Por isso, uma transição é especificada para quando um som é recebido no estado inicial, quando uma imagem é recebida no estado s_1 e quando uma imagem ou som é recebido no estado s_2. Essas transições são adicionadas, levando o controle do sistema para um estado de falha F. Se qualquer ação de entrada for executada no estado de falha, o sistema reinicializa seus relógios para 0. Após 2 unidades de tempo no estado F o sistema emite um sinal de erro e volta ao estado inicial. O TIOA de especificação completo é apresentado na Figura 7.2.

Os exemplos de falhas a seguir utilizam esse modelo como especificação do sistema e para facilitar a representação gráfica da discretização em grid, adotaremos uma granularidade mais grossa, com $g = 1$, de acordo com a abordagem de [Bonifacio, Moura e Simao 2008].

7.1 Falhas Não Dependentes de Tempo

As seções a seguir mostram a identificação das falhas não dependentes de tempo que ocorrem quando o modelo TIOA da implementação candidata e da especificação apresentam divergências em relação às ações e estado alvo de suas transições.

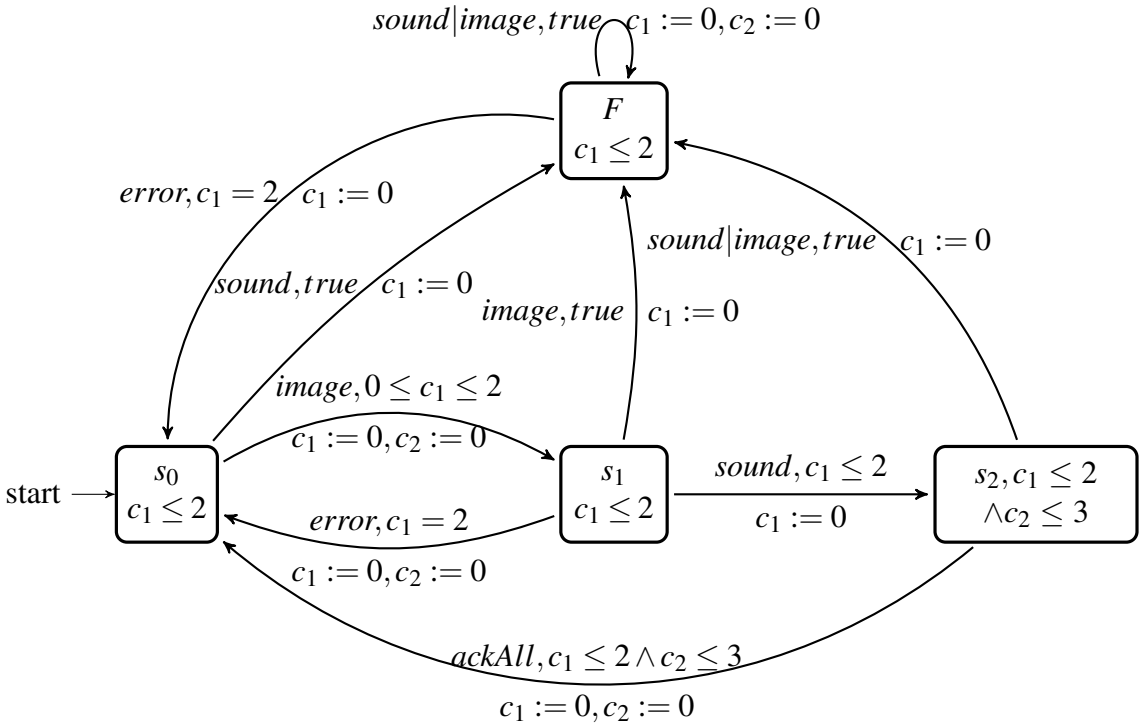

Figura 7.2: O TIOA multimídia completo

7.1.1 Falha de Ação

A falha de ação ocorre devido à implementação de uma transição com ação distinta da especificação. O autômato grid representa uma transição t do TIOA com um conjunto de transições T'. A falha de ação na transição t ocorre quando uma transição de T' apresenta rótulo distinto do especificado, sendo ambos os rótulos de ações de saída. A condição de relógio delimita o intervalo em que o sistema pode habilitar a transição. Como a ação de saída é produzida de forma autônoma pelo sistema, não é possível garantir que a ação ocorra em algum instante do intervalo. Dessa forma, se ao menos uma transição em T' apresenta rótulo distinto entre a especificação e implementação, então a falha é caracterizada. Como assumimos um grid com saídas isoladas (Ver Hipótese 1) é garantido que a ação observada na implementação não é referente a outra transição que ocorre nesse estado.

Definição 10. *Seja $M_G = (S_G, s_G, \Sigma_G, T_G)$ um grid obtido a partir do TIOA de especificação $M = (S, s_0, \Sigma, C, \nu, Inv, T)$ e $M'_G = (S'_G, s'_G, \Sigma'_G, T'_G)$ o grid de uma implementação candidata. Assuma que $T' \subseteq T_G$ é o conjunto de transições que representa uma transição $t = (s_i, \sigma, \delta, \theta, s_j) \in T$. Seja $(q_i, \sigma, q_j) \in T'$ e Ψ um conjunto de palavras de grid tal que para cada $\psi \in \Psi$ temos $s_G \overset{\psi}{\Vdash} q_i$ e $s'_G \overset{\psi}{\Vdash} q'_i$, onde $q'_i \in S'_G$. Uma falha de ação ocorre em t quando, para ao menos um $\psi \in \Psi$, existe $(q'_i, \sigma', q'_j) \in T'_G$, onde $q'_j \in S'_G$, $\sigma' \in \Sigma'_G \setminus g$ e temos que $\sigma' \neq \sigma$.*

Um exemplo de falha de ação pode ocorrer na transição de s_2 até s_0 discretizada no grid da Figura 7.3.

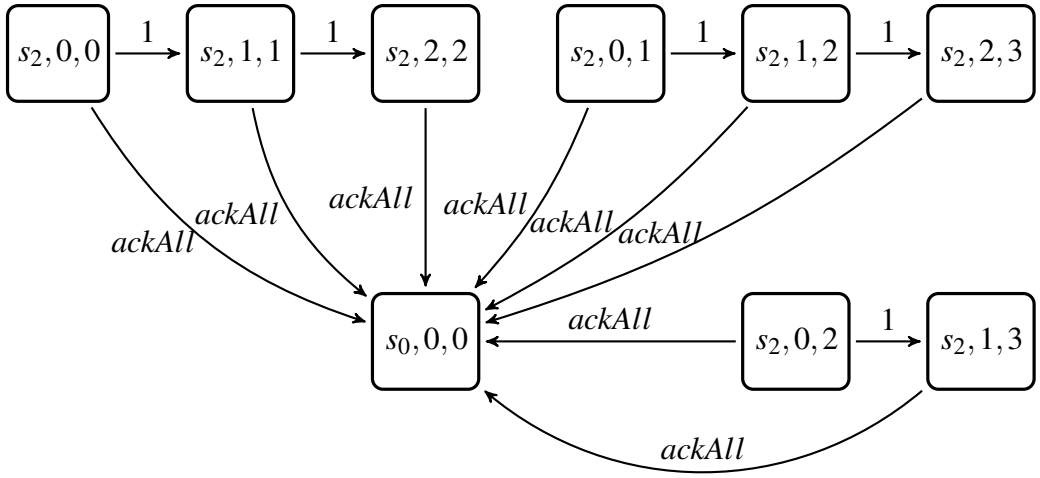

Figura 7.3: Representação em grid da transição entre s_2 e s_0

Adotamos as sequências de movimentos grid, que levam o controle do estado inicial até os estados $(s_2,0,0),(s_2,0,1),(s_2,0,2)$, respectivamente $\psi_1 = (image, sound), \psi_2 = (image,1,sound), \psi_3 = (image,1,1,sound)$. Assuma uma implementação candidata que parte do estado inicial e aplica as sequências de movimentos $\{\psi_1 \cdot (1,error), \psi_2 \cdot (1,error), \psi_3 \cdot (error)\}$, onde as sequências ψ_1, ψ_2, ψ_3 são livres de falhas. A Figura 7.4 mostra o grid da implementação candidata, onde os estados p_0, p_3, p_6 são alcançados pelas sequências de movimentos ψ_1, ψ_2, ψ_3, respectivamente.

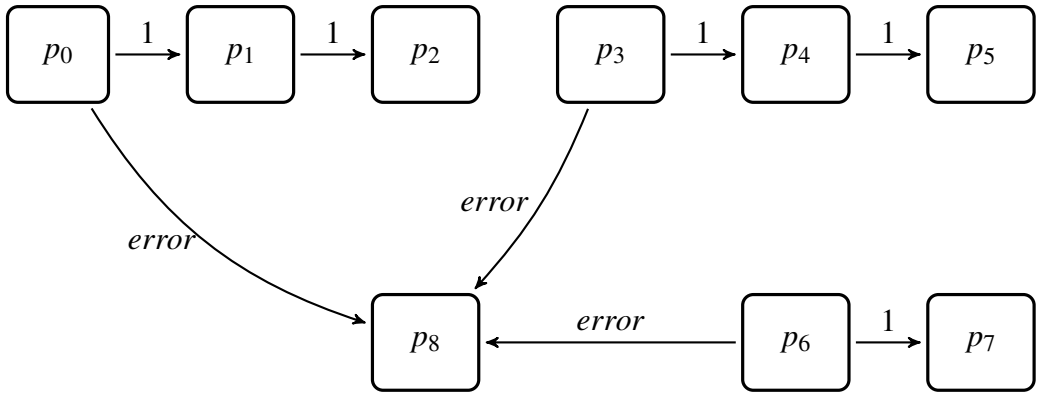

Figura 7.4: Representação em grid da implementação com falha de ação

De acordo com a Definição 10 uma implementação possui uma falha de ação na transição entre s_2 e s_0 quando ao menos umas das sequências em Ψ resulta em uma ação distinta da especificada. Através do modelo de especificação obtemos $\Psi = \{\psi_1, \psi_1 \cdot (1), \psi_1 \cdot (1,1), \psi_2, \psi_2 \cdot (1), \psi_2 \cdot (1,1), \psi_3, \psi_3 \cdot (1)\}$. No estado alvo de cada $\psi \in \Psi$ a especificação deve ter

uma transição com a ação de saída *ackAll*. No entanto, a implementação candidata representada na Figura 7.4 possui uma transição com a ação *error* no estado alvo das sequências ψ_1, ψ_2, ψ_3 que estão contidas em Ψ. Logo, o grid da implementação candidata caracteriza uma falha de ação entre s_2 e s_0.

A falha de ação para transições de ação de entrada não é caracterizada, pois assumimos um grid completo (Ver Hipótese 1). Nesse caso, cada estado possui uma transição para cada ação de entrada. Logo, toda ação de entrada em um determinado estado da implementação candidata possui uma transição na especificação.

7.1.2 Falha de Transferência

A falha de transferência ocorre devido a implementação de uma transição com estado alvo distinto da especificação. Para ilustrar a falha de transferência, tomamos como exemplo o comportamento do estado s_0 do TIOA, representado pela discretização da transição de s_0 até s_1 na Figura 7.5. Além disso, são utilizados os conceitos de projeção e conjunto caracterização.

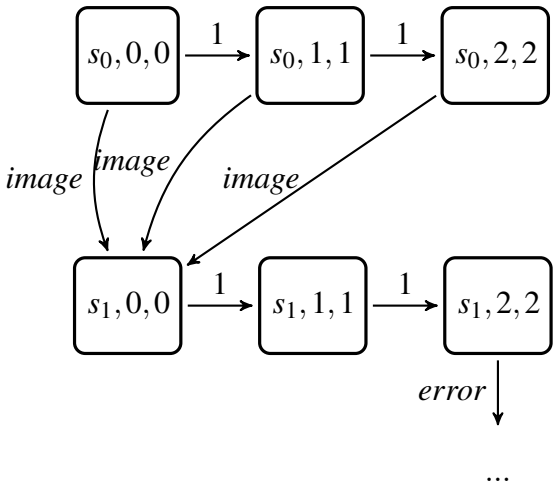

Figura 7.5: Representação em grid da transição entre s_0 e s_1

A projeção é uma função definida para um subconjunto de símbolos do alfabeto do grid sobre uma palavra de grid. Assim, é obtida a sequência de símbolos da palavra de grid exceto àqueles que fazem parte do subconjunto projetado projetado sobre a palavra.

Definição 11. *Seja Σ um alfabeto de ações, $\Gamma \subseteq \Sigma$ um subconjunto de ações e $\psi = (\sigma_1, ..., \sigma_n)$ uma palavra de grid. A projeção de ψ sobre Γ, é uma palavra temporizada denotada por*

$\psi \downarrow_\Gamma = (\phi_1, ..., \phi_k)$, $k \geq 0$, *onde:*

$$\psi_i = \begin{cases} (\sigma_i), \textit{ se } \sigma_i \in \{\Gamma \cup g\} \\ \varepsilon, \textit{ caso contrário.} \end{cases}$$

O conjunto caracterização é um conjunto de sequências de ações de entrada capaz de distinguir se dois estados do grid representam estados distintos do TIOA. Essa distinção é obtida através da sequência de movimentos observada ao aplicar as sequências de entrada desse conjunto a partir dos dois estados. Se a projeção sobre o alfabeto de entrada, que resulta na sequência de ações de saída, é distinta para ao menos um sequência do conjunto caracterização, então os estados do grid representam estados distintos do TIOA. Para a especificação adotada o conjunto de caracterização é $\{(image, 1, 1), (sound, 1, 1)\}$.

Definição 12. *Um conjunto $W = \{w_1, ..., w_n\}$ é um conjunto caracterização para um grid $(S_G, s_G, \Sigma_G, T_G)$ se, para qualquer par de estados $q_i, q_j \in S'_G$, $q_i \in S_i$, $q_j \in S_j$ e $S_i \neq S_j$, existe um $w_k \in W$ tal que $q_i \overset{\omega_i}{\Vdash} p_i$ e $q_j \overset{\omega_j}{\Vdash} p_j$ com $p_i, p_j \in S_G$ temos que $w_k = (\omega_i \downarrow_Y) = (\omega_j \downarrow_Y)$ e $(\omega_i \downarrow_X) \neq (\omega_j \downarrow_X)$ tal que ω_i, ω_j são palavras de grid.*

O conjunto caracterização permite que sejam confrontados o estado alvo de cada transição da implementação com a especificação. Assim, seja T' o conjunto que representa uma transição t do TIOA. A falha de transferência ocorre em t quando o conjunto caracterização é aplicado no estado alvo de cada transição do conjunto T' e a sequência de saída observada na implementação é distinta da especificação.

Definição 13. *Seja $M_G = (S_G, s_G, \Sigma_G, T_G)$ um grid obtido a partir do TIOA de especificação $M = (S, s_0, \Sigma, C, \nu, Inv, T)$ e $M'_G = (S'_G, s'_G, \Sigma'_G, T'_G)$ o grid de uma implementação candidata. Assuma que $T' \subseteq T_G$ é o conjunto de transições que representa uma transição $t = (s_i, \sigma, \delta, \theta, s_j) \in T$. Seja $(q_i, \sigma, q_j) \in T'$, Ψ um conjunto de palavras de grid tal que para cada $\psi \in \Psi$ temos $s_G \overset{\psi}{\Vdash} q_j$ e $s'_G \overset{\psi}{\Vdash} q'_j$ com $q'_j \in S'_G$, e W um conjunto caracterização para M_G. Uma falha de transferência em t ocorre quando, para todo $\psi \in \Psi$, existe um $w_k \in W$ tal que $s'_G \overset{\psi \cdot \omega'}{\vDash} p'_i$ e $s_G \overset{\psi \cdot \omega}{\vDash} p_i$, onde $p_i \in S_G$, $p'_i \in S'_G$, ω e ω' são palavras de grid com $w_k = (\omega \downarrow_Y) = (\omega' \downarrow_Y)$ e $(\omega \downarrow_X) \neq (\omega' \downarrow_X)$.*

Assuma uma implementação candidata que apresenta as sequências de movimentos $\{(image, 1, 1, ackAll), (1, image, 1, 1, ackAll), (1, 1, image, 1, 1, ackAll)\}$ a partir do estado inicial. A Figura 7.6 mostra o grid que representa a implementação candidata. Note que o estado p e $(s_0, 0, 0)$ são os estados iniciais, respectivamente, da implementação e da especificação. Logo, p e $(s_0, 0, 0)$ deveriam apresentar a mesma projeção de saída para o conjunto

de identificação. No entanto, a sequência de movimentos $(image, 1, 1)$ aplicada em p resulta na observação de uma projeção de saída (*ackAll*), diferente da projeção (*error*) observada ao aplicar a mesma sequência em $(s_0, 0, 0)$. Logo, de acordo com a Definição 13, toda transição que possui s_0 como estado alvo na especificação apresenta falha de transferência no grid da implementação candidata.

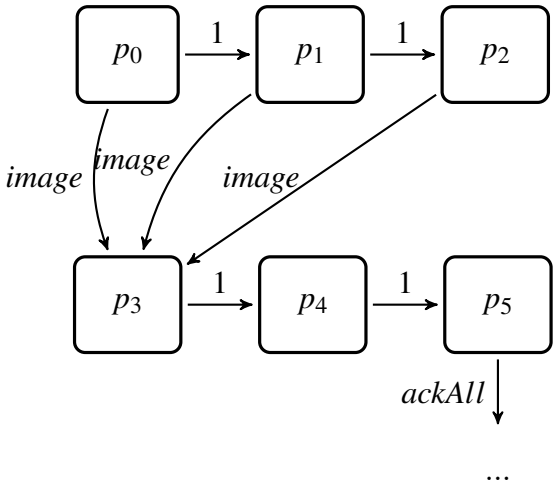

Figura 7.6: Representação em grid da implementação com falha de transferência

7.2 Falhas Dependentes de Tempo

Falhas dependentes do tempo ocorrem quando o modelo TIOA da implementação candidata e da especificação apresentam divergências na evolução do tempo contínuo, sejam por falhas nas condições ou nas reinicializações de relógios. Nesse tipo de falha estão as classes de falha de restrição e de relaxamento nas condições de relógio, que ocorrem devido a implementação de uma condição de relógio com intervalo de habilitação distinto da especificação, e a falha de reinicialização que ocorre devido a implementação de uma função de reinicialização distinta da especificação.

7.2.1 Falha de Restrição de Condição de Relógio

A falha de restrição de condição de relógio ocorre devido a implementação de uma condição mais restrita que a especificada. Quando a condição é mais restrita, a transição deixa de ser habilitada em determinados instantes para o qual foi especificada. Dessa forma, a representação na implementação da transição t do TIOA é obtida por um T' que possui menos transições do que na especificação.

A Definição 14 do modelo proposto dientifica a ocorrência de uma falha de restrição de condição de relógio em t quando o conjunto caracterização é aplicado no estado alvo de cada transição de um subconjunto contido em T' e a sequência de saída observada na implementação é distinta da especificação. Essa caracterização é similar a uma falha de transferência. Isso se deve ao fato da ação de entrada ser aceita em qualquer estado da implementação candidata. Dessa forma, nos instantes em que a transição deixa de ser habilitada, o grid da implementação possui uma transição para a ação de entrada, mas leva a um estado alvo distinto do especificado. Para ilustrar essa falha tomamos como exemplo a transição entre s_0 e s_1 da especificação discretizada anteriormente na Figura 7.5.

Assuma uma implementação candidata que parte do estado inicial e aplica as sequências de movimentos $\{(image,1,1,error),(1,image,1,1,error),(1,1,image,1,1,1)\}$. A Figura 7.7 mostra o grid que representa a implementação candidata.

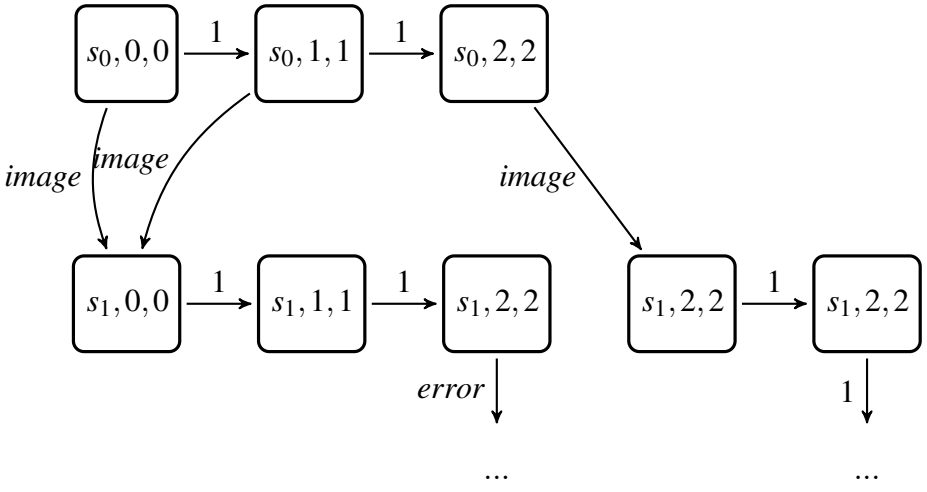

Figura 7.7: Representação em grid da implementação com falha de restrição de relógio

Novamente o conjunto caracterização é $W = \{(image,1,1),(sound,1,1)\}$; e s_0 e p são os estados iniciais da especificação e implementação, logo devem apresentar o mesmo comportamento. No entanto, quando a sequência do conjunto de caracterização é $(image,1,1)$, a projeção de saída na especificação é $(error)$, mas a projeção de saída observada na implementação é (ε). Isso indica que nesse instante de tempo a transição não permanece habilitada como na especificação. Assim, a transição t no TIOA é habilitada para um intervalo mais restrito, representado no grid da implementação candidata, caracterizando uma falha de restrição de condição de relógio entre s_0 e s_1.

Um caso específico pode ocorrer quando $\Psi' = \Psi$. Nesse caso, o conjunto Ψ' carateriza uma falha de restrição de condição e uma falha de transferência. Ambos os casos são

corretos, pois em uma transferência a transição especificada leva a um estado alvo distinto da especificação, logo, a transição especificada possui uma falha de restrição em que a transição deixa de ser habilitada para todo instante especificado. Além disso, a restrição da condição de relógio não é considerada uma falha para ações de saída, pois são ações não controláveis. Essas ações são produzidas pelo sistema de forma autônoma, e não há como garantir que a ação ocorra em cada instante no intervalo de tempo especificado. Por isso, a ocorrência da ação em um instante dentro do intervalo é o suficiente para que a implementação da transição seja correta.

Definição 14. *Seja $M_G = (S_G, s_G, \Sigma_G, T_G)$ um grid obtido a partir do TIOA de especificação $M = (S, s_0, \Sigma, C, \nu, Inv, T)$ e $M'_G = (S'_G, s'_G, \Sigma'_G, T'_G)$ o grid de uma implementação candidata. Assuma que $T' \subseteq T_G$ é o conjunto de transições que representa uma transição $t = (s_i, \sigma, \delta, \theta, s_j) \in T$. Seja $(q_i, \sigma, q_j) \in T'$, Ψ um conjunto de palavras de grid tal que para cada $\psi \in \Psi$ temos $s_G \overset{\psi}{\Vdash} q_j$ e $s'_G \overset{\psi}{\Vdash} q'_j$, $q'_j \in S'_G$, e W um conjunto caracterização para M_G. Uma falha de restrição de condição de relógio em t ocorre quando existe um subconjunto $\Psi' \subseteq \Psi$ tal que, para todo $\psi' \in \Psi'$, existe um $w_k \in W$ tal que $s'_G \overset{\psi' \cdot \omega'}{\vDash} p'_i$, $s_G \overset{\psi' \cdot \omega}{\vDash} p_i$, onde ω' e ω são palavras de grid com $w_k = (\omega' \downarrow_Y) = (\omega \downarrow_Y)$ e $(\omega \downarrow_X) \neq (\omega' \downarrow_X)$.*

7.2.2 Falha de Relaxamento de Condição de Relógio

A falha de relaxamento de condição de relógio ocorre devido a implementação de uma condição menos restrita que a especificada. A condição menos restrita habilita a transição para determinados instantes os quais não foram especificados. Dessa forma, a representação na implementação da transição t do TIOA é obtida por um T' que possui mais transições do que na especificação.

A falha de relaxamento de condição de relógio ocorre quando existe uma transição no grid entre dois estados p e q, rotulada com uma ação de saída, que não existe no grid de especificação. Como assumimos um grid com saídas isoladas (ver Hipótese 1), é possível mostrar que a transição do TIOA com falha de relaxamento é àquela que parte do estado do TIOA que p representa e que possui a mesma ação da transição entre p e q. Note que caso uma transição com essas características não exista, há uma falha de relaxamento de uma condição *false*, logo a transição não é representada na especificação. Para ilustrar essa falha adotamos como exemplo a transição entre s_1 e s_0 discretizada no grid da Figura 7.8.

Assuma uma implementação candidata com sequências de movimentos $\{\psi \cdot 1, \psi \cdot (1, error), \psi \cdot (1, 1, error)\}$, onde a sequência ψ é livre de falhas, $\psi = (image)$ é a sequên-

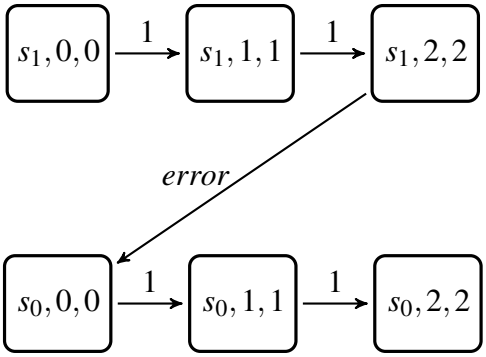

Figura 7.8: Representação em grid da transição entre s_1 e s_0

cia de movimentos que leva o grid da especificação do estado inicial até o estado $(s_1,0,0)$.. A Figura 7.9 mostra o grid que representa a implementação candidata, onde o estado p_0 é alcançado pela sequência de movimentos ψ.

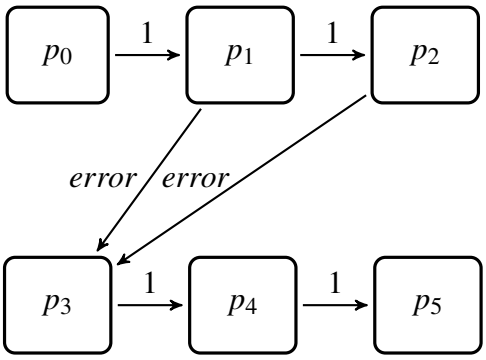

Figura 7.9: Representação em grid da implementação candidata com falha de relaxamento de condição de relógio

A Definição 15 identifica falhas de relaxamento através de transições extras rotuladas com ação de saída no grid da implementação candidata. A sequência de movimentos $\psi \cdot (1, error)$, executada na implementação candidata, não é possível no grid de especificação, pois não existe uma transição rotulada *error* como a transição entre p_1 e p_3. Dessa forma, essa transição representa a ocorrência da ação em um instante não previsto na especificação. Como o estado p_1 representa o estado s_1 do TIOA, a transição do TIOA que habilita no instante não especificado é a transição que parte de s_1 com ação *error*. Logo, o grid da implementação candidata caracteriza uma falha de relaxamento de condição de relógio entre s_1 e s_0.

A falha de relaxamento de condição de relógio não ocorre para ações de entrada, pois assumimos um grid de especificação completo (ver Definição 1). Logo não é possível que um estado possua uma transição de ação de entrada que não é especificada.

Definição 15. *Seja $M_G = (S_G, s_G, \Sigma_G, T_G)$ um grid com saídas isoladas obtido a partir do TIOA de especificação $M = (S, s_0, \Sigma, C, \nu, Inv, T)$ e $M'_G = (S'_G, s'_G, \Sigma'_G, T'_G)$ o grid de uma implementa-*

ção candidata. Assuma que $T' \subseteq T_G$ é o conjunto de transições que representa uma transição $t = (s_i, \sigma, \delta, \theta, s_j) \in T$. Seja $(q_i, \sigma, q_j) \in T'$, Ψ um conjunto de palavras de grid tal que para cada $\psi \in \Psi$ temos $s_G \overset{\psi}{\Vdash} q_i$ e $s'_G \overset{\psi}{\Vdash} q'_i$ com $q'_i \in S'_G$, Ψ' um conjunto de palavras de grid tal que para cada $\psi' \in \Psi'$ temos $s_G \overset{\psi'}{\Vdash} q$ e $s'_G \overset{\psi'}{\Vdash} q'$ com $q \in S_i$, $q' \in S_G$. Uma falha de relaxamento de condição de relógio ocorre em t quando para um conjunto $\Psi'' = \Psi' - \Psi$, existe ao menos um $\psi'' \in \Psi''$ tal que $s'_G \overset{\psi''.\sigma}{\models} p'_i$ e $s_G \overset{\psi''.\sigma}{\models} p_i$, com $\sigma' \in Y \cup \{g\}$ e temos $\sigma \neq \sigma'$.

7.2.3 Falha de Reinicialização de Relógio

A falha de reinicialização de relógio ocorre quando uma função de reinicialização é implementada de maneira distinta da especificada. Quando a função de reinicialização é distinta, a configuração da interpretação de relógio no estado alvo no TIOA especificação é distinta.

Para ilustrar essa falha apresentamos dois exemplos: quando a falha resulta em um relógio com valor maior que o esperado; e quando a falha resulta em um valor menor que o esperado. Para o primeiro exemplo adotamos a transição entre os estados s_0 e s_1, onde a propagação deve ocorrer quando a falha de reinicialização resulta em uma interpretação de relógio na implementação maior que o da especificação. A Figura 7.5 mostra o grid representando a transição discretizada entre s_0 e s_1.

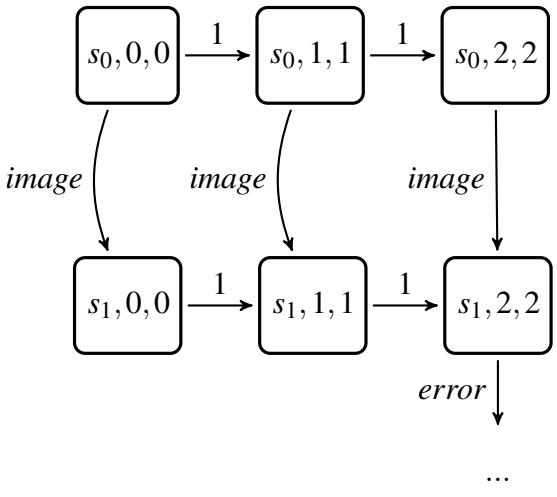

Figura 7.10: Representação em grid da implementação que não reinicia c_1 e c_2

Assuma que a transição de s_0 até s_1 deixe de reiniciar os relógios c_1 e c_2. O grid que representa a transição de s_0 até s_1 sem a reinicialização é mostrado na Figura 7.10

No grid da Figura 7.10 é possível aplicar as sequências de movimentos {

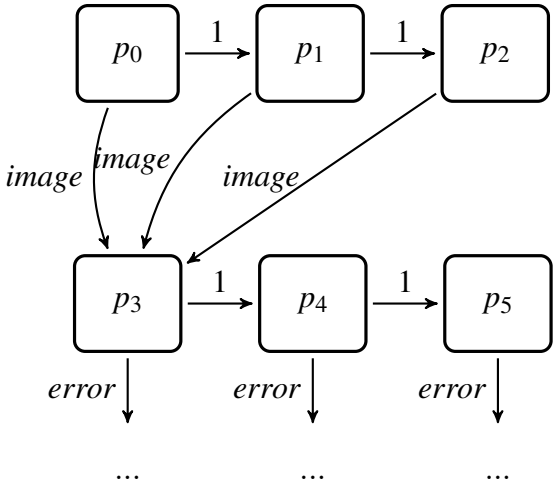

Figura 7.11: Representação em grid da implementação com propagação de falha

$(image, 1, 1, error), (1, image, 1, error), (1, 1, image, error)\}$. Esse conjunto de sequências também é possível no grid da Figura 7.11 que não representa uma implementação que deixa de reinicializar os relógios. Logo, a simples aplicação das sequências $\{(image, 1, 1, error)$, $(1, image, 1,$ $error)$, $(1, 1, image, error)\}$ numa implementação não permite determinar se o grid que a representa é da Figura 7.10 ou da Figura 7.11. Por conta disso, a falha de reinicialização entre s_0 e s_1 não é detectada de maneira precisa, porém, este tipo de falha propaga falhas de outras classes.

A abordagem adotada na Definição 16 se baseia na propagação de uma falha de reinicialização em uma transição t_i para outras classes de falhas em transições t_j. Assuma uma transição $t_i = (s, \sigma_i, \delta_i, \theta_i, s')$ do TIOA com falha de reinicialização sobre um relógio c_i e um conjunto T_t de transições do TIOA que podem ser habilitadas após t_i. Seja $t_j = (r, \sigma_j, \delta_j, \theta_j, r')$ uma transição em T_t, uma falha de reinicialização ocorre em t_i quando a reinicialização $v \oplus \theta_i$ resulta em uma interpretação distinta da especificada. Essa falha é propagada para todo t_j que é habilitado após uma sequência de movimentos na qual não há reinicialização de c_i e existe uma condição para c_i em δ_j.

Ao deixar de reinicializar os relógios, a permite uma interpretação de relógio maior do que a especificada e o intervalo de tempo em que outras transições que utilizem as interpretações dos relógios que apresentam falha permanecem habilitadas é menor que o especificado. Esse comportamento, caracteriza uma restrição de condição de relógio no limite superior. Se após a transição t_i o estado r é alcançado com uma interpretação de relógio menor ao limite inferior da condição de relógio que habilita t_j, o tempo do sistema evolui durante um intervalo no estado r até ocorrer a transição t_j. Quando $v \oplus \theta_i$ resulta em uma interpretação de relógio maior do que a especificada, esse intervalo em que o tempo evolui será menor. Logo, a

condição de relógio é relaxada no limite inferior.

Definição 16. *Seja $M = (S, s_0, \Sigma, C, \nu, Inv, T)$ o TIOA de especificação e $M' = (S', s'_0, \Sigma', C', \nu', Inv', T')$ uma implementação candidata. Assuma que $\nu \oplus \theta_i = k$, onde $k \in \mathbb{Q}_{\geq}$ é a interpretação de relógio ν após a execução de $t_i = (s_i, \sigma_i, \delta_i, \theta_i, r_i) \in T$ com δ_i na forma $\tau_i \leq c_i \leq \tau_j$ e que $T_t \subset T$ é o subconjunto de transições que podem ser habilitadas após t_i, onde cada $t_j = (s_j, \sigma_j, \delta_j, \theta_j, r_j) \in T_t$ com $\delta_j = \tau_x \leq c_i \leq \tau_y$. Seja ψ uma palavra temporizada tal que $s_0 \overset{\psi}{\vDash} s_i$ e Ω um conjunto de palavras temporizadas onde para todo $\omega \in \Omega$, $s_i \overset{\omega}{\vDash} s_j$. Assuma ainda que dada a projeção $\omega' = \omega \downarrow_{(X \cup Y)}$, com $\omega' = (\omega'_1, ...\omega'_n)$ temos $\Delta_k \in \mathbb{Q}_{\geq}$ com $\Delta_k = \omega'_1 + ... + \omega'_n$. Uma falha de reinicialização ocorre em t_i quando $t'_i = (s'_i, \sigma'_i, \delta'_i, \theta'_i, r'_i) \in T'$, tal que $\nu \oplus \theta'_i = k'$, com $k' \in \mathbb{Q}_{\geq}$ e temos $k' \neq k$. Essa falha é propagada para todo $t'_j = (s'_j, \sigma'_j, \delta'_j, \theta'_j, r'_j) \in T'$ com δ'_j na forma $\tau'_i \leq c'_i \leq \tau'_j$ se:*

- *$k' < k$ e $k + \Delta_k < \tau_i$, então com $s_0 \overset{\psi \cdot \omega}{\vDash} s_j$ temos $\nu(c'_i) = k' + \Delta_k$ com $k' + \Delta_k < k + \Delta_k$, $\tau_i - (k' + \Delta_k) > \tau_i - (k + \Delta_k)$ e $\tau_j - (k' + \Delta_k) > \tau_j - (k + \Delta_k)$, resultando em um τ'_i mais restrito e um τ'_j menos restrito.*

- *$k' < k$ e $k' + \Delta_k > \tau_i$, então com $s_0 \overset{\psi \cdot \omega}{\vDash} s_j$ temos $\nu(c'_i) = k' + \Delta_k$ com $k' + \Delta_k < k + \Delta_k$ e $\tau_j - (k' + \Delta_k) > \tau_j - (k + \Delta_k)$, resultando em um τ'_j menos restrito.*

- *$k' > k$ e $k' + \Delta_k < \tau_i$, então com $s_0 \overset{\psi \cdot \omega}{\vDash} s_j$ temos $\nu(c'_i) = k' + \Delta_k$, como $k' + \Delta_k > k + \Delta_k$, $\tau_i - (k' + \Delta_k) < \tau_i - (k + \Delta_k)$ e $\tau_j - (k' + \Delta_k) < \tau_j - (k + \Delta_k)$, resultando em um τ'_i menos restrito e um τ'_j mais restrito.*

- *$k' > k$ e $k + \Delta_k > \tau_i$, então com $s_0 \overset{\psi \cdot \omega}{\vDash} s_j$ temos $\nu(c'_i) = k' + \Delta_k$ com $k' + \Delta_k > k + \Delta_k$ e $\tau_j - (k' + \Delta_k) < \tau_j - (k + \Delta_k)$ resultando em um τ'_j mais restrito.*

Como dado na Definição 15, o grid da Figura 7.11 caracteriza o relaxamento do limite inferior da condição de relógio da transição através das transições extras que partem dos estados p_3, p_4, p_5. Essas transições são exercitadas pelas sequências $\{(1, image, 1, error)$, $(1, 1, image, error)\}$, que não são possíveis no grid da especificação. Já a propagação de restrição do limite superior da condição de relógio da transição não é considerada uma falha, pois a ação de saída não é controlável.

Para ilustrar o segundo caso tomamos como exemplo a transição entre os estados s_1 e s_2. Essa transição exemplifica a reinicialização que resulta em uma interpretação menor que a especificada. A Figura 7.12 mostra o grid representando a transição discretizada de s_1 a s_2, e de s_2 a s_0.

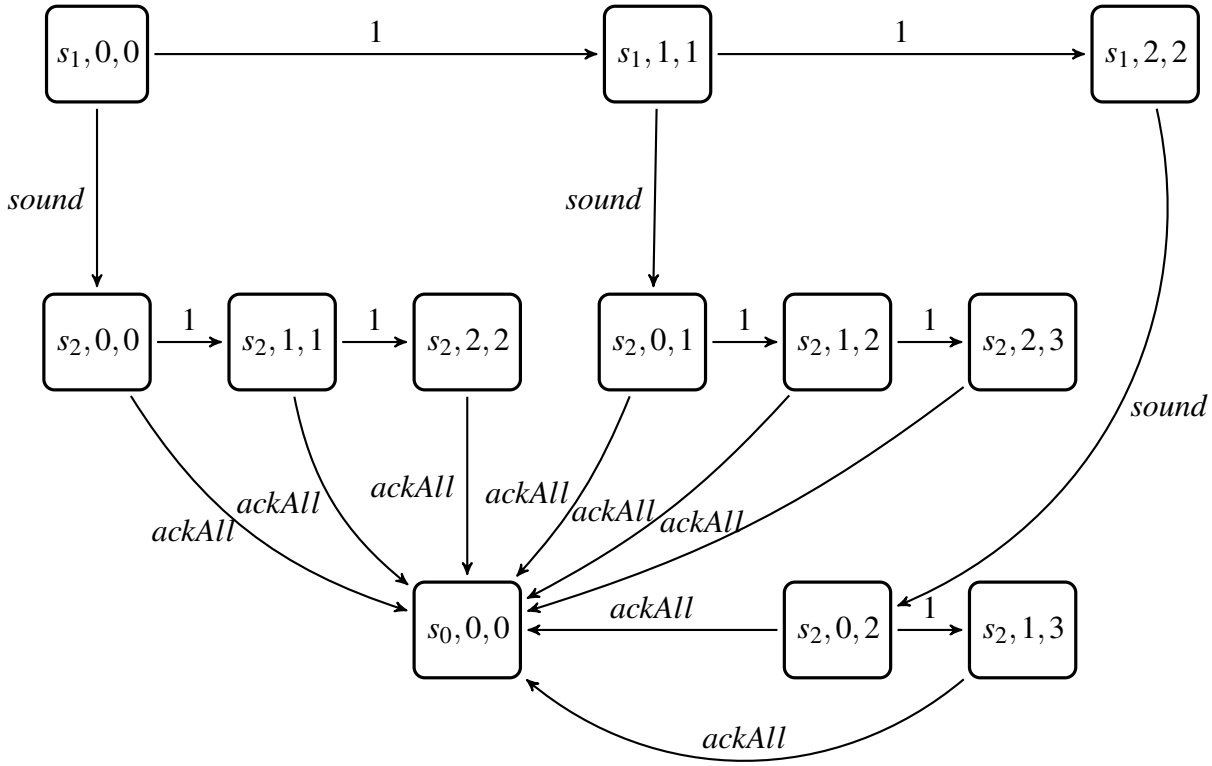

Figura 7.12: Representação em grid da transição entre s_1 e s_2, e entre s_2 e s_0

Assuma que a transição de s_1 até s_2 reinicializa o relógio c_2 para 0. O grid que representa a transição de s_1 até s_2 reinicializando c_1 e c_2 é mostrado na Figura 7.13.

O grid da Figura 7.13, é executa as sequências de movimentos $\{\psi \cdot (sound, 1, 1, ackAll), \psi \cdot (1, sound, 1, 1, ackAll), \psi \cdot (1, 1, sound, 1, 1 ackAll)\}$. Esse conjunto de sequências também é possível no grid da Figura 7.14. Assim, a simples aplicação das sequências de movimentos $\{\psi \cdot (sound, 1, 1, ackAll), \psi \cdot (1, sound, 1, 1, ackAll), \psi \cdot (1, 1, sound, 1, 1 ackAll)\}$, numa implementação candidata, não permite determinar se o grid que a representa é da Figura 7.13 ou da Figura 7.14. Por isso, a falha de reinicialização entre s_1 e s_2 não é detectada.

Se a implementação da reinicialização $\nu \oplus \theta_i$ em t_i resulta em uma interpretação de relógio menor do que a especificada, o intervalo de tempo em que t_j habilita é maior que o especificado. Logo, a condição de relógio é relaxada no limite superior. Se após a transição t_i o estado r é alcançado com uma interpretação de relógio menor ao limite inferior da condição de relógio que habilita t_j, o tempo do sistema evolui durante um intervalo no estado r até ocorrer a transição t_j. Quando $\nu \oplus \theta_i$ resulta em uma interpretação de relógio menor do que a especificada, esse intervalo em que o tempo evolui será maior. Logo a condição de relógio é restringida no limite inferior da condição de relógio.

Como dado na Definição 15 o grid da Figura 7.14 caracteriza uma falha de

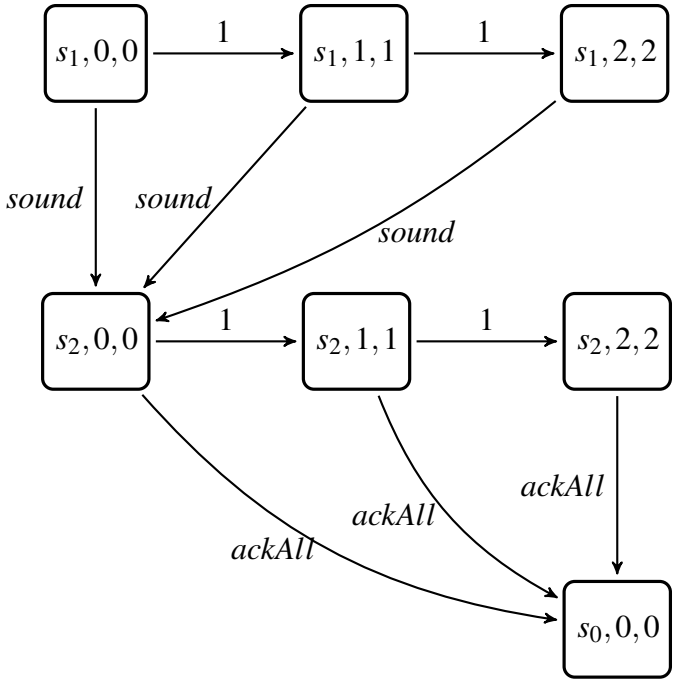

Figura 7.13: Representação em grid da implementação que reinicia c_2

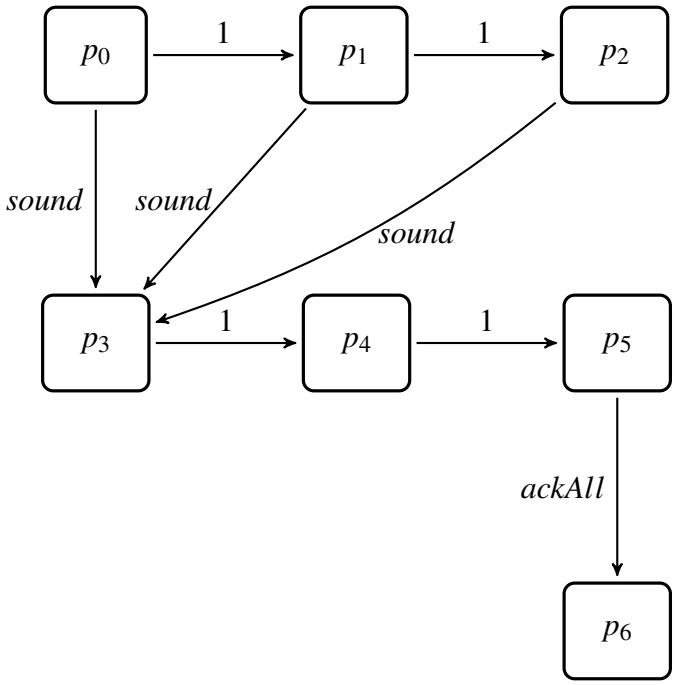

Figura 7.14: Representação em grid da implementação candidata com propagação de falha

relaxamento do limite superior através da transição entre os estados p_5 e p_6 que representa uma transição extra na linha de tempo iniciado no estado $(s_2,0,2)$ da Figura 7.14. Essa transição é exercitada na sequência de movimentos $\{\psi \cdot (1,1,sound,1,1ackAll)\}$ representa esse instante de tempo onde a transição é habilitada na implementação candidata e não é habilitada na especificação. Já a propagada da falha de restrição de limite inferior na transição de s_2 para s_0 não é

considerada uma falha, pois *ackAll* é uma ação de saída.

No exemplo da Figura 7.14 também é mostrado o caso particular de quando $t_j \in T_t$ possui uma condição de relógio composta identificado na Definição 17. Quando a condição é da forma $\delta_i \vee \delta_j$, a falha de reinicialização é propagada para t_j se a interseção do intervalo de habilitação de δ_i e δ_j é diferente da especificicação devido a propagação de falhas em alguma das duas condições. Já para $t_j \in T_t$ com condição de relógio composta na forma $\delta_i \wedge \delta_j$, a falha de reinicialização é propagada para t_j se a disjunção dos intervalos de habilitação de δ_i e δ_j apresenta divergência em relação a especificação. A propagação de falha para δ_i e δ_j é dada como nas Definições 16 e 17. Note que δ_i e δ_j também podem ser condições compostas.

Definição 17. *Seja* $M = (S, s_0, \Sigma, C, \nu, Inv, T)$ *o TIOA de especificação e* $M' = (S', s'_0, \Sigma', C', \nu', Inv', T')$ *uma implementação candidata. Assuma que* $T_t \subset T$ *é o subconjunto de transições que podem ser habilitadas após a execução de* t_i*. Uma falha de reinicialização ocorre em* t_i *quando* $t'_i = (s'_i, \sigma'_i, \delta'_i, \theta'_i, r'_i) \in T'$*, tal que* $\nu(c_i) \oplus \theta'_i(c_i) = k'$*, com* $k' \in \mathbb{Q}_{\geq}$ *e temos* $k' \neq k$*. Essa falha é propagada para todo* t'_j *que possui condição de relógio na forma:*

1. $\delta_i \vee \delta_j$*, quando ao menos uma das condições* δ_i, δ_j *apresenta propagação de falha que resulta em uma representação do intervalo de habilitação* $(\delta_i \cup \delta_j) \setminus (\delta_i \cap \delta_j)$ *distinta da especificação. A propagação de falha para* δ_i *e* δ_j *é dada como nas Definições 16 e 17.*

2. $\delta_i \wedge \delta_j$*, quando ao menos uma das condições* δ_i, δ_j *apresenta propagação de falha que resulta em uma representação do intervalo de habilitação* $(\delta_i \cap \delta_j)$ *distinta da especificação. A propagação de falha para* δ_i *e* δ_j *é dada como nas Definições 16 e 17.*

8 CONCLUSÃO

A abordagem de teste baseado em modelos para sistemas de tempo real tem sido explorada em diversos trabalhos teóricos. Um dos formalismos mais utilizados para representar a evolução contínua do tempo e as características reativas desses sistemas é o modelo TIOA. Uma das tarefas do teste baseado no modelo TIOA é a de análise de cobertura e de detecção de falhas. Um modelo de falhas para um formalismo temporizado tem a importante função de caracterizar potenciais falhas em sistemas de tempo real. Contudo, existe uma dificuldade prática em lidar com modelos de tempo contínuo. Por isso, alternativas baseada em modelos discretizados são adotadas para realizar uma análise de detecção de falhas nestes sistemas.

Alguns trabalhos foram propostos nessa vertente, como o modelo de falhas para TIOA baseado na representação de autômato de região. No entanto, esse modelo de falhas é restrito à abordagens de discretização que adotam granularidades baseada no número de regiões de relógio. Como o número de regiões de relógio aumenta de acordo com o número de relógios do TIOA, essa discretização resulta num espaço de estado exponencial com a necessidade de granularidades muito fina. Logo, a utilização desses métodos na prática se torna inviável. Numa abordagem de discretização mais recente é possível que um espaço de estado mais gerenciável seja obtido através de uma escolha adequada da granularidades adotada no processo de discretização. Porém, a eficácia de detecção sobre os conjuntos de teste obtidos por esse método foi proposta apenas para falhas específicas do modelo grid através do conceito de propostas de teste.

Neste cenário um modelo de falhas foi proposto para TIOA baseado em autômato grid, com o objetivo de caracterizar cada classe de falha de um TIOA através do modelo de autômato grid. Esse modelo de falhas possibilita a análise de detecção de falhas, de maneira generalizada, em sistemas de tempo real. O modelo de falhas proposto é capaz de caracterizar a ocorrência de falhas de ação, falhas de transferência, e falhas de restrição e relaxamento de condição de relógio. Falhas de reinicialização de relógio não são caracterizadas diretamente por uma classe de falha. No entanto, a ocorrência da falha reinicialização pôde ser identificada pela propagação de falhas que resultam em outras classes de falha identificadas pelo modelo

proposto.

Para exemplificar a aplicação do modelo de falhas, um estudo de caso foi realizado para um protocolo multimídia. Nesse estudo são apresentados exemplos de implementações candidatas representadas por autômatos grid para simular a identificação de cada classe de falha.

Como trabalho futuro pretende-se desenvolver um método de geração de conjuntos de teste baseado no modelo de falhas proposto. Espera-se também que esse modelo de falhas forneça as bases para que outros trabalhos voltados à análise de cobertura de falhas e extração de conjuntos de teste sejam desenvolvidos no futuro.

REFERÊNCIAS

ALUR, R. Timed automata. In: *Proceedings of the 11th International Conference on Computer Aided Verification*. London, UK: Springer-Verlag, 1999. (CAV '99), p. 8–22. ISBN 3-540-66202-2. Disponível em: <http://portal.acm.org/citation.cfm?id=647768.733787>.

ALUR, R.; DILL, D. L. A theory of timed automata. *Theor. Comput. Sci.*, Elsevier Science Publishers Ltd., Essex, UK, v. 126, p. 183–235, April 1994. ISSN 0304-3975. Disponível em: <http://portal.acm.org/citation.cfm?id=180782.180519>.

BERTOLINO, A. Software testing research: Achievements, challenges, dreams. In: *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007. p. 85–103. ISBN 0-7695-2829-5.

BLACKBURN R. BUSSER, A. N. M. Why model-based test automation is different and what you should know to get started. In: *International Conference on Practical Software Quality and Testing*. Washington, DC, USA: PSQT/PSTT'2004 East, 2004.

BONIFACIO, A.; MOURA, A.; SIMAO, A. da S. A generalized model-based test generation method. In: *Software Engineering and Formal Methods, 2008. SEFM '08. Sixth IEEE International Conference on*. [S.l.: s.n.], 2008. p. 139 –148.

BONIFACIO, A. L.; MOURA, A. V. *A New Timed Discretization Method for Automatic Test Generation for Timed Systems*. [S.l.], September 2009. 33 p. Http://www.ic.unicamp.br/∼reltech/2009/09-31.pdf.

BONIFáCIO, A. L.; MOURA, A. V. A new method for testing timed systems. *Software Testing, Verification and Reliability*, John Wiley AND Sons, Ltd., p. n/a–n/a, 2011. ISSN 1099-1689. Disponível em: <http://dx.doi.org/10.1002/stvr.454>.

CHOW, T. S. Testing software design modeled by finite-state machines. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 4, n. 3, p. 178–187, 1978. ISSN 0098-5589.

DOI JR, G.; BONIFáCIO, A. Detecção de falhas em autômatos grid. In: *SBRC 2012 - WTF ()*. Ouro Preto, MG: [s.n.], 2012.

EN-NOUAARY, A. A scalable method for testing real-time systems. *Software Quality Control*, Kluwer Academic Publishers, Hingham, MA, USA, v. 16, p. 3–22, March 2008. Disponível em: <http://portal.acm.org/citation.cfm?id=1331426.1331458>.

EN-NOUAARY, A.; DSSOULI, R.; KHENDEK, F. Timed wp-method: Testing real-time systems. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 28, p. 1023–1038, November 2002. ISSN 0098-5589. Disponível em: <http://portal.acm.org/citation-.cfm?id=630831.631295>.

EN-NOUAARY, A.; HAMOU-LHADJ, A. A boundary checking technique for testing real-time systems modeled as timed input output automata (short paper). In: *Quality Software, 2008. QSIC '08. The Eighth International Conference on*. [S.l.: s.n.], 2008. p. 209 –215. ISSN 1550-6002.

EN-NOUAARY, A.; KHENDEK, F.; DSSOULI, R. Fault coverage in testing real-time systems. In: *Real-Time Computing Systems and Applications, 1999. RTCSA '99. Sixth International Conference on*. [S.l.: s.n.], 1999. p. 150 –157.

FUJIWARA, S. et al. Test selection based on finite state models. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA, v. 17, p. 591–603, June 1991. ISSN 0098-5589. Disponível em: <http://portal.acm.org/citation.cfm?id=126218.126234>.

GILL, A. *Introduction to the theory of finite-state machines*. McGraw-Hill, 1962. (McGraw-Hill electronic sciences series). Disponível em: <http://books.google.com.br-/books?id=IDhSAAAAMAAJ>.

HIERONS, R. M. et al. Using formal specifications to support testing. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 41, n. 2, p. 1–76, 2009. ISSN 0360-0300.

KRICHEN, M.; TRIPAKIS, S. Black-box conformance testing for real-time systems. In: *In 11th International SPIN Workshop on Model Checking of Software (SPIN'04), volume 2989 of LNCS*. [S.l.]: Springer, 2004. p. 109–126.

SPRINGINTVELD, J.; VAANDRAGER, F.; D'ARGENIO, P. R. Testing timed automata. *Theor. Comput. Sci.*, Elsevier Science Publishers Ltd., Essex, UK, v. 254, p. 225–257, March 2001. ISSN 0304-3975. Disponível em: <http://dx.doi.org/10.1016/S0304-3975(99)00134-6>.

TRETMANS, J. Formal methods and testing. In: HIERONS, R. M.; BOWEN, J. P.; HARMAN, M. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2008. cap. Model based testing with labelled transition systems, p. 1–38. ISBN 3-540-78916-2, 978-3-540-78916-1. Disponível em: <http://dl.acm.org/citation.cfm?id=1806209% -.1806210>.

UTTING, M.; LEGEARD, B. *Practical Model-Based Testing: A Tools Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 0123725011, 9780080466484.

WAGNER, F. *Modeling Software With Finite State Machines: Practical Approach*. CRC PressINC, 2006. (Modeling Software with Finite State Machines: A Practical Approach). ISBN 9780849380860. Disponível em: <http://books.google.com.br/books?id=5xbQqABp_OAC>.

WEIGLHOFER, M.; WOTAWA, F. Improving coverage based test purposes. In: *Quality Software, 2009. QSIC '09. 9th International Conference on*. [S.l.: s.n.], 2009. p. 219 –228. ISSN 1550-6002.

TRABALHOS PUBLICADOS PELO AUTOR

1. DOI JR, G.; BONIFÁCIO, A., A Tool to Support Model-Based Testing Activities, SBESC 2011, p. 21-26, isbn: 978-0-7695-4929-3, (Qualis CC 2012, B4).

2. DOI JR, G.; BONIFÁCIO, A. Detecção de falhas em autômatos grid, SBRC 2012 - WTF, p. 131-144, issn: 2177-496X, (Qualis CC 2012, B4).