Detecção de Falhas em Autômatos Grid

Gilson Doi Junior[1], **Adilson Luiz Bonifácio**[1]

[1]Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 6.001 – 86.051-980 – Londrina – PR – Brazil

doijunior@gmail.com, bonifacio@uel.br

Abstract. *Real-time systems are, in general, critical systems interacting with the environment through I/O events in which time restrictions are deemed important. Model-based testing is a widely used rigorous approach to provide more accuracy on testing real-time systems. Formal techniques that can handle continuous time evolution is a challenge. Therefore several approaches rely on discretization methods to obtain discretized models, called grid automata, corresponding to original timed models. Test suites can be extracted from grid automata to verify whether given implementations adhere to the specification. However a very important issue on testing real-time systems is the fault coverage. In this work we address a fault coverage model for grid automata following the fault models for finite state machines and timed models.*

Resumo. *Sistemas de tempo real são, em geral, sistemas críticos que interagem com o ambiente externo através de eventos de E/S onde as restrições de tempo são consideradas importantes. Teste baseado em modelos é uma abordagem rigorosa amplamente usada para dar mais precisão no teste de sistemas de tempo real. Técnicas formais que podem lidar com a evolução contínua do tempo é um desafio. Por isso, várias abordagens contam com métodos de discretização para obter modelos discretizados, chamados autômatos grid, correspondentes aos modelos temporizados originais. Conjuntos de teste podem ser extraídos a partir de autômatos grid para verificar se implementações dadas se aderem à especificação. Contudo, uma questão muito importante no teste de sistemas de tempo real é a cobertura de falhas. Neste trabalho abordamos um modelo de cobertura de falhas para autômatos grid seguindo os modelos de falha para máquinas de estados finitos e modelos temporizados.*

1. Introdução

Os avanços tecnológicos de hardware e software tem possibilitado uma maior adesão às técnicas de automatização de tarefas relacionadas ao desenvolvimento de sistemas. Entre esses sistemas estão os chamados sistemas críticos onde a ocorrência de uma falha pode resultar em prejuízos financeiros e danos irreparáveis. Técnicas rigorosas, em especial na fase de teste, são de extrema importância para se minimizar a ocorrência de falhas em sistemas dessa natureza.

Teste baseado em modelos [M. Blackburn 2004, Utting and Legeard 2007] é uma abordagem rigorosa de teste que se apoia em métodos e modelos formais. Sua principal característica é a utilização de uma estrutura formal para especificar sistemas e extrair conjuntos de teste capazes de identificar o maior número de falhas possíveis. Porém,

ainda existem muitos desafios na utilização das técnicas de teste baseado em modelos, apesar de se tratar de um assunto bem estabelecido e com vários estudos já realizados. Um desses desafios é lidar com sistemas de tempo real [Alur 1999, Alur and Dill 1994], onde o instante de tempo influencia o comportamento e determina a ocorrência das ações do sistema. Lidar com o aspecto de tempo contínuo usando modelos formais se torna ainda mais difícil quando o foco é a atividade de teste para tais sistemas.

Alguns trabalhos nessa linha [En-Nouaary et al. 2002, Springintveld et al. 2001, En-Nouaary and Hamou-Lhadj 2008, Bonifácio and Moura 2011] focam técnicas de discretização para lidar com a evolução contínua de tempo. Os modelos discretizados, também chamados de grid, capturam aspectos de tempo, permitindo a análise e detecção de falhas importantes dos sistemas.

Já as estratégias de detecção de falhas podem ser avaliadas conforme os diferentes objetivos de teste, indicando qual a cobertura de falhas provida. As avaliações podem ser qualitativas, através de propostas de teste [Bonifácio and Moura 2011, Weiglhofer and Wotawa 2009], ou quantitativas, através de modelos de falha [Chow 1978, En-Nouaary et al. 1999].

Este trabalho pretende analisar a cobertura de falhas para autômatos grid obtidos conforme técnicas mais recentes de discretização [Bonifácio and Moura 2011, Bonifacio and Moura 2009]. Em geral, autômatos grid recaem no problema de explosão combinatória de estados, comumente encontrado em abordagens que tratam de sistemas temporizados. Grids mais gerenciáveis, então, permitem o estudo e a análise da cobertura de falhas para sistemas de tempo real. A estrategia para se identificar falhas no modelo grid se baseia nos modelos de falha propostos para as Máquinas de Estados Finitos (MEFs) [Chow 1978] e para os *Timed Input/Output Automata* (TIOA) [En-Nouaary et al. 1999].

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta a abordagem de teste baseado em modelos e alguns formalismos relacionados a proposta. Trabalhos relacionados neste contexto sobre teste em sistemas de tempo real são descritos na Seção 3. A Seção 4 apresenta a noção de conformidade e as importantes relações conhecidas para modelos convencionais e temporizados. O conceito de modelo de falhas, bem como os modelos de falha específicos usados como base neste trabalho são apresentados na Seção 5. A Seção 6 descreve uma análise preliminar da capacidade de detecção de falhas em modelos discretizados. As considerações finais e direções futuras de trabalho são apresentadas na Seção 7.

2. Teste Baseado em Modelos

Abordagens tradicionais de teste, em geral, modelam sistemas através de métodos convencionais, incluindo nesses casos o apoio efetivo de linguagem natural. Esse tipo de abordagem possibilita interpretações ambíguas, podendo resultar em produtos com erros. Para se evitar tais ambiguidades as técnicas de teste baseado em modelos utilizam linguagens bem definidas, como o UML [Utting and Legeard 2007], ou modelos formais com forte embasamento matemático. Formalismos são adotados para retirar a ambiguidades do modelo de especificação e possibilitar os métodos de geração de casos de teste. Essa abordagem diminui drasticamente a possibilidade de erros de interpretação, principalmente no levantamento de requisitos [Hierons et al. 2009].

Outro problema encontrado nas metodologias tradicionais é a seleção do conjunto de testes. Muitas vezes essa seleção é realizada de maneira manual e arbitrária, não havendo uma análise mais precisa da eficiência e da eficácia de detecção das falhas. O modelo formal de especificação facilita a análise da estrutura do sistema, mostrando a cobertura provida pela conjunto de testes, além de possibilitar técnicas que automatizem a extração e execução desses testes. Utilizando modelos formais para especificar requisitos de sistema e definir uma rotina rigorosa de extração e execução do conjunto de testes é possível reduzir consideravelmente o número de falhas não detectadas, proporcionando uma maior confiabilidade aos métodos [M. Blackburn 2004].

Existem diversos métodos e técnicas que seguem essa abordagem de teste, porém divergindo em diferentes aspectos, como no modelo formal adotado ou nas características da rotina de extração e execução dos testes. Apesar dessas diferenças, as atividades de teste baseado em modelos podem ser definidas em algumas etapas: (i) modelagem dos requisitos da especificação do sistema; (ii) extração dos conjuntos de teste a partir do modelo de especificação; e (iii) o veredito de conformidade, obtido pela comparação do comportamento da implementação em relação à especificação.

Um dos desafios encontrados na utilização dessa abordagem é a sua adoção em sistemas de tempo real, devido a dificuldade tanto para se modelar sistemas com tempo contínuo quanto para lidar com aspectos temporais em aplicações práticas. A seguir são apresentados os principais modelos formais relacionados com este trabalho.

2.1. Formalismos

Os modelos formais são mecanismos adotados para especificar sistemas de maneira bem definida, utilizados para auxiliar as atividades de teste baseado em modelos.

Existe uma grande variedade de modelos, cada uma com suas próprias características que influenciam na sua capacidade de abstração e expressão dos sistemas, bem como nas restrições impostas. Alguns tipos específicos de sistemas, tais como os sistemas de tempo real e/ou reativos, necessitam de um poder de representatividade maior para modelar suas particularidades, como a evolução contínua do tempo e as respostas a estímulos externos do sistema, respectivamente.

Máquinas de Estados Finitos O modelo MEF é utilizado para especificar sistemas que reagem a estímulos do ambiente externo produzindo respostas [Chow 1978]. Neste caso, os estímulos externos e os eventos autônomos gerados pelo sistema, são entradas e saídas associadas e representadas numa transição da máquina. A seguir, uma MEF é definida, formalmente, como uma tupla $M = (X, Y, S, s_0, \delta, \lambda)$, onde:

- X é o conjunto finito e não vazio de símbolos do alfabeto de entrada;
- Y é o conjunto finito e não vazio de símbolos do alfabeto de saída;
- S é o conjunto finito e não vazio de estados;
- s_0 é o estado inicial da máquina;
- δ é a função de mudança de estado, onde $\delta : (X \times S_1) \rightarrow S_2$;
- λ é a função de saída, onde $\lambda : (X \times S) \rightarrow Y$.

Timed Input/Output Automata O modelo TIOA é usado para especificar sistemas temporizados, com a ocorrência de saídas autônomas produzidas pelo sistema [Kaynar et al. 2006]. Basicamente, um TIOA é definido através de um BTDA[1] o

[1] *Bounded Time Domain Automata*

qual inclui uma descrição semântica da evolução contínua de tempo para modelar sistemas de tempo real [Alur and Dill 1994]. Um BTDA é, formalmente, definido pela tupla $M = (S, s_0, \Sigma, C, \nu_0, Inv, T)$, contendo

- um conjunto finito de estados S;
- um estado inicial s_0;
- um conjunto finito de ações Σ;
- um conjunto de relógios C, onde uma variável relógio $c \in C$ descreve a evolução contínua do tempo;
- uma interpretação inicial de relógios ν_0, $\nu(c) = 0$ para todo $\nu(c) \in \nu_0$. Uma interpretação de relógio ν é o conjunto de interpretações $\nu(c)$,$\forall c \in C$ que descreve o valor de cada relógio em um instante de tempo;
- uma função $Inv : S \rightarrow \Phi_C$ que mapeia cada condição de relógio de um estado. Uma condição de relógio é uma expressão booleana obtida através de combinação das formas atômicas $c < m$, $c = m$ e $c > m$, onde c é um relógio e $m \in \mathbb{Q}_{\geq}$. Essas expressões determinam quando uma transição pode ser executada.
- um conjunto de transições de estado $T \subseteq (S \times \Sigma \times \Phi_C \times [C \curvearrowright \mathbb{Q}_{\geq}] \times S)$.

O modelo TIOA nada mais é que uma variação do modelo BTDA, com um particionamento no conjunto de ações em subconjuntos de entrada e de saída. Formalmente, um TIOA é representado pela tupla $M = (B, X, Y)$, onde:

- $B = (S, s_0, \Sigma, C, \nu_0, Inv, T)$ é um BTDA;
- $X \subseteq \Sigma$ e $Y \subseteq \Sigma$ são os conjuntos finitos de ações de entrada e saída, respectivamente, onde $X \cap Y = \emptyset$;

Autômato grid O autômato grid é uma representação discretizada do modelo TIOA. Essa discretização é obtida através da utilização de uma granularidade escolhida. Logo, a obtenção de um grid depende fortemente da técnica de discretização proposta para o modelo temporizado [Bonifácio and Moura 2011]. Um grid pode ser definido pela tupla $M = (S, s, \Sigma, T)$, onde:

- S é um conjunto finito de estados;
- s é o estado inicial;
- $\Sigma \cup g$ é um conjunto finito de rótulos, onde Σ é o alfabeto de entrada e saída do modelo de tempo, e g a granularidade da discretização;
- $T \subseteq (S \times (\Sigma \cup g) \times S)$ é um conjunto finito de transições.

Este trabalho se baseia em métodos recentes de discretização, onde a obtenção de grids com espaço de estado mais compactos diminui o problema da explosão combinatória [Bonifácio and Moura 2011]. Isso permite que a análise de cobertura de falhas proposta para grids seja aplicada em situações práticas.

3. Trabalhos Relacionados ao Teste em Sistemas de Tempo Real

Sistemas de tempo real são caracterizados pela execução de ações onde os instantes de tempo em que ocorrem são importantes. Alguns formalismos específicos são utilizados para modelar a passagem de tempo. Porém, a característica contínua do tempo impossibilita na maioria das vezes sua manipulação em aplicações práticas. Uma das estrategias é a discretização do tempo contínuo dos modelos formais utilizando granularidades adequadas de tal forma que ainda possibilitem a exploração dos requisitos de tempo definidos pelo sistema.

Alur e Dill [Alur 1999, Alur and Dill 1994] propõem uma abordagem de discretização utilizando uma granularidade baseada no número de regiões de relógio do modelo do sistema. Uma região de relógio define um conjunto de valorações dada através das restrições impostas aos relógios do modelo [Alur 1999, En-Nouaary et al. 2002]. Esse número de regiões cresce de acordo com o número de relógios e valores limites que cada relógio atinge. Uma granularidade muito fina pode, então, resultar num crescimento exponencial no número de estados do sistema tornando inviável sua utilização.

En-Nouaary [En-Nouaary et al. 1999] propôs um modelo de falhas para TIOA utilizando autômato de região. Springintveld [Springintveld et al. 2001] e En-Nouaary [En-Nouaary et al. 2002] também utilizam uma granularidade baseada em regiões de relógio para propor variações dos métodos W [Chow 1978] e Wp [Fujiwara et al. 1991] adaptados para sistemas de tempo real modelados em TIOA. A versão original desses métodos proporcionam cobertura total de falhas para modelos de MEF. Em suas aplicações tradicionais para o modelo de MEF, tanto o método W quanto o método Wp proporcionam uma cobertura total das falhas. Porém, em sistemas de tempo real, conforme o número de relógios aumenta, esses métodos se tornam inviáveis na prática.

Em outro trabalho En-Nouaary [En-Nouaary and Hamou-Lhadj 2008] utiliza uma abordagem distinta que explora a estrutura do TIOA. Cada transição é verificada apenas nos limites superior e inferior do intervalo de cada condição de relógio, bem como no ponto médio desses limites. Isso possibilita lidar com o modelo original sem a necessidade de se obter uma representação discretizada do mesmo. Contudo, os três pontos observados em cada transição não oferecem 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 que possam ocorrer nas lacunas de tempo.

Em outra abordagem, Bonifacio e Moura [Bonifácio and Moura 2011] propõem um *framework* de teste que permite uma discretização mais flexível de um TIOA. A escolha da granularidade pode então ser mais grossa, permitindo a obtenção de um espaço de estado mais gerenciável. Porém, a eficácia do método é demonstrada através de propostas de teste, o que restringe a conjunto de falhas detectáveis.

4. Relações de Conformidade

Uma relação de conformidade verifica se uma implementação se comporta de acordo com a sua especificação. A abordagem mais comum é determinar uma relação de conformidade através de propriedades que devem ser satisfeitas quando uma implementação está de acordo com a especificação.

Algumas técnicas de teste definem que uma implementação está conforme sua especificação se elas sempre se comportam da mesma maneira. Quando essa condição não é totalmente verificável, técnicas utilizam hipóteses que limitam o escopo do comportamento a ser verificado. Alguns exemplos de hipóteses que restringem o escopo de teste são: utilizar o mesmo alfabeto de entrada/saída da especificação; limitar o número de estados a ser considerado nas implementações candidatas; e operações de reinicialização do sistema em todo estado da implementação [En-Nouaary et al. 2002].

A seguir são descritas algumas importantes relações de conformidade.

4.1. *Input-Output Conformance*

A relação *input-output conformance* (ioco) é definida originalmente para implementações dadas por *Input Output Transition Systems* (IOTSs) [Tretmans 2008] e especificações modeladas em *Labelled Transition Systems* (LTSs) [Tretmans 2008].

Essa relação se baseia nas sequências de entrada aplicadas ao sistema, também chamadas de traços. O comportamento observável em resposta a um traço é chamado de saída. A relação ioco define que uma implementação I está em conformidade com uma especificação E, se ao aplicar cada traço de E em I a saída obtida é a mesma observada em E [Tretmans 2008].

Considerando que o conjunto de todo par traço/saída descreve o comportamento do sistema, uma das relações ioco pode ser definida como $I\ ioco\ E \equiv comportamento(E) \subseteq comportamento(I)$.

4.2. *Timed Input-Output Conformance*

A relação *timed input-output conformance* (tioco) é uma extensão da relação ioco com o objetivo de capturar sistemas de tempo real modelados com TIOA. Como sistemas de tempo real são ditados pelo tempo contínuo é necessário que os instantes em que as ações ocorrem sejam identificados. *Delays* são utilizados para descrever os intervalos de tempo decorrido. Logo, os traços e saídas possuem *delays* em suas sequências para identificar intervalos entre as ocorrências de ações.

Dados uma implementação I e uma especificação E, a relação de conformidade tioco é definida de maneira similar ao ioco: $I\ tioco\ E \equiv comportamento(E) \subseteq comportamento(I)$. Porém, neste caso os traços são compostos além das entradas e das saídas, também pelos instantes de tempo em que essas ações ocorrem. Algumas propostas de extração de casos de teste para TIOA utilizam *ticks* de tempo [Krichen and Tripakis 2004]. Na prática essas técnicas acabam se comportando como os modelos discretizados descritos anteriormente, podendo ser considerado um caso particular.

5. Detecção de falhas

Avaliar a capacidade de detecção de falhas de um conjunto de testes observando apenas o número de falhas encontradas é uma métrica grosseira, pois diferentes conjuntos de testes podem exercitar diferentes aspectos da especificação. Métricas mais eficientes focam a amplitude da estrutura do sistema explorado durante a execução dos testes e a capacidade de se exercitar partes específicas em busca de falhas.

Duas abordagens se destacam para essa finalidade, uma com o foco na detecção de falhas específicas, chamada proposta de teste [Bonifácio and Moura 2011, Weiglhofer and Wotawa 2009], e outra que foca na estrutura da especificação e classes de falha, chamada modelo de falhas [Chow 1978, En-Nouaary et al. 1999].

A proposta de teste é considerada uma abordagem qualitativa, pois verifica se o conjunto de testes é capaz de detectar propriedades de falha (ou desejáveis) da implementação, não levando em conta uma relação mais precisa sobre o número de falhas detectadas. O formalismo adotado para especificar o sistema também é utilizado para modelar as propriedades desejáveis ou de falha a serem verificadas. Esse modelo então

representa um comportamento que através de sequências de entradas e saídas observáveis levam o controle do modelo aos estados de falha (ou desejáveis) do sistema.

No modelo de falha se observa a estrutura do sistema e as potenciais falhas que implementações candidatas podem apresentar. Através do modelo de falhas é possível identificar classes que o conjunto de testes é capaz de identificar. Dessa forma, uma análise quantitativa sobre a capacidade de detecção pode ser realizada.

Sabe-se que na abordagem de teste baseado em modelos não se conhece qualquer informação a respeito da estrutura da implementação. No entanto, assume-se que a implementação candidata é modelada com o mesmo formalismo utilizado para representar a especificação. Por isso, o modelo de falhas é caracterizado através das possíveis mutações que a especificação pode apresentar em relação ao modelo original.

Nas próximas subseções são descritos os modelos de falha propostos para os modelos MEF e TIOA.

5.1. Modelo de falhas para MEF

Chow [Chow 1978] apresenta um método de geração de casos de teste para modelos MEF e propõe um modelo de falhas que divide as potenciais falhas em três grupos:

Falhas de operação: ocorre quando uma função de operação apresenta uma saída distinta da especificada.

Falhas de transferência: ocorre quando uma função de transferência leva a execução de uma transição a um estado distinto do especificado.

Falhas de estados extras/ausente: ocorre quando a implementação não está em conformidade com a implementação, mas é possível corrigir isso incluindo ou retirando estados.

As falhas de operação são detectadas pelo conjunto de testes ao exercitar todas as transições do modelo de especificação. As falhas de transferência são detectadas pela utilização de um conjunto de sequências de entrada, também chamado de conjunto de identificação, capaz de identificar cada estado do modelo de especificação.

A proposta não prevê falhas de transições extras, pois por hipótese assume-se MEFs completamente especificadas. Página Cada estado de uma MEF completamente especificada possui transições para cada símbolo do alfabeto de entrada. Outra hipótese assumida é que todo estado do modelo é alcançável a partir do estado inicial. Assim falhas de estados ausentes podem ser detectadas exercitando transições que levem ao estado ausente na implementação, verificando se uma falha de transferência não ocorre.

Como estados extras não tem seu comportamento especificado, esses devem ser testado para cada símbolo possível do alfabeto de entrada. Quando estados extras são permitidos, a verificação é estendida por sequências de maior comprimento proporcional ao número de estados extras suposto.

5.2. Modelo de falhas para TIOA

En-Nouaary [En-Nouaary et al. 1999] propõe um modelo de falha para TIOA, classificando as potenciais falhas em: falhas de tempo, falhas de ação e falhas de transferência de estados.

As falhas de tempo são aquelas decorrentes de alterações relacionadas a semântica de tempo, como as restrições e reinicializações de relógio. Contudo, existem modificações que mantêm o comportamento da implementação equivalente ao da especificação, pois representam falhas não efetivas. A seguir são descritas as falhas relacionadas ao tempo.

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 subcasos:

- 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.

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 execução da ação. No caso de ações de saída, que não são ações controláveis, a restrição da condição é considerada válida uma vez que o intervalo restrito está contido no intervalo especificado.

Falha de relaxamento da condição de relógio: Ao relaxar uma condição de relógio, a ocorrência de uma ação é aceita num intervalo maior que o especificado. No caso de uma ação de saída, a ocorrência além do intervalo da condição de relógio não é permitida pela especificação, mesmo as saídas não sendo controláveis. Por isso, esse tipo de falha ocorre tanto em transições com ações de entrada como de saída, diferente das falhas de restrição da condição de relógio.

Falhas de ação e transferência de estados não dependem da semântica de tempo, se comportando similarmente ao modelo de falhas para MEF. A falha de ação ocorre quando uma transição é implementada com uma ação diferente da especificada. Essa falha é similar a falha de operação, contudo no modelo MEF cada transição disparada possui uma entrada e uma saída associada. No TIOA a transição possui uma única ação, seja de entrada ou de saída. Já a falha de transferência ocorre quando uma transição é implementada com o estado destino diferente do especificado.

O modelo de falhas para TIOA foi proposto para analisar a detecção de falhas para conjuntos de teste obtidos com o método *Timed Wp* [En-Nouaary et al. 2002]. Essa análise se baseia na representação de autômatos região. A ocorrência de falhas efetivas descritas no modelo modifica a estrutura do autômato de região [Alur and Dill 1994] que modela a implementação. Essas alterações podem ser no número de estados e transições, ou no estado destino de transições. Como a estrutura de um autômato de região é composta por estados e transições rotuladas, similarmente a estrutura do modelo MEF, essas alterações podem ser mapeadas através do modelo de falhas para MEF.

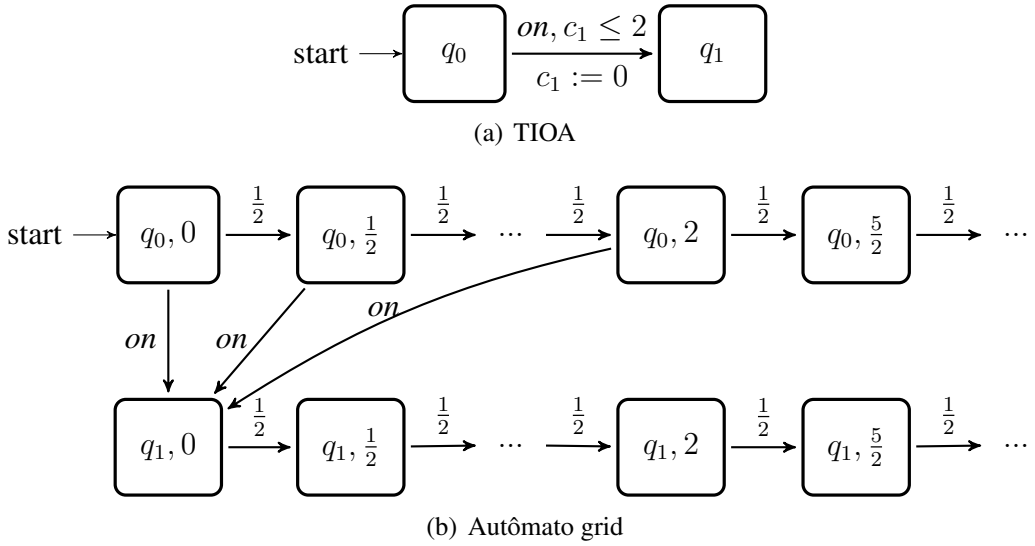
6. Identificação de Falhas em Autômatos Grid

Nesta seção apresentamos a análise realizada sobre a estrutura de autômatos grids e da identificação de suas classes de falha. O objetivo é verificar a capacidade de detecção de falhas de um conjunto de testes para TIOA com base em seus autômatos grid correspondentes.

Com o intuito de ilustrar a identificação dessas classes de falha num autômato grid, assumimos o TIOA especificação e sua correspondente discretização representados

na Figura 1. O TIOA possui uma transição com a ação *on* a partir do estado inicial, que pode ser executada quando o relógio c_1 possui interpretação menor ou igual a 2. Além disso, através dessa transição o relógio c_1 é reinicializado com valor 0.

Nas subseções a seguir, o objetivo é analisar cada classe de falha em TIOA através do autômato grid e propor alternativas sistemáticas para detectá-las.



(a) TIOA

(b) Autômato grid

Figura 1. TIOA e seu grid correspondente

6.1. Falha de ação

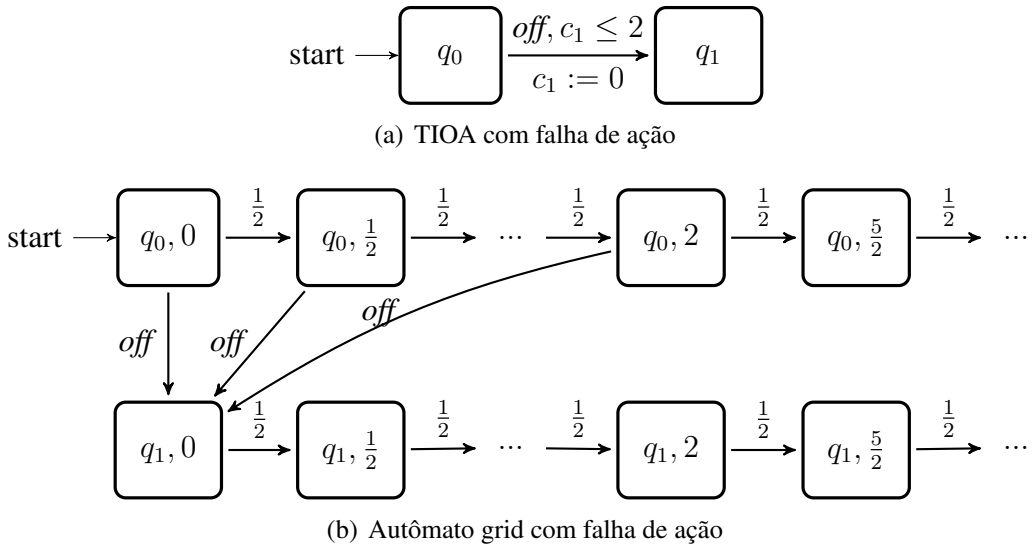
A execução de uma transição num TIOA pode ocorrer em qualquer instante de tempo no intervalo habilitado pela condição de relógio. No autômato grid cada transição do TIOA é simulada por uma transição em cada estado discretizado associado ao estado original do TIOA que habilita sua execução, como mostra a Figura 1.

Uma implementação apresenta falha de ação num TIOA quando uma transição apresenta uma ação distinta da especificação. A implementação candidata da Figura 2(a) para o TIOA da Figura 1(a) apresenta uma falha de ação executando a transição do sistema para uma ação *off* que é diferente da ação *on* no modelo da especificação. Por isso, todas as transições do autômato grid que simulam a transição com falha do TIOA no autômato grid devem ser rotuladas com a ação de falha, como mostra a Figura 1(b) para a ação *off*.

Como essa classe de falhas não depende do tempo, seu comportamento no autômato grid é similar a falha de operação no modelo MEF. Se a especificação for determinística, o estado fonte de uma transição implementada com falha de ação, deixa de responder a ação especificada. Assim um conjunto de testes que exercita cada transição do autômato grid da especificação é capaz de apontar essas falhas pela divergência entre as ações da implementação e da especificação.

6.2. Falha de transferência

Falhas de transferência são caracterizadas por uma transição com estado alvo distinto do especificado. Quando o estado alvo da transição do TIOA muda, o estado alvo de cada transição que a simula no autômato grid também é alterado. Logo, todas as transições



(a) TIOA com falha de ação

(b) Autômato grid com falha de ação

Figura 2. TIOA com falha de ação e seu grid correspondente

do autômato grid que simulam uma transição do TIOA que implementada uma falha de transferência apresentarão a mesma falha.

Essa classe de falha, assim como a falha de ação, não depende da evolução do tempo. A forma de detecção segue de maneira similar àquela utilizada para o modelo MEF. Alguns métodos conhecidos, como o W e o Wp, utilizam conjuntos de identificação [Chow 1978, Fujiwara et al. 1991] que, concatenados a cada caso de teste, são capazes de identificar se o estado alvo da sequência de transições é implementado de maneira correta.

6.3. Falha de condição de relógio

O relaxamento ou restrição das condições de relógio resultam em falhas na implementação à medida que o intervalo de habilitação da transição é alterado. Na simulação em autômato grid essa mudança é refletida na ausência ou ocorrência de transições extras.

Cada estado do TIOA é simulado no autômato grid por um conjunto de estados rotulados pelo par (q, ν), onde ν é uma interpretação de relógio discretizada para o estado original q do TIOA. Quando esse estado é a origem de uma transição do TIOA, cada estado do grid no subconjunto de estados que simula o intervalo onde a condição de relógio está habilitada tem uma transição rotulada simulando a transição do TIOA para o determinado instante de tempo acumulado no intervalo.

Quando ocorre uma falha de restrição da condição de relógio, estados discretizados do autômato grid deixam de habilitar a condição de relógio. Por exemplo, caso uma condição mais restrita, $c_1 < 2$, seja implementada no exemplo da Figura 1(a), o estado do grid com interpretação de relógio $c_1 = 2$ não habilita a condição necessária para executar a transição. Como consequência a transição a partir desse estado que simula esse instante de tempo é ausente, como mostra a Figura 3(a).

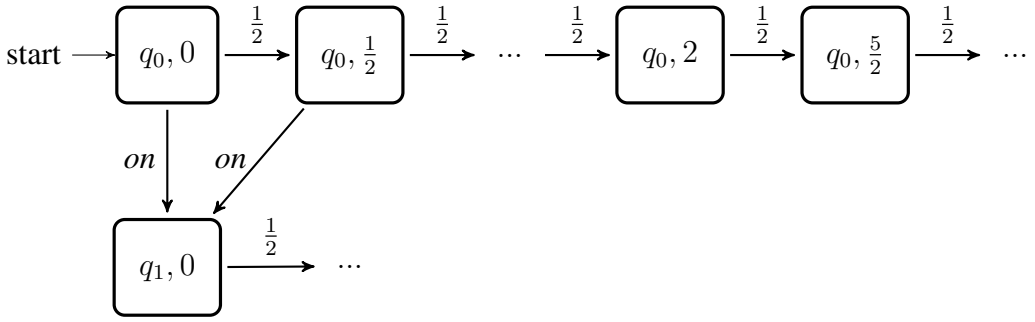
Quando ocorre uma falha de relaxamento da condição de relógio, mais estados passam a habilitar tal condição. Para ilustrar este caso, assuma uma condição mais re-

laxada, $c_1 < 3$", implementada no exemplo da Figura 1(a). O estado com interpretação de relógio $c_1 = \frac{5}{2}$ habilita a condição necessária para executar a transição. Como consequência o estado que simula esse instante de tempo apresenta uma transição extra, como mostra a Figura 3(b).

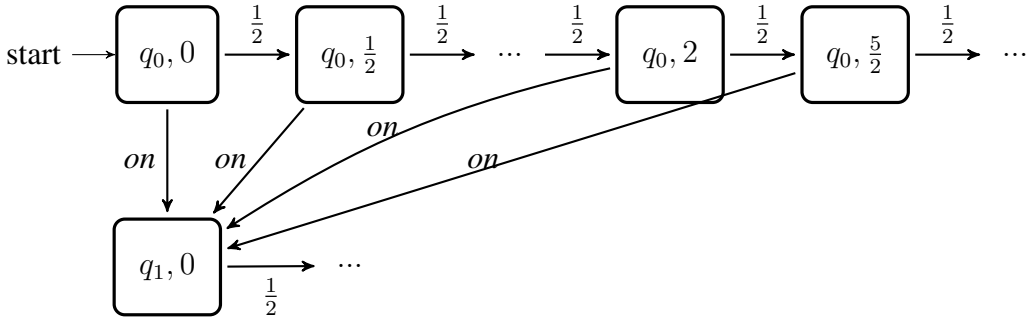
Um conjunto de testes que exercita todas as transições do grid da especificação, detecta a falha de restrição da condição de relógio já que a transição ausente na implementação não pode ser exercitada.

Uma abordagem para detecção de falhas de relaxamento da condição de relógio pode ser obtida através da hipótese em que implementações podem ter transições extras. Essa estrategia é similar à detecção de estados extras para o modelo MEF, pois, da mesma forma, não existe informação sobre o comportamento e a existência dessas transições extras provenientes da especificação.

Outra abordagem para lidar com transições extras, também utilizada no modelo MEF, é assumir que a implementação é determinística e que a especificação é completa. No caso de transições extras, geradas por relaxamento da condição de relógio numa transição, vale lembrar que ações de saída não são controláveis. Por isso, não há garantias de que toda transição de saída será exercitada durante a execução dos testes. Uma implementação com falha de relaxamento da condição de relógio numa transição de saída pode então apresentar um comportamento correto durante a execução dos testes.



(a) Falha de restrição na condição de relógio



(b) Falha de relaxamento na condição de relógio

Figura 3. Grids com falhas na condição de relógio

6.4. Falha de reinicialização de relógio

Falhas de reinicialização de relógio são caracterizadas por apresentar transições de um TIOA com o conjunto de reinicialização de relógios distintos do especificado. A con-

sequência são interpretações de relógio alteradas após a execução da transição para valores diferentes daqueles esperados na especificação.

A Figura 4(a) apresenta uma implementação do TIOA da Figura 1(a) com falha de reinicialização, visto que o relógio c_1 não é reinicializado na transição. Num outro cenário, quando assumimos o TIOA da Figura 4(a) como especificação, a Figura 1(a) representa uma implementação com falha de reinicialização ao implementar uma reinicialização inexistente.

Note que mesmo existindo diferenças na estrutura dos autômatos grid das Figuras 1(b) e 4(b), como observado no estado alvo das transições, estes estados representam o mesmo estado do TIOA original. Porém, essas diferenças com transições para estados alvos distintos nos grids não são suficientes para caracterizar um comportamento de forma a detectar essa classe de falhas.

Uma abordagem para detectar essas falhas é observar como as alterações na estrutura do autômato grid influenciam o comportamento do sistema. Quando uma reinicialização especificada deixa de ser implementada, a transição subsequente que possui uma condição para este relógio apresentará falha de restrição da condição de relógio. Como o valor da interpretação de relógio não é reinicializado na transição, os valores anteriormente especificados não poderão ser mais atingidos. De outra forma, quando uma reinicialização especificada não é implementada, a transição subsequente que possui uma condição para este relógio apresentará uma falha de relaxamento em sua condição de relógio. Como o valor da interpretação de relógio na execução da transição é reinicializado, um limite inferior menor que o esperado para as condições de relógio subsequentes pode ser atingido na evolução do sistema.

Apesar de não ser possível caracterizar as falhas de reinicialização de relógio no autômato grid, é possível que a propagação desse tipo de falha seja detectável através da identificação de falhas nas condições de relógio.

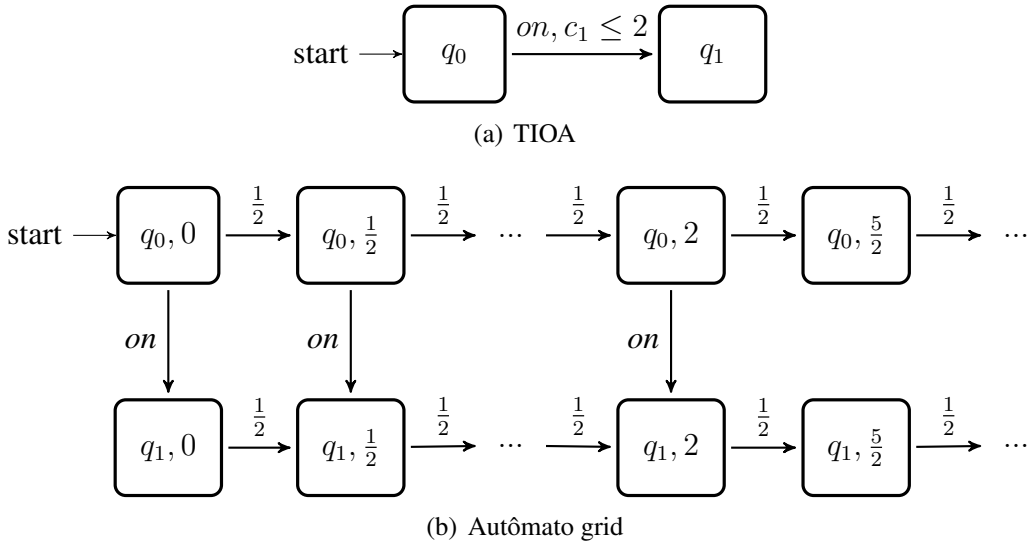
Um terceiro caso particular ocorre quando a transição é implementada com falha de reinicialização de relógio, mas o valor da interpretação do relógio responsável pela falha não é utilizado na condição de relógio de nenhuma transição subsequente. Neste caso, o comportamento do sistema não é alterado, pois independe do relógio ser ou não reinicializado. Essa falha é uma falha considerada não efetiva na execução do sistema.

7. Conclusão

A abordagem de teste baseado em modelos tem sido empregado em sistemas de tempo real. No entanto, sua utilização ainda é um desafio devido a complexidade de se lidar com o aspecto da evolução continua de tempo.

Neste trabalho investigamos a capacidade de se identificar falhas em sistemas de tempo real modelados em TIOA através de autômatos grid. As classes de falha são analisadas de forma que as características de cada tipo de falha possam ser detectadas através da simulação do autômato grid correspondente ao TIOA de especificação. Esta análise foi baseada no modelo de falhas propostos para o modelo TIOA e para o modelo MEF.

Mostramos que as falhas de ação e restrição de relógio podem ser detectadas através de uma cobertura completa das transições do grid.



(a) TIOA

(b) Autômato grid

Figura 4. TIOA sem reinicialização na transição e seu grid correspondente

Já as falhas de transferência podem ser detectadas com a utilização de conjuntos de identificação de estados, pois os modelos grid não possuem a semântica de evolução contínua de tempo. As falhas de relaxamento de condição de relógio apresentam transições extras nos autômatos grid. A detecção dessas transições extras não é possível através da extração de conjuntos de teste a partir de autômatos grid. Uma alternativa é transformar a especificação em modelos completos, de modo que a transição resultante da falha de relaxamento de relógio não seja mais identificada como uma transição extra, mas sim como uma transição com falha de ação.

Observamos também que no caso das falhas de reinicialização de relógio não existe uma forma direta de identificá-las no autômato grid. Contudo, observamos que essa classe de falhas induz a ocorrência de outras falhas no grid. Quando uma operação de reinicialização deixa de ser implementada o autômato grid permite que a falha seja detectada como uma restrição de condição de relógio. Já quando uma operação de reinicialização inexistente é implementada o autômato grid permite que a falha seja detectada como um relaxamento da condição de relógio.

Um direcionamento da pesquisa em andamento é a identificação de outras formas de propagação de falhas que permitam sua detecção assumindo hipóteses e propriedades menos restritivas tanto para a implementação quanto para a especificação. Outro objetivo mais imediato é a formalização das classes de falha apresentadas, de modo a compor precisamente o modelo de falhas para autômatos grid.

Referências

Alur, R. (1999). Timed automata. In *Proceedings of the 11th International Conference on Computer Aided Verification*, CAV '99, pages 8–22, London, UK. Springer-Verlag.

Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235.

Bonifacio, A. L. and Moura, A. V. (2009). A New Timed Discretization Method for Automatic Test Generation for Timed Systems. Technical Report IC-09-31, Institute

of Computing, University of Campinas. http://www.ic.unicamp.br/∼reltech/2009/09-31.pdf.

Bonifácio, A. L. and Moura, A. V. (2011). A new method for testing timed systems. *Software Testing, Verification and Reliability*, pages n/a–n/a. http://onlinelibrary.wiley.com/doi/10.1002/stvr.454/abstract.

Chow, T. S. (1978). Testing software design modeled by finite-state machines. *IEEE Trans. Softw. Eng.*, 4(3):178–187.

En-Nouaary, A., Dssouli, R., and Khendek, F. (2002). Timed wp-method: Testing real-time systems. *IEEE Trans. Softw. Eng.*, 28:1023–1038.

En-Nouaary, A. and Hamou-Lhadj, A. (2008). 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*, pages 209 –215.

En-Nouaary, A., Khendek, F., and Dssouli, R. (1999). Fault coverage in testing real-time systems. In *Real-Time Computing Systems and Applications, 1999. RTCSA '99. Sixth International Conference on*, pages 150 –157.

Fujiwara, S., von Bochmann, G., Khendek, F., Amalou, M., and Ghedamsi, A. (1991). Test selection based on finite state models. *IEEE Trans. Softw. Eng.*, 17(6):591–603.

Hierons, R. M., Bogdanov, K., Bowen, J. P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P., Lüttgen, G., Simons, A. J. H., Vilkomir, S., Woodward, M. R., and Zedan, H. (2009). Using formal specifications to support testing. *ACM Comput. Surv.*, 41(2):1–76.

Kaynar, D. K., Lynch, N. A., Segala, R., and Vaandrager, F. W. (2006). *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan & Claypool Publishers.

Krichen, M. and Tripakis, S. (2004). 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*, pages 109–126. Springer.

M. Blackburn, R. Busser, A. N. (2004). 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.

Springintveld, J., Vaandrager, F., and D'Argenio, P. R. (2001). Testing timed automata. *Theor. Comput. Sci.*, 254:225–257.

Tretmans, J. (2008). Model based testing with labelled transition systems. In Hierons, R. M., Bowen, J. P., and Harman, M., editors, *Formal methods and testing*, pages 1–38. Springer-Verlag, Berlin, Heidelberg.

Utting, M. and Legeard, B. (2007). *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann, 1 edition.

Weiglhofer, M. and Wotawa, F. (2009). Improving coverage based test purposes. In *Quality Software, 2009. QSIC '09. 9th International Conference on*, pages 219 –228.