Redução do Número de Seqüências no Teste de Conformidade de Protocolos

Jorge Francisco Cutigi, Paulo Henrique Ribeiro, Adenilso da Silva Simão, Simone do Rocio Senger de Souza

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
Caixa Postal 668 – 13.560-970 – São Carlos – SP – Brasil

{jcutigi, phr, adenilso, srocio}@icmc.usp.br

Abstract. *Formal specification is a very important step in the development process of protocol, since the specification can be used as basis to the implementation and the conformance testing. Among formal models for protocol specification, Finite State Machines have been often used. This model allows obtaining test sequences for the specified protocol. Several test generation methods have been developed in past decades, aiming at obtaining a test suite that is able to reveal implementation faults. Nevertheless, most of the generated test suites are huge, with a large number of sequences. In this paper, we present an approach to reduce sequences in a test suite. The main goal is the reduction of the number of sequences, since the large number of sequences can turn the test activity impracticable. We present experimental results, which show that the approach reduces the number of sequences while maintaining the effectiveness in revealing faults.*

Resumo. *A especificação formal é uma etapa crucial no ciclo de desenvolvimento de protocolos, uma vez que ela pode ser usada como base para a implementação e para o teste de conformidade. Dentre os modelos formais de especificação de protocolos, as Máquinas de Estados Finitos têm sido muito utilizadas. Esse modelo permite a derivação de seqüências de teste para o protocolo especificado. Vários métodos de geração de seqüências de teste têm sido desenvolvidos há várias décadas, com o objetivo de obter um conjunto de teste que seja capaz de revelar os defeitos de uma implementação. Entretanto, muitas vezes os conjuntos gerados são muito grandes e possuem um grande número de seqüências. Neste artigo é apresentada uma abordagem de redução de seqüências de teste. Busca-se como objetivo principal a redução do número de seqüências do conjunto de teste, uma vez que o grande número de seqüências pode tornar o teste inviável. São apresentados os resultados de dois estudos experimentais, os quais mostram ganhos consideráveis na redução de seqüências nos conjuntos de teste, mantendo a efetividade em revelar defeitos.*

1. Introdução

Um protocolo é um conjunto de regras que regem a troca de mensagens entre entidades em redes de computadores e em sistemas distribuídos complexos. De acordo com [Sidhu et al. 1991], o desenvolvimento de sistemas baseados em protocolos possui quatro etapas: (1) especificação, que consiste na criação do modelo formal do

protocolo; (2) verificação, que consiste em garantir que a especificação esteja correta; (3) implementação, que é a etapa em que a especificação é transformada em software executável; (4) teste de conformidade, que consiste em confrontar o comportamento da implementação com o comportamento do modelo.

Na fase de especificação, várias técnicas podem ser usadas. Dentre as existentes, as Máquinas de Estados Finitos (MEFs) são muito utilizadas devido a sua simplicidade e capacidade de modelar as propriedades gerais de um protocolo [Bochmann and Petrenko 1994]. Além disso, esse tipo de modelo permite a geração automática de conjuntos de teste por meio de vários métodos. Dentre os métodos mais conhecidos, pode-se destacar o método W [Chow 1978], Wp [Fujiwara et al. 1991], HSI [Petrenko et al. 1993, Luo et al. 1994] e H [Dorofeeva et al. 2005].

Grande parte dos métodos existentes geram testes compostos de várias seqüências distintas que devem ser aplicadas no estado inicial do sistema. Em geral, assume-se a existência de um operação *reset*, que leva tanto a MEF quanto sua implementação ao seu estado inicial. O *reset* deve ser inserido no início de cada seqüência do conjunto de teste, portanto, o número de operações *resets* é igual ao número de seqüências de um conjunto de teste. Essa operação pode ainda aumentar o custo do teste [Hierons 2004, Yao et al. 1993, Fujiwara et al. 1991]. Além disso, deve-se assumir que a operação *reset* está implementada de maneira correta [Fujiwara et al. 1991]. Sendo assim, é desejável que um conjunto de teste tenha o mínimo de operações *reset* possível [Hierons 2004, Hierons and Ural 2006].

Como uma solução para o problema do uso de operações *reset*, alguns métodos de geração de conjuntos de teste geram seqüências de verificação [Hennie 1964, Gonenc 1970, Ural et al. 1997, Hierons and Ural 2002, Chen et al. 2005, Ural and Zhang 2006, Hierons and Ural 2006, Simao and Petrenko 2008], que se trata da geração de um conjunto de teste unitário, ou seja, com apenas uma seqüência de teste. Nesses casos, considerados como ideais, o número de seqüências e, conseqüentemente, o número de *resets*, correspondem ao mínimo possível. Porém, esses métodos requerem que a MEF possua uma seqüência de distinção, contudo nem todas as MEFs minimais possuem seqüências de distinção [Lee and Yannakakis 1994].

Os métodos de geração de conjuntos de teste citados têm a propriedade de gerarem conjuntos completos. Um conjunto de seqüências de teste é chamado de *completo* se ele é capaz de detectar todos os defeitos em um determinado domínio. Geralmente, o domínio de defeitos é definido em função do número máximo de estados que uma máquina de estados deve ter para ser equivalente à implementação. Nos conjuntos de teste completos, as seqüências podem ser combinadas para se reduzir o número de seqüências. No entanto, essa abordagem pode reduzir a efetividade do conjunto em revelar defeitos. Dessa forma, a combinação de seqüências, visando a redução do número de seqüências, deve idealmente preservar a completude do conjunto.

Um tópico que possui estreita relação com os conjuntos de teste e que está presente implícita ou explicitamente nos diversos métodos encontrados na literatura

são as condições de suficiência para completude de casos de teste. Um conjunto de condições de suficiência determina quais são as condições que tornam um conjunto de testes completo. Os diversos métodos de geração garantem que o conjunto gerado satisfaz algum conjunto de condições de suficiência e, portanto, possui a propriedade de ser completo. Entretanto, para oferecer essa garantia, normalmente são gerados conjuntos maiores que o necessário. Sendo assim, é possível a utilização de condições de suficiência na geração, minimização e redução de conjuntos de seqüências de teste.

Neste artigo é apresentado um algoritmo para redução do número de seqüências de um conjunto de casos de teste para especificações de protocolos em MEFs, preservando a completude do conjunto. A abordagem se baseia na combinação de seqüências de um conjunto completo. Com essa combinação, o número de seqüências do conjunto é reduzido, além de que a seqüência gerada pela combinação pode gerar redundâncias, o que torna possível uma diminuição no tamanho dessa seqüência. Para garantir a completude do conjunto obtido, verifica-se se as seqüências do novo conjunto satisfazem certas condições de suficiência. Os resultados de uma avaliação experimental são apresentados, avaliando a efetividade da proposta em relação à porcentagem de redução do número de *resets*. Os resultados mostram uma redução de até 80% no número de operações *resets* em relação aos métodos clássicos de geração.

Este artigo está organizado da seguinte forma: na Seção 2 são abordados os principais conceitos relacionados a MEFs, assim como as definições necessárias para o entendimento deste artigo. Na Seção 3 é apresentada a abordagem de redução de *resets*, com os detalhes do algoritmo, análises e um exemplo. Na Seção 4 são apresentados os resultados de uma avaliação experimental da abordagem de redução de *resets*, contendo um estudo de caso em um protocolo de comunicação. Por fim, na Seção 5 são apresentadas as conclusões do artigo, com as limitações da abordagem e trabalhos futuros.

2. Máquina de Estados Finitos

Segundo [Gill 1962], uma MEF é uma máquina hipotética composta por estados e transições, definida a seguir.

Definição 1 *Uma MEF M (determinística e de* Mealy*) é uma tupla $(S, s_1, X, Y, D, \delta, \lambda)$, onde:*

- *S é um conjunto finito de estados, incluindo o estado inicial s_1.*
- *X é um conjunto finito de entradas.*
- *Y é o conjunto finito de saídas.*
- *D é o domínio da especificação, $D \subseteq S \times X$*
- *δ é uma função de transição, $\delta : D \rightarrow S$.*
- *λ é uma função de saída, $\lambda : D \rightarrow Y$.*

Se $D \rightarrow S \times X$ então a MEF é completamente especificada ou completa. Caso contrário, ela é chamada de parcialmente especificada ou parcial. Uma tupla $(s, x) \in D$ é uma transição definida de M. Um seqüência $\alpha = x_1 x_2 ... x_k$ é dito ser uma seqüência de entrada definida no estado $s \in S$ se existe $s_1, s_2, ..., s_{k+1}$, onde $s_1 = s$, tal que $(s_i, x_i) \in D$ e $\delta(s_i, x_i) = s_{i+1}$ para todo $1 < i < k$. Denota-se por $\Omega(s)$ o conjunto de todas as seqüências de entradas definidas no estado s.

Uma MEF pode ser representada por um grafo direcionado, no qual cada estado é representado por um vértice e cada transição é representada por uma aresta direcionada. O estado inicial é indicado por uma seta incidente ao nó da MEF. Cada aresta possui um rótulo que indica o par *entrada/saída* e o próximo estado. Um exemplo dessa representação é apresentado na Figura 1. O conjunto S de estados representa o conjunto de todas as configurações possíveis do sistema em relação aos símbolos de entrada e saída. A MEF da Figura 1 possui o conjunto de estados $S = \{s_1, s_2, s_3, s_4\}$, o alfabeto de entrada $X = \{a, b\}$ e o alfabeto de saída $Y = \{0, 1\}$.

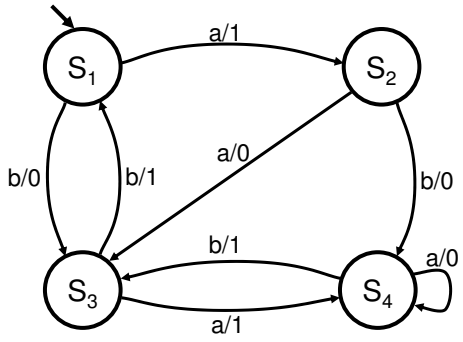


Figura 1. MEF

Uma MEF é minimal se nenhum par de estados da máquina é equivalente, ou seja, para todo par de estados (s_i, s_j), existe ao menos uma seqüência de entradas, definidas em s_i e s_j, cujas saídas produzidas sejam distintas. Uma MEF é fortemente conexa se para todo par de estados (s_i, s_j), existe uma seqüência de entradas, definidas em s_i, que leve de s_i até s_j.

A função de transição δ e a função de saída λ são estendidas para seqüência de entradas, incluindo a seqüência vazia, que é denotada por ϵ. Tem-se que $\delta(s, \epsilon) = s$ e $\lambda(s, \epsilon) = \epsilon$ para todo $s \in S$. Seja β uma seqüência de entradas e $\delta(s, \beta) = s'$, então, para todo $x \in X$ define-se $\delta(s, \beta x) = \delta(s', x)$ e $\lambda(s, \beta x) = \lambda(s, \beta)\lambda(s', x)$. Uma seqüência de transferência χ de s_i para s_j, é uma seqüência que conduz M do estado s_i para o estado s_j, ou seja, $\delta(s_i, \chi) = s_j$.

Dois estados $s_i, s_j \in S$ são equivalentes se existe $\gamma \in \Omega(s_i) \cap \Omega(s_j)$, tal que $\lambda(s_i, \gamma) = \lambda(s_j, \gamma)$. Esse conceito pode ser aplicado em estados de MEFs diferentes. MEFs são equivalentes se seus estados iniciais são equivalentes. De forma análoga, dois estados, $s_i, s_j \in S$ são distinguíveis se existe uma seqüência $\gamma \in \Omega(s_i) \cap \Omega(s_j)$ tal que $\lambda(s_i, \gamma) \neq \lambda(s_j, \gamma)$. MEFs são distinguíveis se seus estados iniciais são distinguíveis.

A operação *reset* é uma operação especial que leva a MEF de qualquer estado para o estado inicial e com saída nula. É denotada pela letra r e aparece como primeiro símbolo em uma seqüência de teste. O tamanho de um conjunto de seqüências de teste é obtido pelo número de símbolos de entrada contido no conjunto adicionado com o número de operações *resets*.

Na geração de testes a partir de MEFs, assume-se que a implementação pode ser modelada como uma MEF contida em um domínio de defeitos. Essa hipótese, conhecida como hipótese de teste, é necessária para que um conjunto fi-

nito de testes possa ser gerado [Chow 1978, Ural et al. 1997, Hierons and Ural 2006, Hennie 1964]. $\Im(M)$ denota o domínio de defeitos definido pelo conjunto de todas as MEFs com o mesmo alfabeto de entrada e no máximo o mesmo número de estados de M, utilizado por grande parte dos métodos de geração, como por exemplo, os métodos W [Chow 1978], Wp [Fujiwara et al. 1991], HSI [Petrenko et al. 1993, Luo et al. 1994], H [Dorofeeva et al. 2005], entre outros.

Um caso de teste T é *completo* se para cada MEF $N \in \Im(M)$ tal que N e M são distinguíveis, existe uma seqüência pertencente a T que distingue N de M. Ou seja, se o caso de teste é completo, ele é capaz de revelar todos os defeitos de uma implementação de M que possa ser modelada por uma MEF de $\Im(M)$.

Condições de suficiência determinam quais são as propriedades que tornam um caso de teste capaz de revelar todos os defeitos de um dado domínio. Ou seja, se determinado caso de teste satisfaz as condições de suficiência, garante-se que esse caso de teste é completo. Condições de suficiência para conjuntos de seqüências de teste foram definidas em [Petrenko et al. 1996], as quais permitiram provar a completude de diversos métodos de geração já existentes. Em [Dorofeeva et al. 2005] foram definidas novas condições que generalizavam as condições de [Petrenko et al. 1996]. Posteriormente, em [Simao and Petrenko 2009] foram definidas condições de suficiência mais abrangentes, as quais generalizam todas as anteriores.

Um algoritmo para a verificação da completude de casos de teste também é apresentado em [Simao and Petrenko 2009], o qual foi utilizado para verificar a completude dos conjuntos no presente trabalho.

3. Redução de *resets*

Os conjuntos de teste gerados pelos métodos clássicos geralmente apresentam redundâncias, ou seja, há seqüências ou subseqüências que poderiam ser descartadas ou substituídas por outra de tamanho menor. Com isso, obtém-se um conjunto menor, mantendo a propriedade de completude do conjunto. Uma estratégia que também pode diminuir o conjunto é a combinação de seqüências. Essa estratégia consiste na concatenação das seqüências de teste de forma a manter a completude do conjunto. A redução do número de seqüências é relevante no sentido de que o número de operações *reset* também diminui.

Neste artigo, é proposta uma estratégia de redução do número de seqüências de conjuntos de seqüências de teste. A estratégia consiste em dois passos: (1) Concatenar as seqüências de um conjunto de seqüências de teste, de forma a obter um conjunto ainda completo e com menos seqüências que o conjunto original; (2) A partir do conjunto gerado no passo 1, identificar em cada seqüência do conjunto concatenado o menor prefixo dela que ainda mantém a completude do conjunto, para que assim sejam eliminados símbolos de entrada desnecessários.

O processo de concatenação de seqüências utilizado no passo 1 do algoritmo é realizado de duas formas:

- **Sobreposição:** Essa forma de concatenação consiste na sobreposição de entradas em duas seqüências de entrada. A concatenação por sobreposição

se dá em uma seqüência $\alpha = \chi\phi$ concatenada com uma seqüência $\beta = \phi\gamma$, gerando uma seqüência $\omega = \alpha\gamma$, onde $\delta(s_1, \chi) = s_1$. Neste artigo, a operação de concatenação por sobreposição é denotada por $concatSP(\alpha, \beta)$.

- **Seqüência de transferência:** Ao fim da seqüência α deve ser adicionada uma seqüência de transferência χ que leva a MEF do estado $s = \delta(s_1, \alpha)$ ao estado inicial s_1. Antes de concatenar a seqüência β escolhe-se a menor seqüência de transferência. Dessa forma, a concatenação final resulta em $\omega = \alpha\chi\beta$. Neste artigo, a operação de concatenação por seqüência de transferência é denotada por $concatST(\alpha, \beta)$.

A concatenação por sobreposição é preferível em relação à que utiliza seqüências de transferência, uma vez que, com a sobreposição, entradas são reaproveitadas, o que leva a uma diminuição no tamanho da seqüência ω. O contrário acontece com o uso de seqüência de transferência, em que o tamanho da seqüência ω obtida é maior que a soma dos tamanhos de α e β, o que aumenta o tamanho da seqüência. Nota-se que essas duas formas de concatenação mantém em ω as transições originais de α e β. A Figura 2 ilustra as duas estratégias de concatenação, em que a seqüência $\alpha = aabab$ é concatenada por sobreposição com a seqüência $\beta = abaa$, que resulta na seqüência $aababaa$. A seqüência $\beta = abaa$ também é concatenada com a seqüência $\gamma = aaab$ por meio da seqüência de transferência $\chi = bb$, resultando na seqüência $abaabbaaab$.

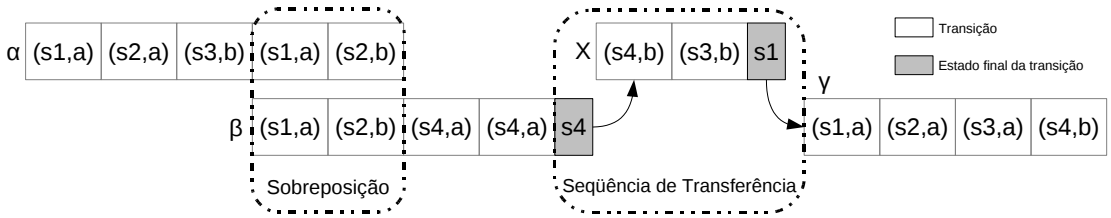


Figura 2. Sobreposição e Seqüência de Transferência

O algoritmo de redução recebe como entrada uma MEF determinística, minimal, de *Mealy* e fortemente conexa, e um conjunto de seqüências de teste. O algoritmo é dividido em dois passos, como seguem.

O **passo 1** do algoritmo tem como objetivo a obtenção de um conjunto com menos seqüências e que ainda seja completo. Tem-se um conjunto completo TS gerado por algum método de geração. Com isso, seleciona-se duas seqüências α e β, tal que $\{\alpha, \beta\} \subseteq TS$, que serão concatenadas de acordo com o processo de concatenação citado, gerando uma seqüência ω definida no estado inicial. Feito isso, é verificado se o conjunto $TS - \{\alpha\} - \{\beta\} \cup \{\omega\}$ satisfaz as condições de suficiência, isto é, que ele ainda é completo. Caso positivo, remove-se α e β de TS, insere ω e repete-se o processo com TS. Caso contrário, repete-se o processo com outra escolha de seqüências α e β. Ao fim, tem-se o conjunto TS completo e com um número menor de seqüências. Em resumo, o passo 1 recebe o conjunto TS e o concatena o máximo de seqüências possíveis utilizando a estratégia de sobreposição. Com o conjunto gerado, a forma de concatenação por seqüência de transferência pode ser utilizada. O algoritmo 1 apresenta o passo 1.

Data: MEF M e Conjunto TS de seqüências de teste
Result: Conjunto TS concatenado
while *existe* $\alpha, \beta \in TS$, *tal que* $TS - \{\alpha, \beta\} \cup \{concatSP(\alpha, \beta)\}$ *é completo* **do**
    $TS \leftarrow TS - \{\alpha, \beta\} \cup \{concatSP(\alpha, \beta)\}$;
end
while *existe* $\alpha, \beta \in TS$, *tal que* $TS - \{\alpha, \beta\} \cup \{concatST(\alpha, \beta)\}$ *é completo* **do**
    $TS \leftarrow TS - \{\alpha, \beta\} \cup \{concatST(\alpha, \beta)\}$;
end
return TS

Algoritmo 1: Algoritmo do passo 1

Exemplificando o processo de concatenação descrito, considere a MEF da Figura 1 e o conjunto completo gerado pelo método W

$$TS = \{raabb, rbabb, raaabb, rababb, rbaabb, rbbabb, rabaabb, rabbabb\}$$

de tamanho 48 e 8 operações *resets*.

Toma-se $\alpha = aabb$ e $\beta = babb$, ambas sem as operações *resets*. Nesse caso, a concatenação por sobreposição é possível, pois a seqüência α pode ser decomposta em $\chi = aab$ e $\phi = b$, com $\delta(s_1, \chi) = s_1$, e a seqüência β pode ser decomposta em $\phi = b$ e $\gamma = abb$. Com isso, tem-se $\omega = concatSP(\alpha, \beta) = \alpha\gamma = aabbabb$. Obtém-se o conjunto $TS = \{raabbabb, raaabb, rababb, rbaabb, rbbabb, rabaabb, rabbabb\}$. Verifica-se que esse conjunto satisfaz as condições de suficiência propostas em [Simao and Petrenko 2009]. A partir desse conjunto resultante, repete-se o mesmo procedimento. Neste exemplo, apenas mais uma sobreposição é possível, com $\alpha = bbabb$ e $\beta = abbabb$), que resulta em $\omega = bbabbabb$. Com isso, tem-se o conjunto completo $TS = \{raabbabb, rbbabbabb, raaabb, rababb, rbaabb, rabaabb\}$.

Após não haver mais possibilidade do uso de sobreposição na concatenação, faz-se então o uso de seqüências de transferência. Exemplificando, toma-se $\alpha = aabbabb$ e $\beta = aaabb$. Nesse caso, uma seqüência de transferência χ deve ser inserida entre α e β. Tem-se então $\delta(s_1, aabbabb) = s_1$, o que indica que χ deve ser uma seqüência que leva a MEF do estado s_1 ao próprio estado inicial s_1, resultando na seqüência vazia $\chi = \epsilon$. Com isso, tem-se a seqüência $\omega = \alpha\chi\beta$, que resulta na seqüência $\omega = aabbabbaaabb$. Com isso tem-se o conjunto $TS = \{raabbabbaaabb, rbbabbabb, rababb, rbaabb, rabaabb\}$. Verifica-se que o conjunto satisfaz as condições de suficiência propostas em [Simao and Petrenko 2009]. Ao fim do processo, tem-se o conjunto completo resultante $TS = \{raabbabbaaabbbbabbabbababbbaabbabaabb\}$. Nota-se que o conjunto agora contém apenas uma seqüência, o que indica a redução máxima de *resets* que pode ser obtida.

Após a criação do conjunto TS concatenado, tem-se o **passo 2** do algoritmo, o qual consiste na redução desse conjunto TS. Nessa etapa, para cada seqüência α em TS, deve-se identificar o menor prefixo β de α necessário para ainda manter a completude do conjunto. Com isso, remove-se símbolos de entradas desnecessários. O passo 2 é apresentado no algoritmo 2.

```
Data: MEF M e Conjunto TS de seqüências de teste
Result: Conjunto TS_reduzido
1 for cada seqüência α ∈ TS do
2     seja β o menor prefixo de α tal que TS − {α} ∪ {β} é completo;
3     TS ← TS − {α} ∪ {β};
4 end
5 return TS;
```

Algoritmo 2: Algoritmo do passo 2

Por exemplo, tomando-se o conjunto $TS = \{raabbabbaaabbbbabbabbababbbaabbabaabb\}$, tem-se apenas uma seqüência, a qual apenas o prefixo $\beta = aabbabbaaabbbbabbabbababbbaabb$ é necessário para manter a completude do conjunto.

Ao fim da execução dos passos 1 e 2, o conjunto final e reduzido da abordagem é

$$TS \text{ reduzido} = \{raabbabbaaabbbbabbabbababbbaabb\}$$

de tamanho 31 e com 1 operação *reset*, o que indica uma redução de 87,5% no número de operações *resets* e uma redução de 35,4% em relação ao tamanho do conjunto.

Diferentes resultados podem ser obtidos com a execução de uma mesma MEF e um mesmo conjunto de seqüências, dependendo da ordem em que as seqüências são consideradas. Para evitar que o algoritmo tenha o desempenho influenciado pela ordem das seqüências, elas são selecionadas de forma aleatória. Executando-se 10 vezes o algoritmo com o exemplo apresentado, a média de redução do tamanho do conjunto foi de de 47,5% e a média de redução de *resets* foi de 77,6%, demonstrando que o resultado do exemplo apresentado é significativo. Deve-se ressaltar que o conjunto de testes obtidos possui o mesmo poder de revelar defeitos modelados pelo domínio de defeitos.

4. Avaliação Experimental

A fim de avaliar a abordagem proposta, estudos experimentais foram conduzidos de forma a verificar a redução do número de operações *resets* no conjunto final. Para isso, dois estudos foram realizados. O primeiro trata de experimentos conduzidos com MEFs aleatórias. O segundo estudo aplica a redução de *resets* em um protocolo de comunicação.

No primeiro estudo experimental foram geradas de maneira aleatória MEFs completas e minimais com 5 entradas, 5 saídas e com o número n de estados variando de 4 a 15, sendo que para cada valor de n foram geradas 30 MEFs diferentes, totalizando 360 MEFs. O processo de geração das MEFs, descrito em [Simao and Petrenko 2009] é realizado em três etapas: Na primeira etapa, um estado é selecionado com estado inicial e marcado como alcancável. Então, para cada estado estado s não marcado como alcançável, o gerador aleatório seleciona um estado alcançável s', uma entrada x e uma saída y e adiciona uma transição de s' para s com entrada x e saída y, marcando s como alcançável. Feito isso, tem-se início a segunda etapa, em que o gerador adiciona, se necessário, mais transições aleatórias.

Na terceira etapa, é verificado se a MEF é minimal. Caso não seja minimal, a MEF é descartada e uma outra MEF é gerada.

Os conjuntos gerados para realizar a redução foram obtidos a partir dos métodos W [Chow 1978], HSI [Petrenko et al. 1993, Luo et al. 1994] e H [Dorofeeva et al. 2005]. Portanto, foram gerados três conjuntos de seqüências para cada uma das 360 MEFs, totalizando 1080 conjuntos de seqüências de teste.

Considerando os conjuntos obtidos pelo método W, os resultados obtidos após a condução dos experimentos evidenciaram uma redução média de *resets* em 89,4% com desvio padrão de 8,1%. Em relação aos conjuntos gerados pelo método HSI, os *resets* foram reduzidos em 82,3%, com desvio padrão de 14,8%. Já em relação aos conjuntos gerados pelo método H, os resultados mostraram uma redução média de 71,2% com desvio padrão de 16,3%. Considerando os três métodos, a média de redução de *resets* foi de 80,9%. Esses dados são apresentados na Tabela 1.

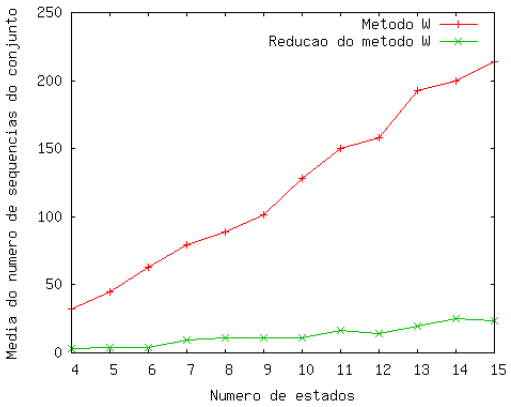
Tabela 1. Resultados gerais para o Estudo 1

Método	Redução de *resets* obtida	Desvio Padrão
W	89,4%	8,1%
HSI	82,3%	14,8%
H	71,2%	16,3%
Média	80,9%	

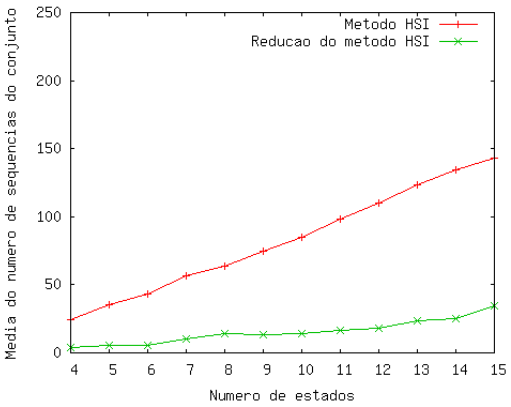
Os gráficos da Figura 3.a, 3.b e 3.c mostram o crescimento médio do número de *resets* pelos métodos W, HSI e H, respectivamente. Nota-se que o crescimento de *resets* obtido na redução é bem menor que o crescimento obtido na geração. Os gráficos mostram também que as taxas de crescimento de *resets* da redução dos métodos H e HSI são muito parecidos, apesar do método HSI gerar conjuntos maiores que o método H. Outro fato interesante inferido por meio dos gráficos é que os números de seqüências do conjunto resultante da redução são bem parecidos, independente do método utilizado.

A Figura 3.d apresenta um gráfico boxplot [McGill et al. 1978], que representa a distribuição dos dados das taxas de redução para cada método. Nos dados do método H, a taxa de redução se concentra na maior parte dos casos entre 30% e 98%, com maior densidade entre 60% e 85% e um *outlier* inferior que aponta um caso em que a redução não foi possível. Para o método HSI, os dados mostram uma taxa de redução entre 50% e 98%, com maior densidade entre 70% e 90%. Porém, a redução do método HSI apresenta alguns *outliers* inferiores, que indicam que em alguns casos a redução foge da área de maior densidade apresentada no gráfico. O comportamento em relação ao método W é parecido com o HSI, em que *outliers* também aparecem. Porém, o intervalo de maior densidade dos dados é menor que os método H e HSI, ou seja, as taxas de redução são muito próximas, independente da MEF e do conjunto W reduzido.

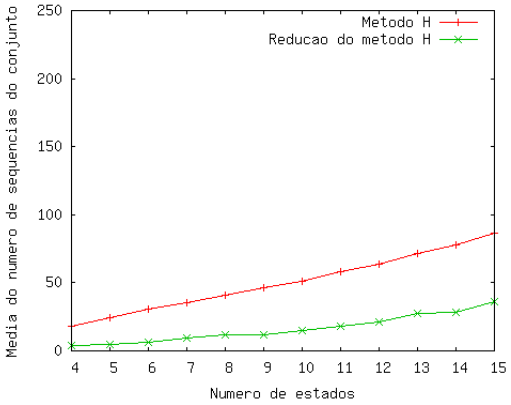
O segundo estudo experimental foi baseado na aplicação da abordagem de redução envolvendo uma especificação em MEF de um protocolo. O estudo de caso é construído sobre o protocolo INRES (INitiator-RESponder) [Tan et al. 1996,



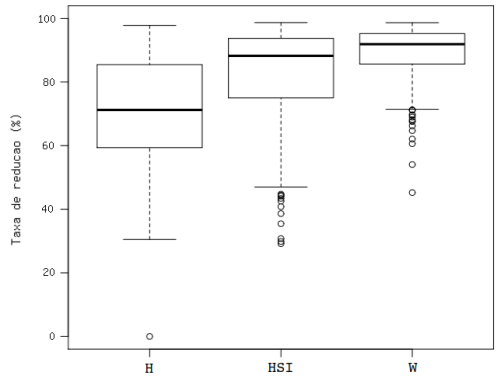
(a) Variação do número de seqüências para o método W



(b) Variação do número de seqüências para o método HSI



(c) Variação do número de seqüências para o método H



(d) Boxplot das taxas de redução de cada método

Figura 3. Gráficos do primeiro estudo experimental

Hogrefe 1991], que contém aspectos essencias dos protocolos de comunicação e mantém a regra de comunicação entre as duas entidades do protocolo: *Initiator* e *Responder*. Na Figura 4 é apresentada a MEF que especifica o comportamento do *Responder* do protocolo INRES. A MEF é determinística, reduzida e parcialmente especificada, a qual contém 4 estados, 5 entradas, 7 saídas e 11 transições.

Por se tratar de uma MEF parcial, os métodos HSI e H foram utilizados para a geração do conjunto de seqüências de teste, os quais foram submetidos ao processo de redução das operações *resets*. Cada conjunto de seqüências de teste foi submetido 10 vezes no processo de redução, de modo a obter um resultado sem influências da escolha aleatória de seqüências. O conjunto gerado pelo método HSI contém 21 operações *resets* e, quando aplicada a redução, esse número é reduzido em 65%. O conjunto gerado pelo método H contém possui 17 *resets*, o qual foi reduzido a uma taxa de 62%. Deve-se observar que, por se tratar de uma MEF parcial, o número de seqüências geradas é geralmente menor e, conseqüentemente, as possibilidades de redução são menores. Contudo, obteve-se uma redução significativa.

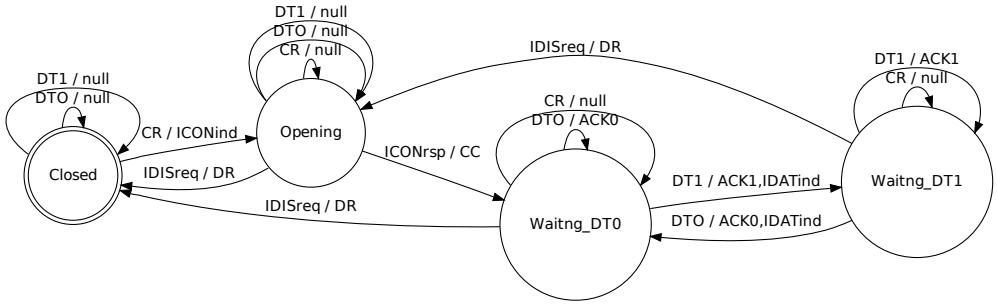


Figura 4. Protocolo INRES – Responder

5. Conclusões

Neste artigo foi investigado o problema da redução de conjuntos de seqüências de teste a partir de MEFs. Em particular, o problema da redução do número de operações *resets* foi priorizado, uma vez que essa operação, em grande parte das vezes, é muito custosa para a aplicação. Para isso, um algoritmo de redução baseado nas condições de suficiência [Simao and Petrenko 2009] foi proposto. A abordagem mostrou ganhos significativos, os quais foram comprovados por meio da condução de uma avaliação experimental com conjuntos gerados pelos métods W, HSI e H. Ganhos médios de 80% na redução de operações *resets* foram verificados. Com isso, mostrou-se também a viabilidade das condições de suficiência definidas em [Simao and Petrenko 2009], que são capazes de reconhecer conjunto menores que os gerados pelos métodos citados.

Como perspectivas para trabalhos futuros, o problema da concatenação aleatória de seqüências é um ponto em que pode haver evoluções. A identificação de propriedades nas seqüências podem direcionar uma melhor escolha delas, otimizando o processo final de redução, além de escolher duas seqüências que obrigatoriamente gerariam ainda um conjunto completo.

Referências

Bochmann, G. V. and Petrenko, A. (1994). Protocol testing: review of methods and relevance for software testing. In *ISSTA '94: Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis*, pages 109–124, New York, NY, USA. ACM.

Chen, J., Hierons, R. M., Ural, H., and Yenigun, H. (2005). Eliminating redundant tests in a checking sequence. In *TestCom 2005*, number 3502 in lncs, pages 146–158.

Chow, T. S. (1978). Testing software design modeled by finite-state-machines. *IEEE Transactions on Software Engineering*, 4(3):178–186.

Dorofeeva, R., El-Fakih, K., and Yevtushenko, N. (2005). An improved conformance testing method. In *FORTE*, pages 204–218.

Fujiwara, S., Bochman, G. V., Khendek, F., Amalou, M., and Ghedamsi, A. (1991). Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17(6):591–603.

Gill, A. (1962). *Introduction to the Theory of Finite-State Machines*. McGraw-Hill, New York.

Gonenc, G. (1970). A method for design of fault detection experiments. *IEEE Transactions on Computers*, 19(6):551–558.

Hennie, F. C. (1964). Fault detecting experiments for sequential circuits. pages 95–110.

Hierons, R. M. (2004). Using a minimal number of resets when testing from a finite state machine. *Inf. Process. Lett.*, 90(6):287–292.

Hierons, R. M. and Ural, H. (2002). Reduced length checking sequences. *IEEE Transactions on Computers*, 51(9):1111–1117.

Hierons, R. M. and Ural, H. (2006). Optimizing the length of checking sequences. *IEEE Transactions on Computers*, 55(5):618–629.

Hogrefe, D. (1991). Osi formal speci cation case study: the inres protocol and service. Technical report, University of Bern.

Lee, D. and Yannakakis, M. (1994). Testing finite-state machines: State identification and verification. *IEEE Trans. Comput.*, 43(3):306–320.

Luo, G., Petrenko, R., and Bochmann, G. V. (1994). Selecting test sequences for partially-specified nondeterministic finite state machines. In *In IFIP 7th International Workshop on Protocol Test Systems*, pages 91–106.

McGill, R., Tukey, J. W., and Larsen, W. A. (1978). Variations of box plots. *The American Statistician*, 32(1):12–16.

Petrenko, A., von Bochmann, G., and Yao, M. Y. (1996). On fault coverage of tests for finite state specifications. *Computer Networks and ISDN Systems*, 29(1):81–106.

Petrenko, A., Yevtushenko, N., Lebedev, A., and Das, A. (1993). Nondeterministic state machines in protocol conformance testing. In *Protocol Test Systems*, pages 363–378.

Sidhu, D., Chung, A., and Blumer, T. P. (1991). Experience with formal methods in protocol development. *SIGCOMM Comput. Commun. Rev.*, 21(2):81–101.

Simao, A. S. and Petrenko, A. (2008). Generating checking sequences for partial reduced finite state machines. In *TestCom '08 / FATES '08: Proceedings of the 20th IFIP TC 6/WG 6.1 international conference on Testing of Software and Communicating Systems*, pages 153–168, Berlin, Heidelberg. Springer-Verlag.

Simao, A. S. and Petrenko, A. (2009). Checking fsm test completeness based on sufficient conditions. *IEEE Transactions on Computers*. (Aceito para publicacao. Versão preliminar disponível em: www.crim.ca/Publications/2007/documents/plein_texte/ASD_PetA_0710_20.pdf).

Tan, Q. M., Petrenko, A., and Bochmann, G. V. (1996). A test generation tool for specifications in the form of state machines. In *in Proceedings of the International Communications Conference*, pages 225–229.

Ural, H., Wu, X., and Zhang, F. (1997). On minimizing the lengths of checking sequences. *IEEE Transactions on Computers*, 46(1):93–99.

Ural, H. and Zhang, F. (2006). Reducing the lengths of checking sequences by overlapping. *Lecture Notes on Computer Science*, (3964):274–288.

Yao, M., Petrenko, A., and Bochmann, G. v. (1993). Conformance testing of protocol machines without reset. In *Proceedings of the IFIP TC6/WG6.1 Thirteenth International Symposium on Protocol Specification, Testing and Verification XIII*, pages 241–256, Amsterdam, The Netherlands, The Netherlands. North-Holland Publishing Co.