ANDERSON PEREIRA DAS NEVES

MÉTODO PARA VERIFICAÇÃO DE PROPRIEDADES DE REDES DE PETRI UTILIZANDO RESOLVEDORES SMT.

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Fabiano Silva

CURITIBA

2013

SUMÁRIO

LISTA DE ABREVIATURAS

BCP *Boolean Constraint Propagation*.

CNF *Conjunctive normal form*.

DPLL *Davis-Putnam-Logemann-Loveland*.

SAT Satisfatibilidade Booleana.

SMT Satisfiability Modulo Theories.

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho apresenta um método para verificar as propriedades de alcançabilidade e de bloqueio em rede de Petri. Rede de Petri é um modelo formal utilizado para modelar sistemas. O método proposto consiste em representar uma propriedade da rede de Petri em uma instância SMT, podendo então submetê-la à um resolvedor SMT, a fim de que o resultado desta resolução defina se a rede tem a propriedade a ser verificada.

O método proposto foi implementado e avaliado em um conjunto de redes de Petri. A modelagem apresenta como resultado as transições e o número de vezes que as mesmas são disparadas, as marcações intermediarias e todos os estados da rede para cada disparo necessário para atingir um estado desejado. Durante os testes obteve-se tempos muito próximos ao do sistema Lola que é uma ferramenta especifica para anÃ¡lise de redes de Petri.

ABSTRACT

This work presents a method for verifying the properties of reachability and deadlock in Petri Net. Petri Net is a formal method used in systems modeling. The proposed method consists in represent properties of the Petri net in a SMT instance, then submit it to a resolver SMT, so that the result of this resolution set if the network has the property to be verified.

The method proposed has been implemented and evaluated in a set of Petri nets. The results of this modeling presents the transitions and the number of times that they were shot, the markings and all intermediate states of the network for each shot needed to achieve a desired state. During tests times were obtained very close to the Lola system that is a specific tool for the analysis of Petri nets.

CAPÍTULO 1

INTRODUÇÃO

Muitos sistemas são modelados como sistemas formais para validar seu funcionamento, uma vez que implementar um sistema com falhas pode gerar grandes perdas de recursos. Para verificação do comportamento dos sistemas modelados e existência de erros nestes sistemas são analisadas algumas características presentes no modelo formal do sistema. Contudo nem sempre estas característica são fáceis de ser analisadas e demandam muito tempo para análise. Um modelo formal muito utilizado para modelagem de sistemas são redes de Petri.

Rede de Petri é um modelo formal utilizado para modelar sistemas, proposto por Carl Adam Petri's em sua dissertação "Communication with automata"[32]. As redes de Petri podem ser utilizadas para modelar vários problemas, possibilitando a análise do comportamento de um sistema modelado. No entanto a análise de certas propriedades tais como alcançabilidade e bloqueio são complexas e demandam uma grande quantidade de processamento.

Existem vários trabalhos que abordam a análise de propriedades de rede de Petri e apresentam técnicas para resolvê-las como apresentado em [38] e [33], onde as propriedades das redes de Petri são transformadas em problemas de planejamento, e então um planejador é utilizado pra verificar se a rede tem a propriedade verificada.

Da mesma forma que nos trabalhos [38] e [33] este trabalho apresenta uma método para verificar as propriedades de alcançabilidade e bloqueio em redes de Petri, porém utilizando um resolvedor SMT (*Satisfiability Modulo Theories*) .

A propriedade de alcançabilidade consiste em verificar se, a partir de uma rede e uma marcação inicial é possível alcançar determinada marcação. Com esta verificação é possível descobrir em um sistema modelado a possibilidade de chegar a um estado desejado.

A propriedade de bloqueio consiste em verificar se, a partir de uma marcação inicial de uma rede, pode-se chegar a uma marcação em que não seja mais possível realizar qualquer transição para outra marcação. Se uma rede contém bloqueio possivelmente seja pela ocorrência de erro no sistema modelado ou na atividade de modelagem.

Satisfatibilidade módulo teoria (SMT) é um formalismo que generaliza a lógica proposicional adicionando razão de igualdade, aritmética, *arrays*, entre outras teorias [15]. SMT estuda e aplica métodos práticos para a resolução de fórmulas lógicas que devem ser interpretadas a partir de uma determinada teoria de fundo.

Resolver uma instância SMT, assim como uma instância do problema da satisfatibilidade booleana (SAT), significa encontrar valores para as variáveis que a tornem verdadeira, ou seja, satisfatível. Quando não existe uma valoração que torne a instância verdadeira, esta é dita insatísfatível. No entanto, as variáveis de uma instância SMT não são apenas booleanas, mas sim determinadas pela teoria utilizada.

O formalismo SMT vem sendo cada vez mais utilizado em aplicações industriais e em áreas mais fundamentais, como programação de computadores, planejamento, geração de casos de teste e outras aplicações formais devido aos avanços nos resolvedores [16].

O método proposto neste trabalho consiste em fazer a representação de uma propriedade da rede de Petri em uma instância SMT e então submetê-la a um resolvedor SMT a fim de que o resultado desta resolução defina se a rede atende a propriedade a ser verificada.

A representação das propriedades da rede para uma instância SMT dá-se pelo mapeamento de forma incremental dos lugares da rede em variáveis da instância SMT e as transições são o conjunto de regras SMT de transformação das variáveis que levam de um estado à outro.

A instância gerada representa todos os estados da rede para determinado número de disparos de transição. A verificação da propriedade consiste em avaliar se dentro destes estados é possível chegar à marcação desejada, uma vez que verificação de alcançabilidade e bloqueio podem ser vistos como o problema de chegar à uma marcação.

Neste trabalho, busca-se tirar proveito da velocidade dos resolvedores SMT atuais e do

fato de que muitos problemas podem ser transformados em instâncias SMT de maneira simples, tornando possível a análise de propriedades de redes de Petri.

A velocidade de resolução das instâncias dos problemas é uma característica dos resolvedores e é avaliada em competições internacionais. Neste trabalho busca-se analisar os resultados da resolução de instâncias utilizando um resolvedor SMT e o analisador de redes de Petri Lola [47] para verificar se a velocidade do resolvedor SMT reflete na análise das propriedades das redes.

O método apresenta uma nova abordagem para verificar a propriedade de alcançabilidade e de bloqueio em redes de Petri, podendo ser utilizado para criar novos programas para a análise destas propriedades.

Este trabalho está organizado da seguinte maneira: o Capítulo 2 aborda uma introdução a SAT(satisfatibilidade booleana) e apresenta o formalismo SMT, mostrando como é uma fórmula SMT e os métodos utilizados para a sua resolução. O Capítulo 3 apresenta o formalismo de rede de Petri, suas principais características, as propriedades utilizadas e alguns métodos para verificar a existência das mesmas.

O Capítulo 4 apresenta a transformação das propriedades da rede de Petri para instâncias SMT. No Capítulo 5 são apresentados os experimentos utilizados para teste do método. Por fim, o Capítulo 6 apresenta as conclusões e os trabalhos futuros.

CAPÍTULO 2

SATISFIABILITY MODULO THEORIES - SMT

O problema SMT está intimamente ligado ao problema SAT, sendo assim este capítulo abordará inicialmente uma introdução sobre SAT a fim de apresentar o problema e basear o entendimento dos resolvedores SMT.

2.1 SAT

Lógica proposicional é um sistema formal no qual fórmulas representam sentenças declarativas ou proposições, que podem ser verdadeiras ou falsas. Uma proposição é a afirmação ou negação de um fato, como por exemplo "1+1=10", onde o valor verdade desta proposição é falso caso seja considerado o sistema numérico decimal ou verdadeiro se considerado o sistema binário.

A partir das variáveis proposicionais é possível criar fórmulas compostas, utilizando os conectivos lógicos $\neg$ (negação), $\wedge$ (e), $\vee$ (ou). As regras que definem uma fórmula são:

- Um átomo (variável proposicional) é uma fórmula;
- Se α é uma fórmula, $\neg\alpha$ também é uma fórmula;
- Sendo α e β fórmulas, então $(\alpha \wedge \beta)$ e $(\alpha \vee \beta)$ também serão;
- Todas as fórmulas são geradas pela aplicação das regras anteriores.

A definição de quando uma valoração satisfaz uma fórmula tornado-a verdadeira é dada por:

- Uma variável proposicional α é satisfeita se seu valor for "$Verdadeiro$";
- $\neg\alpha$ tem o seu valor "$Verdadeiro$"se α assumir o valor "$Falso$";

- $(\alpha \wedge \beta)$ será "$Verdade$"se α e β forem "$Verdade$"e será "$Falso$"para qualquer outros valores para α e β;

- $(\alpha \vee \beta)$ será "$Falso$"se α e β forem "$Falsos$"e será "$Vedade$"para qualquer outros valores para α e β;

Uma fórmula CNF é uma conjunção de cláusulas, sendo uma cláusula uma disjunção de literais, onde um literal é uma variável ou sua negação. Uma cláusula é dita satisfeita quando pelo menos uma de suas variáveis assumiu o valor verdadeiro e não-satisfeita quando todas as suas variáveis assumem o valor falso. Uma fórmula é dita satisfeita se todas as clausulas forem verdadeiras e falsa para qualquer outra valoração.

O problema da satisfatibilidade booleana (SAT) consiste em verificar se existe uma valoração para as variáveis de uma fórmula que a torne verdadeira, ou seja, satisfatível. Uma fórmula CNF é formada por variáveis e conectivos lógicos. A Figura 2.1 mostra um exemplo de instância CNF.

$$\boxed{\neg(x1) \wedge (x1 \vee x2)}$$

Figura 2.1: Exemplo de instância CNF.

O problema SAT foi o primeiro a ser provado como NP-Completo [11] e mesmo não existindo um algorítmo eficiente para estes problemas, os resolvedores modernos conseguem resolver instâncias reais destes problemas. Muitas aplicações têm tirado proveito dos avanços das pesquisas nesta área, tais como aplicações em inteligencia artificial e métodos formais.

Os resolvedores SAT modernos utilizam duas principais abordagens, uma que se baseia no algorítmo DPLL(Davis-Putnam-Logemann-Loveland) [13, 14] e outra baseada em busca heurística [39]. O algorítmo DPLL[13] é um refinamento proposto por Martin Davis, George Logemann e Donald Loveland do algorítmo apresentado por Martin Davis e Hilary Putnam em [14].

O algorítmo DPLL (Algorítmo 1) consiste basicamente em selecionar uma variável x_i da instância f e escolher uma valor verdade para a mesma, como apresentado nas linhas 7 e 8. O BCP (*Boolean Constraint Propagation*) propaga a valoração escolhida onde as

variáveis e os operadores com valor verdade conhecido são substituídos por seus valores. O problema é resolvido recursivamente, como apresentado na linha 9. Se o resultado for verdade a linha 10 retorna verdadeiro, caso contrario as linhas 12 a 15 são executadas da mesma forma com a negação da variável x_i. Caso os dois casos não satisfaçam a instância, a linha 16 retorna que não é possível satisfazer a instância, uma vez que os dois valores possíveis para a variável já foram testados.

```
   Entrada: Uma instância f;
   Saída: Satisfatível ou Não Satisfatível;
 1 se f = 0 então
 2 |  retorna Não Satisfatível;
 3 fim
 4 se f = 1 então
 5 |  retorna Satisfatível;
 6 fim
 7 Escolhe uma variável x_i que aparece em f;
 8 A = A ∪ {x_i};
 9 se DPLL(BCP(f,x_i))=Satisfatível então
10 |  retorna Satisfatível;
11 fim
12 A = (A \ x_i) ∪ {¬x_i};
13 se DPLL(BCP(f,¬x_i))=Satisfatível então
14 |  retorna Satisfatível;
15 fim
16 retorna Não Satisfatível;
```

Algoritmo 1: DPLL apresentado em [30].

Muitas melhorias foram criadas baseando-se no algorítmo DPLL, possibilitando que os resolvedores SAT conseguissem resolver instâncias de problemas reais. Algumas das alterações mais impactantes do algorítmo DPLL foram apresentadas por Marques-Silva e Sakallah em [23]. Sendo a primeira, o retrocesso não cronológico, que consiste em encontrar a decisão que causou o conflito na fórmula SAT e desfazer todas as valorações até ela, diferentemente do que ocorria no DPLL padrão, onde somente a última valoração era desfeita. Outra grande melhoria apresentada foi o aprendizado, que busca aprender o motivo do erro possibilitando evitar que o mesmo erro seja repetido.

Atualmente os resolvedores SAT participam de competições, como por exemplo, o SAT-Race 2010 [2] onde os resolvedores resolvem instâncias com mais de 10.950.109 variáveis e 32.697.150 cláusulas de vários domínios, como criptografia e verificação de hardware e software. Esta competição apresenta o poder dos resolvedores. O CryptoMiniSat [1], primeiro colocado na competição, resolveu as instâncias com tempo médio de 138.3 segundos e Lingeling [19], o segundo colocado, com tempo médio de 111.4 segundos.

Mesmo com todos os avanços para a resolução de instâncias SAT existem alguns problemas onde são necessárias linguagens mais ricas (como a aritmética), onde uma teoria de apoio é necessária para capturar o significado das fórmulas, estas são chamadas fórmulas SMT [16]. Na Seção 2.2 é apresentado o formalismo SMT, como é uma fórmula SMT e os métodos utilizados para a sua resolução.

2.2 SMT

As primeiras referências à SMT (*Satisfiability Modulo Theories*) como um processo para decidir uma fórmula a partir de uma teoria de fundo aparecem por volta de 1970. Porém os estudos somente tiveram um incremento em 1980, com os trabalhos apresentados por Nelson e Oppen [27, 28], Shostak [41] e Boyer e Moore [6], onde são descritos procedimentos de decisão em métodos formais. Outro avanço foi apresentado por Nelson e Oppen em [26], onde são descritos métodos para combinar resoluções de mais de uma teoria de fundo, incrementando as possibilidades de uso deste resolvedores.

As variáveis em uma fórmula SMT, diferentemente de SAT, estão relacionadas à uma teoria de fundo. As fórmulas utilizam os conectivos lógicos $\neg$ (negação), $\wedge$ (e), $\vee$ (ou). As regras que definem uma fórmula são:

- Átomos são as variáveis da fórmula, valorados conforme a teoria de fundo utilizada;
- Os módulos são fórmulas constituídas por variáveis e símbolos da teoria de fundo ao qual o módulo pertence;
- Se α é uma fórmula formada por módulos, $\neg\alpha$ é uma fórmula;

- Sendo α e β fórmulas, então $(\alpha \wedge \beta)$ e $(\alpha \vee \beta)$ também serão;
- Todas as fórmulas são geradas pela aplicação das regras anteriores.

Os valores das fórmulas SMT são definidos pelas seguintes regras:

- Ás variáveis das fórmulas SMT assumem valores de acordo com as teorias de fundo, por exemplo, utilizamos a teoria dos inteiros, as variáveis podem assumir valores numéricos inteiros;
- Os módulos pode assumir valor de "$Verdadeiro$"ou "$Falso$"de acordo com a teoria de fundo utilizada, como por exemplo, o modulo $1 + 1 = 10$ é falso para a teoria dos inteiros que utiliza o sistema decimal;
- Sendo α um módulo, $\neg\alpha$ é "$Verdadeiro$"se α assumir o valor "$Falso$";
- Sendo α e β fórmulas, então $(\alpha \wedge \beta)$ será "$Verdade$"se α e β forem "$Verdade$"e "$Falso$"para qualquer outros valores para α e β;
- Sendo α e β fórmulas, então $(\alpha \vee \beta)$ será "$Falso$"se α e β forem "$Falsos$"e será "$Vedade$"para qualquer outros valores para α e β;

Na década de 1990 os resolvedores SMT tiveram um grande desenvolvimento com tentativas isoladas de desenvolver resolvedores mais rápidos, como Armando et. al [4] e Giunchiglia e Sebastiani [21], que aproveitando-se dos resolvedores SAT existentes conseguiram resolver instâncias grandes.

Os resolvedores de instâncias SMT são denominados por Franco e Martin [20] como resolvedores SMT, fazendo uma referência aos resolvedores SAT. Uma fórmula SMT é composta por uma fórmula SAT com módulos. Na Figura 2.2 temos uma instância SMT, onde $a \geq 3$ é um exemplo de módulo da instância. Se por exemplo, a instância da Figura 2.2 fosse interpretada com base na teoria dos inteiros, um valor que a satisfaz é $a = 5$, pois este valor para a variável a torna os módulos $\neg\, a \geq 6$,$a \geq 3$ e $a \geq 5$ verdadeiros, o que é suficiente para satisfazer a instância. Desta forma, pode-se verificar que a valoração das variáveis depende da teoria de fundo a partir da qual a instância deve ser interpretada.

$$\boxed{\neg(a \geq 3 \vee a \geq 6) \wedge (a \geq 3 \vee a \geq 5)}$$

Figura 2.2: Exemplo de instância SMT.

As regra para valoração de uma instância SMT são definidas pela teoria de fundo utilizada. Na Seção 2.3 são apresentadas as principais teorias de fundo definidas na SMT-Lib [43].

2.3 Teorias SMT

As teorias de fundo definem tipos e funções e associam um tipo às variáveis da fórmula SMT [9]. Por exemplo, a teoria Ints define os tipos inteiro e booleano e declara que qualquer variável numérica é do tipo inteiro. A partir das teorias e suas combinações são definidas as lógicas que limitam os tipos de expressões que podem ser utilizados.

Franco e Martin [20] afirmam que várias aplicações SMT trabalham com instâncias envolvendo duas ou mais teorias neste sentido a SMT-Lib[1] apresenta seis teorias, sendo elas: Core, Ints, Reals, Reals_Ints, ArraysEx, Fixed_Size_BitVectors. Na sequência é dada uma breve descrição de cada teoria de acordo com a documentação oficial da SMT-Lib [9].

A teoria Core define os elementos básicos da logica booleana e são incorporadas em todas as outras teorias. Dentre os quais estão o tipo booleano; as constantes $True$ e $False$; a operação *not* que faz a negação da variável ou clausula; a família de funções *and* e *or* ; a função de igualdade entre elementos do mesmo tipo; a função de desigualdade entre elementos do mesmo tipo e, por fim, as funções $if - then - else$ que tomam um primeiro argumento booleano e dois argumentos adicionais do mesmo tipo. Em suma nesta teoria são definidos as regras que determinam a valoração das fórmulas exceto para os módulos de outras teorias.

A teoria Int descreve os elementos básicos da aritmética inteira, como o tipo inteiro; define que todos os numerais são inteiros; define as funções soma ($+$), subtração ($-$), multiplicação ($*$), módulo (mod), divisão inteira (div) e valor absoluto (abs); define as

[1]SMT-LIB é uma iniciativa internacional que visa facilitar a pesquisa e desenvolvimento em Satisfatibilidade Modulo Teorias

funções de comparação que retornam um booleano maior (>), menor (<), menor igual (<=) e maior igual (>=).

A teoria Reals define o tipo real. Tanto numerais quanto decimais são reais. São apresentadas as funções soma (+), subtração (−), multiplicação (∗) e divisão inteira (*div*). Apresenta as funções de comparação maior (>), menor (<), menor igual (<=) e maior igual (>=) que retorna um booleano.

A teoria Reals_Ints combina a teoria Reals e a teoria Ints, porém não é exatamente a união das definições das duas teorias. Os tipos reais e inteiros são definidos. Os numerais são definidos como inteiros e os decimais são definidos como reais. Define as funções soma (+), subtração (−), multiplicação (∗), módulo (*mod*), divisão inteira (*div*) e valor absoluto (*abs*). Define as funções de comparação maior (>), menor (<), menor igual (<=) e maior igual (>=) que retornam um booleano. A função *to_real* mapeia um inteiro para um real. A função *to_int* mapeia um real para um inteiro. A função *is_int* mapeia um real para um booleano.

A teoria ArraysEx define os tipos de matrizes, um tipo parâmetro e funções para ler e escrever elementos de matrizes. É definido o novo tipo matriz que recebe dois parâmetros: o primeiro é o índice e o segundo é o tipo dos valores dos elementos da matriz. A função *select* extrai valores da matriz, e utiliza dois parâmetros, o primeiro é a matriz e o segundo é o índice, o resultado é o valor da matriz na posição do índice. A função *store* cria uma nova matriz com um valor modificado em determinado índice e recebe a matriz, o índice do valor a ser alterado e o novo valor para ser colocado no lugar do índice. Esta função retorna uma matriz com o valor modificado no índice. Duas matrizes são iguais se os valores em cada um do índices são iguais nas duas matrizes.

A teoria Fixed_Size_BitVectors descreve o comportamento de vetores de bits, onde um tipo novo de vetor de bits é definido para cada extensão de vetor. O vetor de bits é uma representação binária sem sinal de um número inteiro não negativo, com o bit menos significativo do lado direito. Um tipo BitVector n é definido para cada numeral maior que zero. A função $extract(i\ , j)$ extrai um sub-vetor do índice i até o índice j. As funções *bvnot* e *bvneg* são unárias. As funções *bvand*, *bvor* e *bvadd* são binárias. A função binária

de comparação é *bvult*.

De acordo com Cok [9] todas as teorias contém a teoria Core, responsável pela lógica proposicional. Considerando a possibilidade de combinar diferentes teorias, uma instância SMT pode de maneira simples representar problemas complexos onde instâncias SAT não são suficientes. A seguir são apresentados os modos de resolução de instâncias SMT e as abordagens utilizadas para isso.

2.4 Resolução de Instâncias SMT

Resolvedores SMT são classificados como ansioso ou preguiçoso de acordo como a abordagem utilizada para resolução. A abordagem preguiçosa é a mais utilizada nos resolvedores SMT. Essa abordagem tem conseguido tirar proveito tanto dos avanços dos resolvedores SAT modernos quanto dos resolvedores teoria, responsáveis por verificar se o modelo encontrado pelo resolvedor SAT satisfaz também a teoria utilizada .

Na subseção 2.4.1 é apresentada uma breve introdução à abordagem ansiosa e na subseção 2.4.2 uma apresentação da abordagem preguiçosa e suas variações.

2.4.1 Resolvedor Ansioso

Na abordagem denominada por Nieuwenhuis et. al [29] como ansiosa, a instância SMT é traduzida diretamente para uma instância SAT, evitando a necessidade de desenvolver resolvedores especializados para a teoria, bem como os processos de combinações destes resolvedores.

Para a tradução das instâncias SMT para SAT existem dois principais métodos: o Small Domain Instantiation definido por Pnueli et. al [34] e Per-Constraint Encoding, proposto por Seshia et. al [40].

Um modelo de resolvedor SMT ansioso é apresentado na Figura 2.3, onde é utilizado um codificador para converter a instância SMT diretamente para uma instância SAT e utiliza-se um resolvedor SAT para verificar a sua satisfatibilidade. No entanto esta abordagem é pouco utilizada uma vez que o tamanho das instância apos a conversão é

muito maior e todos os resolvedores SMT modernos competitivos utilizam a abordagem preguiçosa que será apresentada na subseção 2.4.2.

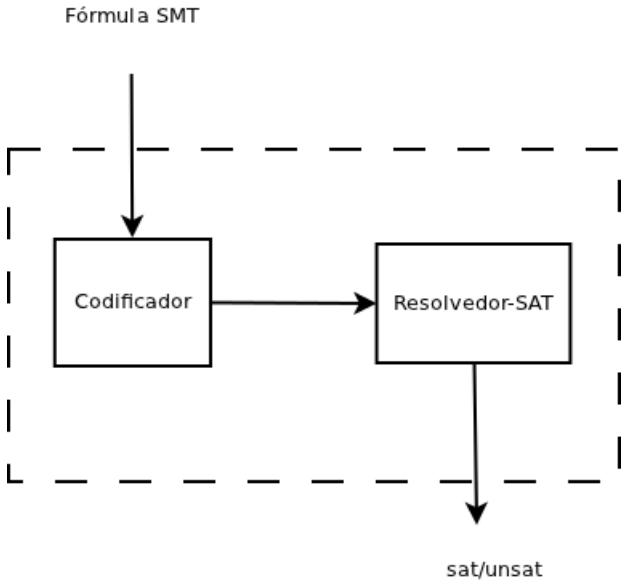

Figura 2.3: Modelo geral da abordagem ansiosa. Adaptado de [7]

2.4.2 Resolvedor Preguiçoso

A abordagem de resolução descrita nesta subseção é denominada por Nieuwenhuis et. al [29] como preguiçosa. Os resolvedores SMT preguiçosos utilizam os seguintes passos básicos:

1. Transformar a instância SMT em uma instância SAT.
2. Resolver a instância transformada com um resolvedor SAT.
3. Verificar se o modelo encontrado pelo resolvedor SAT satisfaz a instância SMT.
4. Caso o modelo não satisfaça a instância aprenda o modelo e retorne ao passo 1.

A instância SMT é inicialmente transformada em uma instância SAT. Para esta transformação os módulos são substituídos por variáveis, um exemplo é mostrado na Figura 2.4, onde a instância SMT da Figura 2.2 é transformada em uma instância SAT.

$$\boxed{\neg(a \geq 3 \vee a \geq 6) \wedge (a \geq 3 \vee a \geq 5)} \rightarrow \boxed{a \geq 3 = x1 \text{ , } a \geq 6 = x2 \text{ e } a \geq 5 = x3} \rightarrow \boxed{\neg(x1 \vee x2) \wedge (x1 \vee x3)}$$

Figura 2.4: Transformação da instância SMT apresentada na Figura 2.2 em uma instância SAT com a substituição dos módulos $a \geq 3$ por $x1$, $a \geq 6$ por $x2$ e $a \geq 5$ por $x3$.

Após a transformação, caso o resolvedor SAT necessite de um formato especial como CNF a instância deve ser convertida. Após a resolução da instância SAT um modelo[2] é devolvido ou é definido que a instância é insatisfatível.

Caso o modelo encontrado satisfaça os módulos da instância SMT, esta é considerada satisfatível. Caso a instância não seja satisfeita o modelo é aprendido[3] e a instância submetida novamente ao resolvedor SAT até que seja encontrado um modelo que satisfaça a instância ou que não seja mais encontrado nenhum modelo, ou seja, o resolvedor SAT retorne que a instância é insatisfatível. De acordo com [16] este processo sempre converge, pois há um número finito de átomos e, consequentemente, um número finito de modelos que podem ser criados.

O modelo geral de um resolvedor preguiçoso é apresentado na Figura 2.5, onde pode-se ver a integração entre o resolvedor SAT e o resolvedor teoria. Esta abordagem pode utilizar dois métodos de integração do resolvedor SAT com o resolvedor teoria: *online* ou offline. O método de integração deve ser definido durante a implementação do resolvedor.

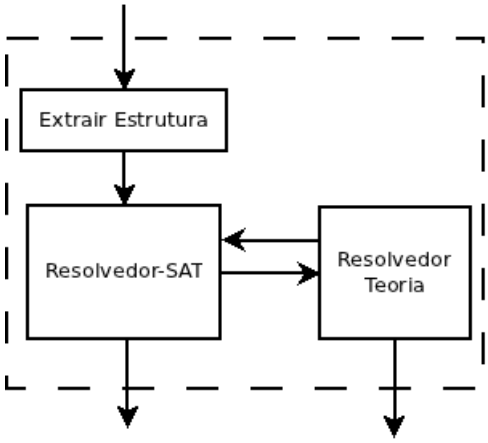

Figura 2.5: Modelo geral da abordagem preguiçosa. O método de integração pode ser *online* ou offline. Adaptado de [7].

[2]Modelo é uma valoração para as variáveis de uma instância que a satisfaz.

[3]Aprender: método utilizado pelo Resolvedor-SAT para não repetir uma valoração que não satisfaz a instância. Para maiores informações ver [36].

A descrição apresentada anteriormente utiliza um método definido como offline, pois o resolvedor teoria só começa a verificar se a valoração é válida para a instância SMT quando o resolvedor SAT encontra um modelo para a instância SAT.

O método de resolução *offline* é muito utilizado pela facilidade de implementação, dado que o mesmo não necessita de muitas alterações no resolvedor SAT ou de integrações muito complexas. Entretanto este método faz com que o resolvedor SAT trabalhe construindo uma valoração completa que não satisfaz a instância SMT, pois o processo de verificar se a valoração encontrada também satisfaz a instância SMT só é feita após o resolvedor SAT ter encontrado um modelo que satisfaz a instância SAT. A Figura 2.6 apresenta a resolução de uma instância SMT utilizando o método offline.

Converte a instância SMT para a instância SAT.

$$\boxed{\neg(a \geq 3 \vee a \geq 6) \wedge (a \geq 3 \vee a \geq 5)}$$

Submete a instância para o resolvedor SAT.

$$\Downarrow$$

$$\boxed{\neg(x1 \vee x2) \wedge (x1 \vee x3)}$$

Retorna o modelo da instância SAT.

$$\Downarrow$$

$$\boxed{\neg x1 \ x3}$$

Verifica se o modelo satisfaz a instância SMT.

$$\Downarrow$$

$$\boxed{\neg(a \geq 3) \ (a \geq 5)}$$

Se o modelo não satisfaz a instância SMT o modelo SAT é aprendido,

e a instância é submetida novamente ao resolvedor SAT .

$$\Downarrow$$

$$\boxed{\neg x2 \ x3}$$

Verifica se o modelo satisfaz a instância SMT.

$$\Downarrow$$

$$\boxed{\neg(a \geq 5) \ (a \geq 6)}$$

Se o modelo não satisfaz a instância SMT o modelo SAT é aprendido,

e a instância é submetida novamente ao resolvedor SAT .

$$\Downarrow$$

$$\boxed{x1 \ \neg x2}$$

Verifica se o modelo satisfaz a instância SMT.

$$\Downarrow$$

$$\boxed{(a \geq 3) \ \neg(a \geq 5)}$$

O modelo satisfaz a instância SMT finaliza a execução.

Figura 2.6: Resolução *offline* da instância SMT apresentada na Figura 2.2.

A método *offline* apesar de simples, não tem resultados muito bons e praticamente todos os resolvedores com resultados competitivos implementam o método denominado *online* que apesar de mais complexo obtém resultados melhores.

O método *online* submete valorações parciais para a verificação de satisfatibilidade da instância SMT. Caso uma valoração parcial seja identificada insatisfatível, o modelo é aprendido e o resolvedor SAT é forçado a fazer o retrocesso a fim de encontrar uma valoração que possa satisfazer a instância SMT. Dessa forma, quando o resolvedor SAT encontrar um modelo, este modelo sempre satisfará a instância SMT.

Na abordagem *online* o aprendizado de conflitos com a instância SMT ocorre durante a resolução da instância pelo resolvedor SAT. Este método de resolução polpa trabalho, já que impede que o resolvedor SAT trabalhe em uma valoração que não satisfará a instância SMT.

A maioria dos resolvedores SMT implementam o método online, pois o trabalho de encontrar uma valoração satisfatível para uma instância SAT não é trivial e pode demorar muito tempo.

A Figura 2.7 apresenta a resolução de uma instância SMT pelo método online, porém a interação entre o resolvedor SAT e o resolvedor teoria pode variar entre os resolvedores SMT, ou seja, nem todo resolvedor faz a interação com o resolvedor teoria a cada variável valorada. A frequência que esta interação é feita depende do foco de cada implementação.

Converte a instância SMT para a instância SAT.

$$\boxed{\neg(a \geq 3 \vee a \geq 6) \wedge (a \geq 3 \vee a \geq 5)}$$

Submete a instância para o resolvedor SAT.

$$\Downarrow$$

$$\boxed{\neg(x1 \vee x2) \wedge (x1 \vee x3)}$$

Valora a variável $x1 = Falso$.

$$\Downarrow$$

$$\boxed{\neg(a \geq 3)}$$

Verifica se a valoração satisfaz a instância SMT.

$$\Downarrow$$

$$\boxed{(x3)}$$

Valora a variável $x3 = Verdade$.

$$\Downarrow$$

$$\boxed{\neg(a \geq 3)\ (a \geq 5)}$$

Verifica que a valoração não satisfaz a instância SMT.

$$\Downarrow$$

$$\boxed{\neg(x1 \vee x2) \wedge (x1 \vee x3)}$$

Valora a variável $x1 = Verdade$.

$$\boxed{(a \geq 3)}$$

Verifica se a valoração satisfaz a instância SMT.

$$\Downarrow$$

$$\boxed{\neg(x2)}$$

Valora a variável $x2 = Falso$.

$$\boxed{(a \geq 3)\neg(a \geq 6)}$$

Verifica se a valoração satisfaz a instância SMT.

Figura 2.7: Resolução *online* da instância SMT apresentada na Figura 2.2.

As competições, como a SMT-Exec [42], mostram o poder dos resolvedores modernos, como o Z3 [15], CVC4 [5] e MathSat5 [18]. Durante a SMT-Exec, os resolvedores tem

um conjunto de 95.492 *benchmarks* (totalizando 59,2 GB) com 383 famílias utilizando mais de 22 lógicas. As famílias usadas nestes *benchmarks* podem ser de problemas industriais, randômicas, entre outras. Os três melhores resolvedores da SMT-Exec na lógica QF_UFLIA foram o Z3, que resolveu 195 instâncias em 4944,4 segundos, o CVC4 resolveu 158 instâncias em 7490,2 segundos e o MathSat5 resolveu 190 instâncias em 8921,5 segundos.

2.5 Considerações

Neste capítulo é apresentado o formalismo SAT, utilizado como base para os resolvedores do formalismo SMT. São descritas a resolução de fórmulas SMT e os principais métodos de resolução utilizados nos resolvedores modernos.

O formalismo SMT é utilizado neste trabalho pela facilidade de representar os problemas e a velocidade dos resolvedores SMT modernos. Após apresentado o formalismo SMT, no Capítulo 3 são apresentadas as definições de redes de Petri necessárias para o entendimento deste trabalho.

CAPÍTULO 3

REDES DE PETRI

Rede de Petri é um modelo gráfico e matemático utilizado para modelar sistemas a eventos discretos [25], proposto por Carl Adam Petri em sua dissertação "*Communication with automata*" [32] em 1962, utilizado para descrever a relação entre condições e eventos.

Este modelo pode ser aplicado para modelagem e análise de vários sistemas. Porém um ponto a se ressaltar é que, quanto mais geral a modelagem, mais complexa torna-se a sua análise, pois as redes de Petri tendem a se tornar muito complexas mesmo para pequenos problemas.

Na próxima seção serão apresentados alguns conceitos básicos de redes de Petri utilizados neste trabalho.

3.1 Conceitos

Uma rede de Petri R é apresentada por Cardoso e Valette [8] como uma quádrupla $R =< P, T, Pre, Post >$. Onde:

- P é o conjunto finito de lugares, de dimensão m;
- T é o conjunto finito de transições, de dimensão n;
- $Pre : P \times T \rightarrow \mathbb{N}$ são os lugares de entrada ou incidência anterior, sendo $\mathbb{N}$ o conjunto dos números naturais;
- $Post : P \times T \rightarrow \mathbb{N}$ são os lugares de saída ou incidência posterior, sendo $\mathbb{N}$ o conjunto dos números naturais.

Uma rede de Petri marcada N é aquela que contém marcas em seus lugares. Uma rede com marcas é representada por [8] como $N =< R, M_0 >$. Onde:

- R é uma rede de Petri;

- M_0 é a marcação inicial.

Uma Rede de Petri pode ser representada graficamente por um grafo bipartido com dois tipos de vértices, denominados lugares e transições. Os lugares geralmente são representados graficamente por círculos e as transições são representadas graficamente por retângulos, como pode ser visto na Figura 3.1, onde o círculo com o nome "Lugar1"é a representação gráfica de um lugar e o retângulo com nome "Transição1"é a representação de uma transição.

As arestas orientadas deste grafo são chamadas de arcos, que podem ter um peso associado, definindo quantas marcas devem ser consumidas ou geradas por uma transição. Quando este peso for um, o número no arco pode ser omitido.

Dois tipos especiais de transições podem aparecer em uma rede de Petri. Transições poço, quando o disparo desta consome mas não gera marcas, ou seja, não exitem arcos que ligam a transição a qualquer local de saída. Transição fonte, uma transição que não possui pré-condição para ser disparada, ou seja, não existem arcos que ligam qualquer local de entrada a esta transição.

As marcas ou fichas são círculos preenchidos que ficam localizados dentro dos lugares e são consumidos ou gerados pelas transições. Marcas representam os recursos disponíveis. Na Figura 3.1 pode-se ver a representação gráfica de uma rede de Petri com três lugares e uma transição, pode-se ver ainda que o arco que liga o *Lugar1* à *Transição1* tem peso dois.

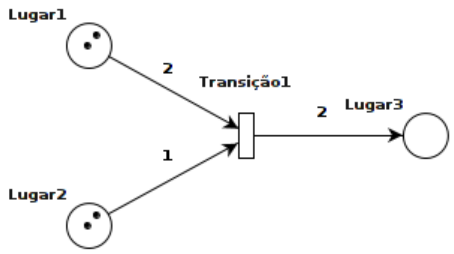

Figura 3.1: Representação gráfica de uma rede de Petri.

Uma Rede de Petri pode ser representada por matrizes que podem ser utilizadas para verificar as propriedades das redes. Na subseção 3.1.1 será apresentada esta notação.

3.1.1 Notação Matricial

Uma Rede de Petri pode ser definida como um conjunto de matrizes, onde cada rede é representada por duas matrizes e um vetor. A matriz $Pre(p,t)$ representa os arcos que ligam lugares a transições (Pré-condições). A matriz $Post(p,t)$ representa os arcos que ligam transições a lugares (Pós-condições). Além das duas matrizes um vetor M é utilizado para representar as marcas em cada um dos lugares da rede (recursos disponíveis).

A representação matricial da rede apresentada na Figura 3.2 é mostrada na Figura 3.3, onde pode-se notar que a matriz Pre contém os pesos dos arcos de entrada para cada transição. O valor zero representa a não existência de arcos. A matriz $Post$ contém os pesos dos arcos que ligam as transições a lugares e também contém zero para representar a ausência de arcos entre a transição e um determinado lugar. O vetor M contém o número de marcas contidas em cada lugar da rede.

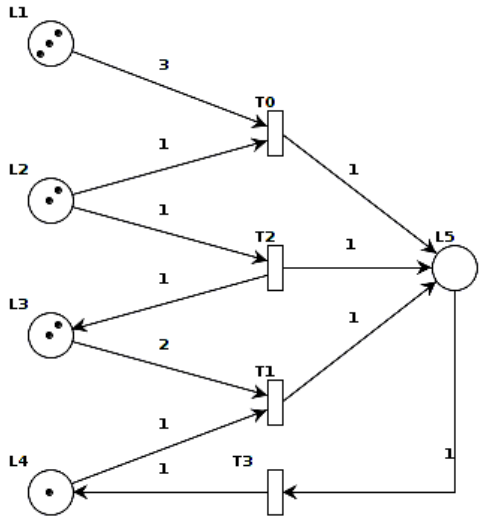

Figura 3.2: Exemplo de Rede de Petri.

$$Pre(p,t) = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Post(p,t) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad M = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

Figura 3.3: Representação matricial da rede de Petri apresentada na Figura 3.2.

A representação matricial da rede de Petri é utilizada para determinar algumas propriedades sem que seja necessário fazer a análise do grafo. Na próxima subseção é apresentado o conceito de disparo de transições, responsável pela dinâmica em uma rede de Petri.

3.1.2 Disparo de uma Transição

Um conceito muito importante em redes de Petri é o disparo de uma transição. Para que seja possível o disparo de uma transição é necessário que a mesma esteja sensibilizada, ou seja, que os lugares de entrada contenham no mínimo o número de marcas dos arcos de entrada, formalmente definido por Cardoso e Valette [8] como $\forall p \in P, M(p) \geqslant Pre(p,t)$. Onde $M(p)$ representa a marcação do lugar p e $Pre(p,t)$ são os lugares de entrada.

Caso a transição esteja sensibilizada, o disparo pode ocorrer fazendo com que as marcas sejam consumidas dos locais de entrada e geradas nos locais de saída. A Figura 3.4 apresenta a rede da Figura 3.1 após o disparo da transição $Transicão1$.

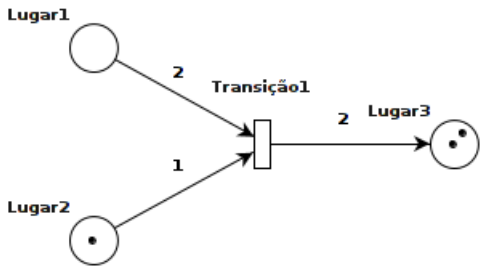

Figura 3.4: Rede de Petri da Figura 3.1 após o disparo da transição $Transicão1$.

O disparo de uma transição é descrito formalmente por Cardoso e Valette [8] como $\forall p \in P, M'(p) = M(p) - Pre(p,t) + Post(p,t)$. Onde $M'(p)$ é a marcação após o disparo da transição, que representa a marcação atual subtraídas as pré-condições da transição disparada e adicionando as suas pós-condições.

A equação fundamental permite a análise de acessibilidade das marcações, pois descreve a inserção e remoção de marcas nos lugares através de uma sequência de disparo de transições. A equação fundamental é definida de acordo com a Figura 3.5.

$$\boxed{M' \ = \ M \ + \ Cs \text{ com } M \ > \ 0 \text{ e } s \ > \ 0}$$

Figura 3.5: Equação fundamental da Rede de Petri [8].

Onde:

- C é a matriz de incidência onde $C = Post - Pre$;
- s é o vetor que descreve a sequência de disparo das transições, cada posição deste vetor representa uma transição e quantas vezes ela disparou;
- M é a marcação inicial;
- M' é a marcação alcançável.

A partir dos disparos das transições as redes são transformadas de um estado para outro. Serão apresentadas as propriedade de redes de Petri utilizadas neste trabalho na seção 3.2.

3.2 Propriedades de uma Rede de Petri

Nesta Seção serão abordadas as propriedades de redes de Petri que foram utilizadas neste trabalho. Na subseção 3.2.1 é apresentada a propriedade de bloqueio em redes de Petri e na subseção 3.2.2 é apresentada a propriedade de alcançabilidade.

3.2.1 Bloqueio

Uma rede de Petri está bloqueada se não existir nenhuma transição sensibiliza, ou seja, se não for possível gerar nenhum estado sucessor. Um bloqueio em uma rede de Petri é um conjunto de transições que não podem disparar [31]. Bloqueio é apresentado por Song e Lee [44] como um estado onde o fluxo da rede está parado à espera por um recurso.

Confirmar, prevenir e ainda determinar o que levou a rede a entrar em bloqueio em redes de Petri podem ajudar a reparar possíveis erros nos sistemas modelados. Um exemplo

ocorre em Song e Lee[44], que apresenta um método para corrigir problemas em programas ADA, onde o programa é transformado em uma rede e é utilizado um algoritmo para encontrar bloqueio na rede e eliminá-los.

Pode-se ver um exemplo de bloqueio em rede de Petri na Figura 3.6, onde a parte "a" apresenta a rede em estado inicial e a parte "b" apresenta a rede em bloqueio.

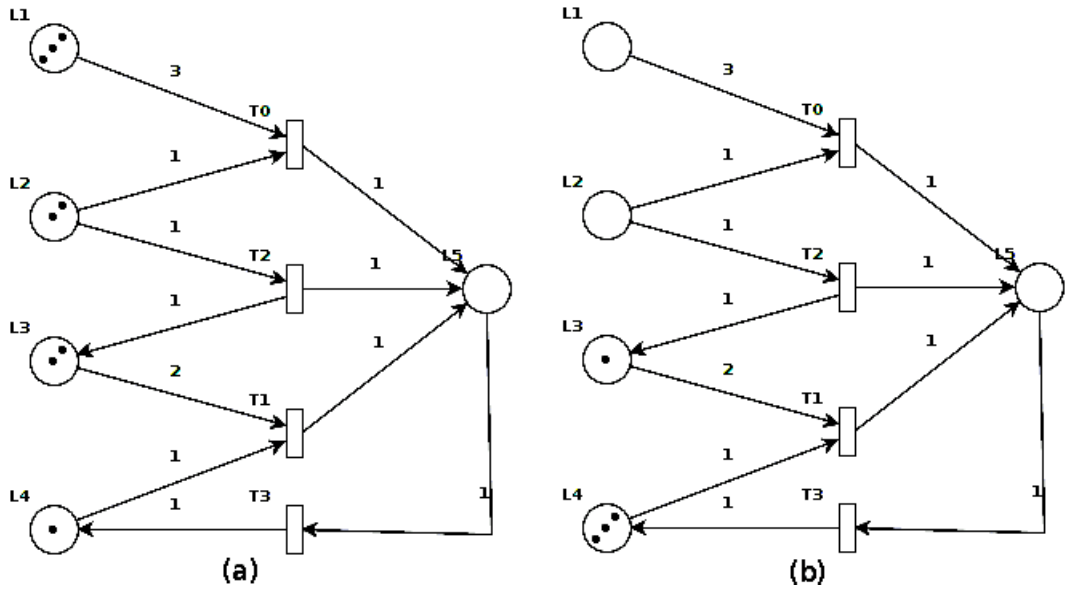

Figura 3.6: Na parte (a) pode-se ver a rede em seu estado inicial e na parte (b) a mesma rede bloqueada.

3.2.2 Alcançabilidade

Dada uma rede de Petri, uma marcação M' é dita acessível a partir de uma marcação inicial M_0, se existir uma sequência s de disparos de transições que leve a rede da marcação M_0 à marcação M'.

Marcações acessíveis de uma rede de Petri marcada é definida por Cardoso e Valette [8] como o conjunto das marcações possíveis a partir da marcação inicial, através de uma sequência de disparos definida como $A(R, M) = \{M_i, \exists s \quad M \xrightarrow{s} M_i\}$. Onde:

- $A(R, M)$ representa o conjunto de todas as marcações acessíveis a partir de M;
- s é a sequência de disparos;
- M é a marcação inicial;

- M_i é a marcação acessível a partir da marcação M.

Na seção 3.3 são apresentados alguns métodos para a verificação das propriedades apresentadas na seção 3.2.

3.3 Verificação

Um método para verificação de alcançabilidade em redes de Petri é a construção do grafo de marcações acessíveis. O grafo de marcações acessíveis é um grafo onde cada nó é uma marcação acessível e os arcos são etiquetados pela transição que possibilita a transformação de um estado a outro.

De acordo com Cardoso e Valette [8] o grafo de marcações acessíveis é equivalente à máquina de estados[37] da rede de Petri e também ocasiona a perda da possibilidade de diferenciar transições paralelas e conflitantes. Um exemplo de grafo de marcações acessíveis é apresentado na Figura 3.7, onde pode-se ver todos os possíveis estados para a rede apresentada na Figura 3.1.

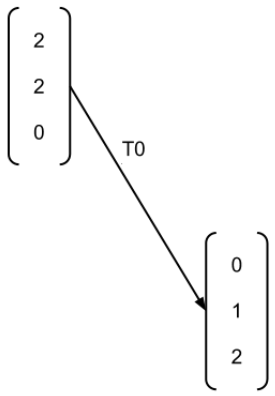

Figura 3.7: Grafo de Marcações Acessíveis da rede de Petri apresentada na Figura 3.1.

A Figura 3.8 apresenta uma rede onde o grafo de marcações acessíveis não pode ser utilizado, pois em redes com infinitos estados o grafo se torna infinito. Um outro método para verificação desta propriedade em redes de Petri é a utilização de desdobramento.

Desdobramento foi proposto por Mcmillan e Probst [24] para evitar a explosão combinatória do espaço de estados. Este processo consiste na transformação da rede de Petri

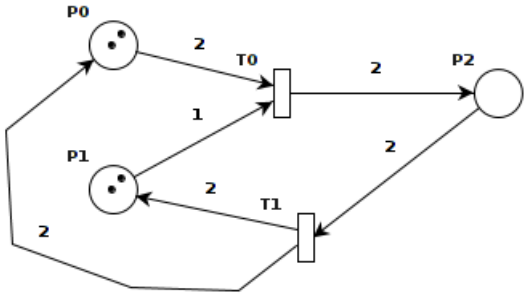

Figura 3.8: Alteração da rede da Figura 3.1 que torna impossível criar o grafo de marcações acessíveis.

condições evento em uma rede de ocorrência que preserva os marcações accessíveis. O desdobramento de uma rede é uma outra rede, normalmente infinita, porém com uma estrutura mais simples. Desdobramento é uma importante técnica para checar propriedades das redes de Petri.

O algoritmo proposto por Mcmillan e Probst [24] foi adaptado por Esparza et. al [17], pois o algoritmo anterior pode gerar desdobramentos maiores que o necessário. O algoritmo adaptado por Esparza et. al [17] é utilizado como base para diversas ferramentas de desdobramento.

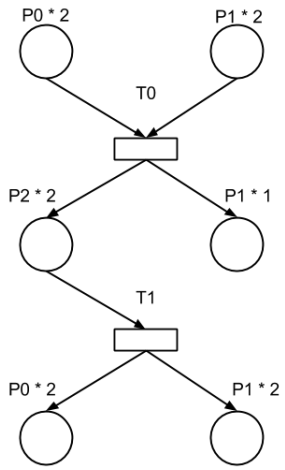

Figura 3.9: A Figura apresenta o desdobramento da rede apresentada na Figura 3.8.

Desdobramento é utilizado por grande parte dos programas de verificação de proprie-

dades de redes de Petri, como o Mcsmodels [22] que faz verificação de alcançabilidade e bloqueio em redes de Petri 1-limitadas e o sistema Lola [47] que é capaz de verificar as propriedades de bloqueio e alcançabilidade além de outras propriedades não abordadas neste trabalho utilizando redes k-limitadas. Como apresentado em [47] é possível utilizar a ferramenta para resolver problemas tais como, exploração de redes bioquímicas, solidez dos modelos de processos de negócios, entre outros problemas. Na Seção 3.4 é apresentada a ferramenta para análise de propriedades de rede de Petri que foi utilizada para verificação do método implementado.

3.4 Lola(*A Low Level Analyser Petri Net*)

Para a verificação do método apresentado no Cápitulo 4 foi utilizado o sistema Lola[47], criado para a validação de técnicas de redução do espaço de estados para redes de Petri lugar/transição, e implementa várias técnicas de redução do espaço de estados.

O sistema Lola não utiliza uma técnica de verificação completa, em alguns casos podendo ocasionar que este rode para sempre[47]. No entanto, seus métodos de redução tiveram um bom resultado apresentando tempo de resolução baixo para o conjunto de redes de Petri testadas. Outra limitação é a incapacidade de fazer checagem de uma marcação incompleta, ou seja, o número de marcas de cada lugar deve estar explícito na marcação a ser buscada. Outra limitação é a necessidade de expressar uma marcação exata não sendo possível a verificação de um lugar que contenha um número de marcas maior ou menor do que o descrito no estado buscado.

3.5 Considerações

Neste capítulo é apresentado o formalismo de redes de Petri, muito utilizado para modelar sistemas discretos. São apresentadas as propriedades bloqueio e alcançabilidade que serão verificadas utilizando o método proposto.

O capítulo apresenta o funcionamento das redes de Petri e como pode ser feita a verificação destas propriedades. São apresentados os principais métodos de verificação

de propriedades de redes de Petri que servem como base para o desenvolvimento deste trabalho. Por fim é apresentada a ferramenta de análise de propriedades em rede de Petri que será utilizado neste trabalho para verificar o método descrito no Capítulo 4.

Após as definições apresentadas no Capítulo 2 e Capítulo 3, será apresentado no Capítulo 4 um método de transformação da rede de Petri para uma instância SMT para que seja possível verificar as propriedades nas redes de Petri utilizando-se de um resolvedor SMT.

CAPÍTULO 4

TRANSFORMAÇÃO DAS PROPRIEDADES DE REDES DE PETRI EM INSTÂNCIAS SMT

A modelagem de redes de Petri para instâncias SMT permite o uso de um resolvedor SMT para a análise da instância. Dessa forma pode-se verificar a ocorrência de determinadas propriedades de uma rede de Petri, como alcançabilidade e existência bloqueio. Este Capítulo apresenta o método proposto para a transformação das propriedades de redes de Petri em uma instância SMT.

4.1 Método Proposto

Desenvolveu-se neste trabalho um método onde cria-se uma variável na instância SMT para cada lugar na rede. Cada transição da rede é uma regra de transformação das variáveis, ou seja, definem a mudança de estados dos lugares da rede. Expansão é a quantidade de vezes que o conjunto das regras é aplicado.

Cada variável é representada pelo nome do lugar na rede de Petri e o número da expansão a qual esta variável pertence. Por exemplo, as variáveis que representam os lugares da rede apresentada na Figura 4.1 em seu estado inicial com expansão zero, ou seja, sem que nenhuma transição tenha sido disparada são:

- $P1\_e0$, representa o lugar $P1$;
- $P2\_e0$, representa o lugar $P2$;
- $P3\_e0$, representa o lugar $P3$;
- $P4\_e0$, representa o lugar $P4$.

A marcação inicial da rede é definida na instância SMT por uma conjunção de igualdade entre as variáveis do estado inicial e o valor da marcação inicial de cada lugar

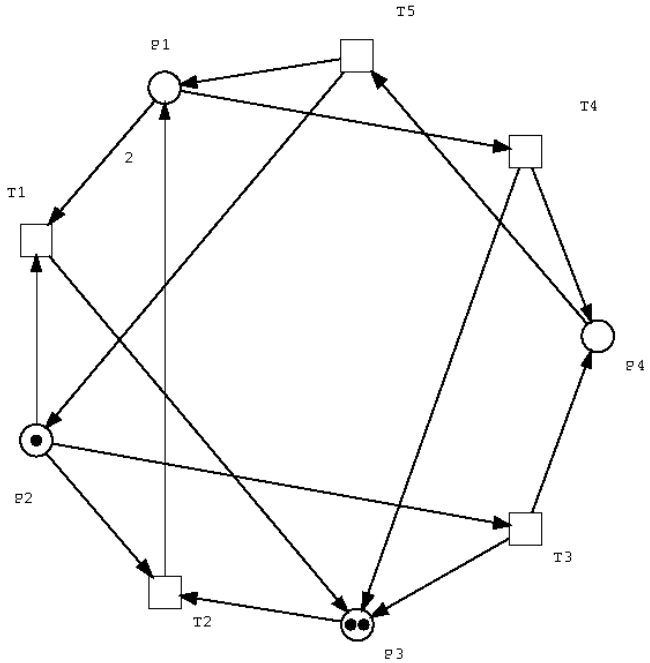

Figura 4.1: Rede de Petri.

na rede. A marcação inicial pode ser definida como: dada uma rede de Petri marcada $N =< R, M_0 >$, o estado inicial da rede é dado por $\forall p \in P,\ p\_e0 = M_0(p)$ onde p_e0 é a variável que representa o lugar p com expansão zero da instância e $M_0(p)$ é a marcação inicial da rede de Petri para o lugar p.

A Figura 4.2 representa a descrição da marcação inicial da rede da Figura 4.1.

$$(assert\ (and\ (=\ P1\_e0\ \ 0)(=\ P2\_e0\ \ 1)(=\ P3\_e0\ \ 2)(=\ P4\_e0\ \ 0)\ ))$$

Figura 4.2: Conjunção que representa o estado inicial da rede da Figura 4.1.

Dada a rede N apresentada anteriormente, as transições são definidas por $\forall x \in X, \forall p \in P, \exists t \in T,\ p\_ex' = p\_ex - Pre(t,p) + Pos(t,p)$ com $X \in \mathbb{N}$, onde x' representa o índice da expansão atual e x o índice a expansão anterior, p_ex' representa a marcação do lugar p na expansão ex', p_ex representa a marcação do lugar p na expansão anterior, Pre e Pos representam os pesos dos arcos de entrada e saída respectivamente e definem quantas marcas devem ser retiradas ou inseridas pela transição representada na variável p_ex.

Em uma rede de Petri uma transição é responsável por transformar a marcação dos lugares. Neste método uma transição é disparada a cada expansão da instância e transforma as variáveis que representam seus locais de entrada e de saída. A escolha de qual transição é disparada a cada expansão é realizada pelo resolvedor SMT.

A transformação dos lugares de entrada é dada por ($p\_ex \ = \ (\ p\_ex' \ - \ Pre(t,p))\ )$ e os locais de saída por ($p\_ex \ = \ (p\_ex' \ + \ Pos(t,p))\ )$, onde p_ex' é a variável que representa o lugar p na expansão ex', e p_ex corresponde ao lugar p na expansão ex. Por fim, a variável $Pre(t,p)$ e $Pos(t,p)$ corresponde ao peso dos arcos que ligam o lugar p à transição t ou vice-versa e consequentemente o número de marcas a serem consumidas ou geradas.

Os locais que não são modificados pelo disparo da transição devem ter os seus valores propagados para a próxima expansão, para isso é inserido a condição ($p\_ex \ = \ p\_ex'$). É inserida ainda a condição ($p\_ex \ >= \ 0$) para que não seja possível a existência de variáveis de entrada com número de marcas negativas.

A Figura 4.3 apresenta a primeira expansão da rede apresentada na Figura 4.1. Esta instância descreve o comportamento da rede para o disparo de uma transição qualquer e quais os estados podem ser alcançados. Na Figura 4.3 as variáveis com $\_e0$ são pertencentes ao estado inicial e as com $\_e1$ pertencem ao estado após o disparo da primeira transição.

```
(assert (or
(and(= P4_e1 (- P4_e0 1)) (= P1_e1 (+ P1_e0 1))
  (= P2_e1 (+ P2_e0 1)) (> P4_e1 0)
  (= P3_e1 P3_e0) )
(and(= P1_e1 (- P1_e0 2)) (= P2_e1 (- P2_e0 1))
  (= P3_e1 (+ P3_e0 1)) (> P1_e1 0) (> P2_e1 0)
  (= P4_e1 P4_e0) )
(and(= P2_e1 (- P2_e0 1)) (= P3_e1 (- P3_e0 1))
  (= P1_e1 (+ P1_e0 1)) (> P2_e1 0) (> P3_e1 0)
  (= P4_e1 P4_e0) )
(and(= P2_e1 (- P2_e0 1)) (= P3_e1 (+ P3_e0 1))
  (= P4_e1 (+ P4_e0 1)) (> P2_e1 0)
  (= P1_e1 P1_e0) )
(and(= P1_e1(- P1_e0 1)) (= P4_e1 (+ P4_e0 1))
  (= P3_e1 (+ P3_e0 1)) (> P1_e1 0)
  (= P2_e1 P2_e0) )
))
```

Figura 4.3: Representação das cinco transições da rede da Figura 4.1 para a primeira expansão na instância SMT.

O processo de expansão da instância consiste em adicionar à instância a possibilidade de mais um disparo de transição. Para tanto a instância é acrescida de um novo trecho com as definições das transições incrementando os índices de expansão, como é apresentado na Figura 4.4. O processo de expansão é feito até que o estado desejado seja alcançado.

```
(assert (or
(and(= P4_e1 (- P4_e0 1)) (= P1_e1 (+ P1_e0 1))
  (= P2_e1 (+ P2_e0 1)) (> P4_e1 0) (= P3_e1 P3_e0))
(and(= P1_e1 (- P1_e0 2)) (= P2_e1 (- P2_e0 1))
  (= P3_e1 (+ P3_e0 1)) (> P1_e1 0) (> P2_e1 0) (= P4_e1 P4_e0) )
(and(= P2_e1 (- P2_e0 1)) (= P3_e1 (- P3_e0 1))
  (= P1_e1 (+ P1_e0 1)) (> P2_e1 0) (> P3_e1 0) (= P4_e1 P4_e0) )
(and(= P2_e1 (- P2_e0 1)) (= P3_e1 (+ P3_e0 1))
  (= P4_e1 (+ P4_e0 1)) (> P2_e1 0) (= P1_e1 P1_e0) )
(and(= P1_e1(- P1_e0 1)) (= P4_e1 (+ P4_e0 1))
  (= P3_e1 (+ P3_e0 1)) (> P1_e1 0) (= P2_e1 P2_e0) )
))
(assert (or
(and(= P4_e2 (- P4_e1 1)) (= P1_e2 (+ P1_e1 1))
  (= P2_e2 (+ P2_e1 1)) (> P4_e2 0) (= P3_e2 P3_e1))
(and(= P1_e2 (- P1_e1 2)) (= P2_e2 (- P2_e1 1))
  (= P3_e2 (+ P3_e1 1)) (> P1_e2 0) (> P2_e2 0) (= P4_e2 P4_e1))
(and(= P2_e2 (- P2_e1 1)) (= P3_e2 (- P3_e1 1))
  (= P1_e2 (+ P1_e1 1)) (> P2_e2 0) (> P3_e2 0) (= P4_e2 P4_e1) )
(and(= P2_e2 (- P2_e1 1)) (= P3_e2 (+ P3_e1 1))
  (= P4_e2 (+ P4_e1 1)) (> P2_e2 0) (= P1_e2 P1_e1) )
(and(= P1_e2(- P1_e1 1)) (= P4_e2 (+ P4_e1 1))
  (= P3_e2 (+ P3_e1 1)) (> P1_e2 0) (= P2_e2 P2_e1) )
))
```

Figura 4.4: Representação das cinco transições da rede da Figura 4.1 para duas expansões na instância SMT.

4.1.1 Verificar Alcançabilidade

O estado que se deseja alcançar pode ser descrito como: dada uma rede de Petri marcada $N =< R, M_0 >$, o estado que se deseja alcançar é definido como $\exists x \in \mathbb{N}, \forall p \in$

P, $p\_ex = M'(p)$, onde ex representa a última expansão necessária para alcançar a marcação desejada, p_ex representa a variável correspondente ao lugar p na última expansão feita na instância e $M'(p)$ representa a marcação desejada para o lugar p.

Para descrever na instância SMT o estado que se deseja atingir é feita uma conjunção da marcação que se deseja atingir para cada lugar. O modelo não necessita que todas as variáveis sejam expressas, somente as que são necessárias para verificar a propriedade desejada. A Figura 4.5, apresenta um exemplo de estado desejado para a rede da Figura 4.1 este estado é alcançável disparando a transição $T2$.

$$(assert\ (and\ (=\ P1\_e1\ \ 1)(=\ P2\_e1\ \ 0)(=\ P3\_e1\ \ 1)(=\ P4\_e1\ \ 0)\ ))$$

Figura 4.5: Representação SMT de um estado desejado da rede apresentada na Figura 4.1.

4.1.2 Verificar Bloqueio

Para verificar se a rede possui bloqueio somente é necessário que o estado final a ser buscado seja qualquer um onde nenhuma transição esteja habilitada, ou seja, que pelo menos um local de entrada para cada transição não tenha o número de marcas necessárias para o disparo. Na Figura 4.6 pode-se ver a modelagem do estado desejado da rede para que o resolvedor SMT busque a existência de bloqueio.

$$\begin{array}{l}(assert\ (and\\ (or\ (<\ \ P1\_e1\ 2)\ (<\ \ P2\_e1\ 1))\\ (or\ (<\ \ P2\_e1\ 1)\ (<\ \ P3\_e1\ 1))\\ (<\ \ P2\_e1\ 1)\\ (<\ \ P1\_e1\ 1)\\ (<\ \ P4\_e1\ 1)\\ ))\end{array}$$

Figura 4.6: Conjunção que representa um exemplo de busca por bloqueio para a rede da Figura 4.1.

Após a transformação, a instância é checada utilizando um resolvedor SMT. Caso a instância seja satisfatível, a marcação é alcançável, caso contrário, a instância é expandida

e checada novamente até que se encontre uma valoração satisfatível.

Este processo de incremento e checagem pode não terminar caso a marcação ou propriedade buscada não possa ser atingida, portanto deve ser determinado um número máximo para que este processo seja realizado. Devido à este fator, algumas propriedades de redes de Petri não podem ser checadas utilizando essa modelagem. É impossível por exemplo, determinar se uma marcação qualquer é realmente não alcançável pela impossibilidade de determinar se todas as marcações já foram verificadas.

Durante o desenvolvimento do método apresentado foram utilizados algumas redes para testes. Durante os testes foram analisados alguns problemas, o primeiro deles foi a resolução de laços elementares, uma vez que a instância da maneira que foi representada não conseguia lidar com este tipo de estrutura. Outro problema foi a impossibilidade de disparo de mais de uma transição em uma expansão, o método criado para lidar com estes problemas é apresentado na Seção 4.2.

4.2 Melhorias no Método

Durante as pesquisas foram encontradas algumas modificações que poderiam melhorar o método, e alguns destes métodos foram implementados, as descrições destas melhorias são apresentadas nas próximas subseções.

4.2.1 Verificação de Variável de Entrada

A primeira melhoria no método foi a retirada da verificação da variável que representa o lugar de entrada da transição. Essa verificação foi inserida na declaração das variáveis, já que nenhuma variável pode ser negativa. Esta estrutura tem que ser inserida pelo fato do SMT não ter um tipo inteiro sem sinal, portanto, toda vez que se declara uma variável nova na instância é inserida uma cláusula que a impede de assumir valores negativos. Por exemplo, após declarar a variável $P1$ é inserida a cláusula $(assert(P1 >= 0))$, que impede que $P1$ assuma valores negativos. Esta retirada elimina a necessidade de inserir a mesma verificação em diferentes transições, como era feito anteriormente, diminuindo o tamanho

da instância.

4.2.2 Laços Elementares

A modelagem anterior não conseguia lidar com os laços elementares nas redes, por exemplo, se p_ex' não conter marcas e $Pre(t,p) = 1$ e $Pos(t,p) = 2$, o resultado do disparo da transição $p\_ex = p\_ex' - Pre(t,p) + Pos(t,p)$ vai ser p_ex com uma marca, mesmo com p_ex' sem marcas suficientes para o disparo. Isso porque o resultado de $-Pre(t,p) + Pos(t,p)$ é um, mesmo com a pré-condição não satisfeita. Deste modo, a transição disparava gerando resultados errados.

Para resolver este problema, criou-se uma alternativa alterando a estrutura da rede, eliminando estes laços através da inserção de lugares complementares. Este método cria um novo lugar e uma nova transição para cada laço encontrado na rede, ocasionando um crescimento da rede e aumento no número de disparos de transições necessárias para chegar ao estado desejado. A Figura 4.7 apresenta uma rede com sete laços elementares.

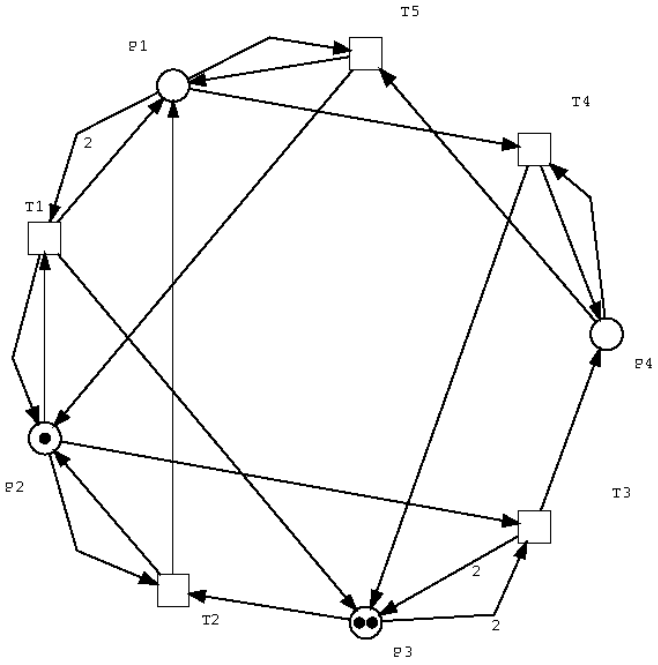

Figura 4.7: Rede de Petri com laços elementares.

A Figura 4.8 apresenta a rede da Figura 4.7 sem os laços elementares, com as estruturas necessárias para a eliminação dos laços aparecendo em destaque. Pode-se notar que a estrutura da rede aumenta consideravelmente e o número de disparos para atingir uma

determinada marcação também, consequentemente o tamanho da instância aumenta bem como seu número de expansões. Esta alteração não altera nenhuma propriedade da rede, somente elimina os laços.

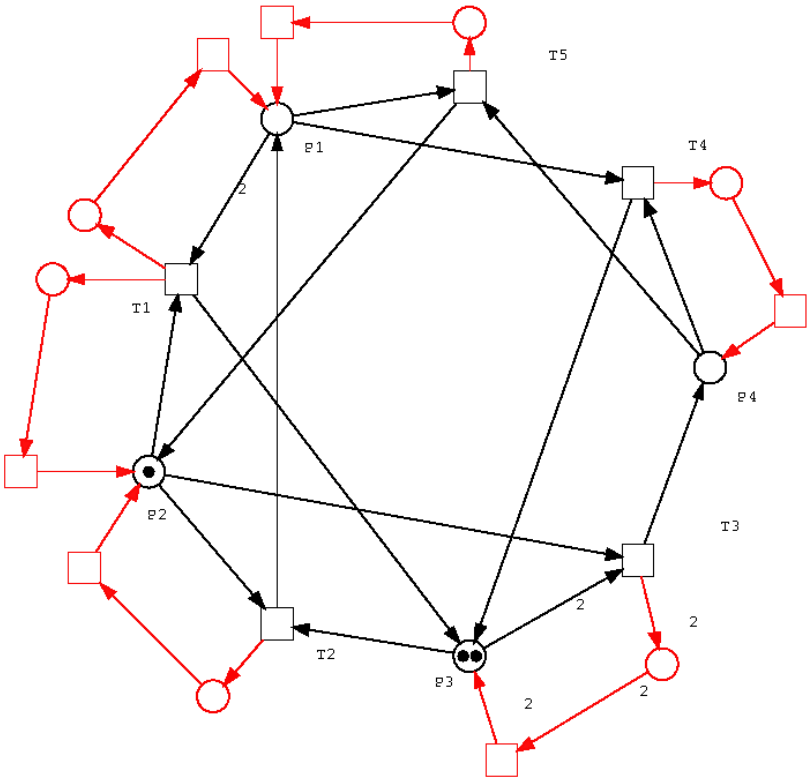

Figura 4.8: Rede de Petri apresentada na Figura 4.7 com a inserção dos lugares complementares.

Para evitar a necessidade de inserir estruturas na rede e evitar o consequente crescimento da instância, criou-se uma alternativa criando uma nova estrutura para lidar com os laços elementares.

Primeiramente é feita uma análise na estrutura da rede a fim de identificar cada um dos laços elementares contidos na rede. Após encontrar, os laços são expressos da seguinte forma: dada uma rede de Petri marcada $N =< R, M_0 >$ um laço é $\forall x \in X, \forall p \in P, \exists t \in T,\ p\_ex = p\_ex' - Pre(t,p) + Pos(t,p) \wedge p\_ex' >= Pre(t,p)$, ou seja, deixa-se de verificar se a transição gerou um estado onde alguma variável ficou com valor negativo e passa a se verificar se a variável na expansão anterior tem marcas suficientes para um determinado disparo. Esta verificação elimina o problema de retirar e colocar marcas em um mesmo local com um disparo.

A Figura 4.9 apresenta a representação da transição $T1$ da rede apresentada na Figura

4.7. Na Figura 4.9 parte **a)** é mostrada a representação que não suportava laços, pois analisando o trecho da instância pode-se concluir que mesmo se o lugar $P1\_e0$ não tiver nenhuma marca ainda é possível encontrar uma valoração, contrariando o funcionamento da rede, o que justifica a retirada dos laços com a inserção de lugares complementares. A parte **b)** mostra a alteração feita na instância para representar os laços elementares, impedindo que uma transição dispare sem ter o número de marcas necessárias.

$(and$

$(= \ P1\_e1 \ (+ \ (- \ P1\_e0 \ 2) \ (+ \ P1\_e0 \ 2) \ )) \ (>= \ P1\_e1 \ 0)$

$(= \ P2\_e1(+(-P2\_e0 \ 1)(+P2\_e0 \ 1) \ )) \ (>= \ P2\_e1 \ 1)$

$(= \ P3\_e1 \ (+ \ P3\_e0 \ 1))$

$(= \ P4\_e1 \ P4\_e0) \ )$

a)

$(and$

$(= \ P1\_e1 \ (+ \ (- \ P1\_e0 \ 2) \ (+ \ P1\_e0 \ 2) \ )) \ (>= \ P1\_e0 \ 2)$

$(= \ P2\_e1(+(- \ P2\_e0 \ 1)(+ \ P2\_e0 \ 1) \ )) \ (>= \ P2\_e0 \ 1)$

$(= \ P3\_e1 \ (+ \ P3\_e0 \ 1))$

$(= \ P4\_e1 \ P4\_e0) \ )$

b)

Figura 4.9: A parte a) mostra a representação da transição $T1$ da Figura 4.7 sem suporte a laços elementares na instância e a parte b) mostra a representação da mesma transição com alteração realizada na instância.

O método apresentado na parte **b)** da Figura 4.9 não necessita que seja inserida nenhuma nova variável e o crescimento da instância é menor que a inserção de lugares complementares na rede.

4.2.3 Múltiplos Disparos de uma Transição

Pretendendo um menor tempo ao fazer a checagem da instância, foi desenvolvido um método que insere uma nova variável para representar o número de vezes que uma

determinada transição dispara em uma expansão. Esta modificação possibilita que uma mesma transição possa ser disparada mais de uma vez em cada expansão da instância.

A nova definição de transição é dada por: em uma rede de Petri marcada $N =< R, M_0 >$ uma transição é $\exists n \in \mathbb{N}, \forall x \in X,\ \forall p \in P,\ \exists t \in T,\quad p\_ex = p\_ex' - (n\_t\_x * Pre(t,p)) + (n\_t\_x * Pos(t,p)) \wedge (p\_ex' - ((n\_t\_x - 1) * Pre(t,p))) >= Pre(t,p)$, onde n_t_x é o número de vezes que a transição disparou em uma determinada expansão da instância.

Além da modificação da representação da transição, foi necessária a inserção de uma cláusula para evitar que a variável que representa o número de disparos da transição tenha um valor diferente do número real de disparos, que pode ocasionar uma sequencia de disparos errada.

A valoração da variável que determina o número de disparos de uma transição em uma expansão é definida pelo resolvedor SMT, respeitando as regras da rede de Petri representadas na instância, como ter marcas suficientes para disparar uma transição determinado número de vezes, que é representado por $(p\_ex' - ((n\_t\_x - 1) * Pre(t,p))) >= Pre(t,p)$. A Figura 4.10 apresenta a representação da transição $T1$ da rede apresentada na Figura 4.7 conforme a nova definição apresentada.

$$
\begin{array}{l}
(and \\
(= \ P1\_e1 \ (+ \ (- \ P1\_e0 \ (* \ n\_T1 \ 2) \ ) \\
(+ \ P1\_e0 \ (* \ n\_T1 \ 2)) \ )) \\
(>= \ (- \ P1\_e0 \ (* \ (- \ n\_T1 \ 1) 2) \ ) \ 2) \\
(= \ P2\_e1 (+ \ (- \ P2\_e0 \ (* \ n\_T1 \ 1)) \\
(+ \ P2\_e0 \ (* \ n\_T1 \ 1)) \ )) \\
(>= \ (- \ P2\_e0 \ (* \ (- \ n\_T1 \ 1) \ 1)) \ 1) \\
(= \ P3\_e1 \ (+ \ P3\_e0 \ (* \ n\_T1 \ 1))) \\
(= \ P4\_e1 \ P4\_e0) \ )
\end{array}
$$

Figura 4.10: Representação da transição $T1$ com possibilidade de ser disparada mais de uma vez por expansão.

O método apresentado na Figura 4.10 faz com que a instância cresça, pois além de

uma nova variável para cada transição, a representação das transições também tem um aumento em seu tamanho. Além de uma restrição extra para garantir que só é possível que uma variável que represente uma transição tenha um valor diferente de zero se a transição tiver disparado. Este método possui como vantagem a possibilidade de disparo de uma transição várias vezes em uma expansão da instância, diminuindo o número de expansões necessárias para chegar a determinado estado.

4.2.4 Disparo de Transições Paralelas

Após feita uma análise da alteração apresenta na Seção 4.2.3 foi modelada uma forma onde, apenas alterando o modo de propagar os valores das variáveis não alteradas pode-se disparar mais de uma transição paralela por expansão da instância. Uma transição é dita paralela com outra quando não compartilham nenhum lugar entre si.

Para que seja possível disparar mais que uma transição paralela em uma mesma expansão para que isto ocorra é necessário uma alteração na propagação das variáveis não alteradas para que o resolvedor possa decidir qual propagação deve ser feita na expansão, possibilitando que as transições paralelas possam disparar ao mesmo tempo, uma vez que as regras aplicadas a duas transições paralelas separadamente ou juntas somente diferem na propagação das variáveis não alteradas,ou seja se devem ser propagadas as variáveis não alteradas por uma uma transição ou pelo conjunto de transições disparadas.

No entanto fazer a verificação de todas as transições paralelas e suas combinações é muito custoso. A Figura 4.11 apresenta uma rede de Petri com transições paralelas que será utilizada para a representação do método.

A Figura 4.12 apresenta a representação das transições paralelas $t2$ e $t5$ da Figura 4.11. O primeiro *and* representa a transição $t2$, sendo que suas três primeiras linhas representam seu comportamento conforme abordado na Seção 4.2.3. A quarta linha apresenta um *or* com as possíveis propagações. O primeiro *and* das propagações define a propagação para o disparo da transição $t2$, o segundo define a propagação para o disparo das transições $t2$ e $t4$ em paralelo e o último *and* define a propagação para o disparo em paralelo das transição $t2$ e $t5$.

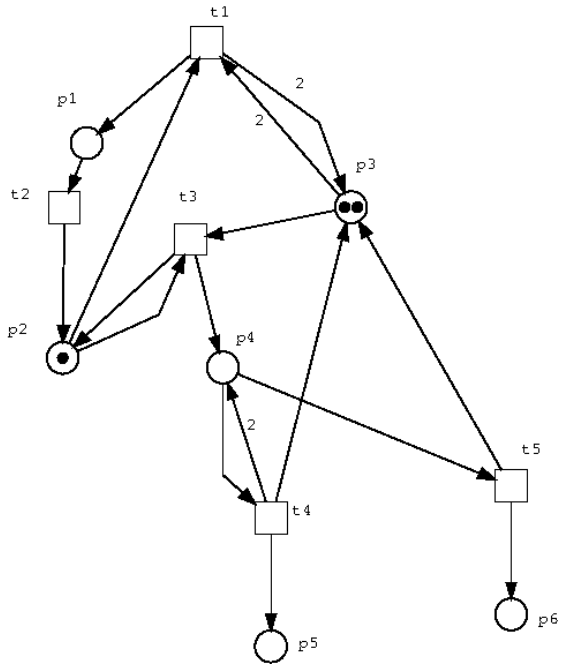

Figura 4.11: Rede com transições paralelas

A definição da transição $t5$ segue o mesmo processo. As duas últimas linhas são para garantir que as variáveis que representam os números de disparos das transições tenham um valor condicionado à valoração de todas as variáveis que fazem parte da representação das transições.

```
(or
(! (and
(= p1_e1 (- p1_e0 (* n_t2_e1 1)))
(>= 1(- p1_e0 (* (- n_t2_e1 1)1)))
(= p2_e1 (+ p2_e0 (* n_t2_e1 1)))
(or
(and(= p3_e1 p3_e0)(= p4_e1 p4_e0)(= p5_e1 p5_e0)(= p6_e1 p6_e0)(> n_t2_e1 0))
(and(= p6_e1 p6_e0)(> n_t2_e1 0)(> n_t4_e1 0))
(and(= p5_e1 p5_e0)(> n_t2_e1 0)(> n_t5_e1 0))
) ) : named t2_e1)
(! (and
(= p4_e1 (- p4_e0 (* n_t5_e1 1)))
(>= 1 (- p4_e0 (* (- n_t5_e1 1)1)))
(= p3_e1 (+ p3_e0 (* n_t5_e1 1)))
(= p6_e1 (+ p6_e0 (* n_t5_e1 1)))
(or
(and(= p1_e1 p1_e0)(= p2_e1 p2_e0)(= p5_e1 p5_e0)(> n_t5_e1 0))
(and(= p5_e1 p5_e0)(> n_t2_e1 0)(> n_t5_e1 0))
) ) : named t5_e1)
)
(or (and(> n_t2_e1 0)(t2_e1)) (and(= n_t2_e1 0)(not t2_e1)))
(or (and(> n_t5_e1 0)(t5_e1)) (and(= n_t5_e1 0)(not t5_e1)))
```

Figura 4.12: Representação da transição $t2$ e $t5$ da rede apresentada na Figura 4.11 com a possibilidade de ser disparada junto com a transição $t4$ ou $t5$.

Com esta modificação em uma expansão da instância duas ou mais transições paralelas podem disparar várias vezes, mas a alteração no modo de propagar os valores não alterados deve comportar os valores de cada transição independente, bem como a combinação entre elas. Os lugares não alterados por uma transição ou pelo conjunto de transições são diferentes e todos os possíveis conjuntos de propagação devem estar expressos em cada

transição.

O disparo de transições paralelas em uma mesma expansão diminui o número de expansões necessárias para chegar ao estado desejado, porém a busca de quais transições podem ser disparadas em conjunto e quais variáveis devem ser propagadas é muito custosa. Construir uma instância que possa tirar proveito de todo o paralelismo da rede é muito demorado e a instância resultante é grande e complexa para ser resolvida.

4.3 Considerações

Este capítulo apresenta o método desenvolvido neste trabalho bem como as melhorias no método realizadas sobre o método inicial. É apresentado um exemplo do funcionamento do método onde uma rede e a representação das propriedades são apresentada em instâncias SMT. Este exemplo mostra como o método gera a instância que posteriormente deve ser submetida ao resolvedor SMT.

O Capítulo 5 apresenta os programas implementados de acordo com o método apresentado, a descrição dos testes realizados e as instâncias utilizadas nestes testes.

CAPÍTULO 5

EXPERIMENTOS

Para validação do método apresentado neste trabalho foram realizados testes com alguns conjuntos de redes de Petri extraídas da modelagem de alguns sistemas biológicos e redes geradas para verificação de existência de bloqueios. Uma descrição sobre as redes utilizadas e os programas testados são apresentados nas próximas subseções.

5.1 Conjunto de Redes Utilizadas

Para a realização dos testes utilizou-se um conjunto de 100 redes descritas no Apêndice A. O Apêndice B apresenta o número de transições, lugares e arcos que compõem cada rede utilizada. Algumas redes utilizadas são listadas nesta seção. As redes do número 1 ao 37 são redes de sistemas biológicos, exportadas utilizando o GINsim [46] esta ferramenta para modelagem e simulação de redes reguladoras genéticas. O GINsim disponibiliza várias redes reguladoras que podem ser facilmente exportadas para um formato de rede de Petri. Estas redes tem como característica serem k-limitadas e terem um grande número de arcos, que geram muitos laços elementares.

Da lista, as redes de 38 a 44 são redes apresentadas por Keijo Heljanko [22] para a verificação de existência de bloqueio. Estas redes são 1-limitadas, com número de arcos equilibrado. A lista de redes apresentada contém o nome do problema a partir do qual a rede foi modelada e uma pequena descrição do problema.

1. *Core engine*: Modelagem lógica do mecanismo principal para controle do ciclo celular do brotamento de levedura.

2. *MCP model*: Modelagem lógica do ponto de verificação morfogenética do brotamento da levedura.

3. *Exit model*: Modelagem lógica do módulo de saída do brotamento de levedura.

4. *Coupled model*: Modelo lógico combinando o controle do ciclo celular do brotamento de levedura com a verificação morfogenética e um módulo de saída detalhado.

5. *Mammalian cell cycle*: Modelo genérico booleano para o controle de pontos de restrição do ciclo celular de mamífero.

6. *ErbB2 model*: Definição e simulação do modelo lógico para o controle de ativação da rede ErbB2.

7. *Drosophila cell cycle*: Ciclo celular da Drosophila.

8. *Direct transcription Orlando 2008*: Transcrição direta do modelo booleano do ciclo celular do brotamento da levedura publicado por Orlando et al. 2008.

9. *Direct transcription Irons 2009*: Transcrição direta do modelo booleano do ciclo celular do brotamento da levedura publicado por Irons 2009.

10. *Direct transcription Davidich e Bornholdt 2008*: Transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008.

11. *Adaptation Davidich e Bornholdt 2008*: Adaptação da transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008.

12. *Direct transcription Li et al. 2004*: Transcrição direta do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004.

13. *Adaptation Li et al. 2004*: Adaptação da transcrição do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004.

14. *Gap model A*: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região A do embrião.

15. *Gap model B*: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região B do embrião.

16. *Gap model C*: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região C do embrião.

17. *Gap model D*: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região D do embrião.

18. *Pair Rule*: Análise dinamica do par de regra do módulo de controle cross-regulatory na segmentação do Drosophila melanogaster.

19. *Segment-Polarity model 1 cell*: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Grafo regulamentar de uma celula encapsulada.

20. *Segment-Polarity model 6 cells*: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Modelo com seis células.

21. *Detailed Segment-Polarity model 1*: Modelo celular estendido para considerar 18 componentes.

22. *Detailed Segment-Polarity model 2*: Modelo multi-celular abrangendo duas celulas.

23. *Reduced Segment-Polarity model 2*: Versão reduzida de um modelo multi-celular abrangendo duas celulas.

24. *Ap Boundary*: Modelo lógico de valores múltiplos do processo de padronização que define o limite anterior-posterior celular.

25. *Apterous dependent network*: Modelo booleano envolvendo 10 variáveis booleanas. Rede ápteras dependente.

26. *Dorsal-ventral boundary wing imaginal disc*: Formação do limite dorso-ventral do disco imaginal da asa de Drosophila.

27. *Th 17*: Modelo qualitativo do controle da diferenciação de células Th1/Th2 com 17 componentes reguladores.

28. *Th Boolean*: Modelo booleano simplificado do controle da diferenciação de células Th.

29. *TCR Signalisation Pathway*: Modelo booleano da via de sinalização do TCR.

30. *Th differentiation full*: Modelo completo e abrangente da rede regulatória e vias de sinalização que controlam a diferenciação das células Th.

31. *Th differentiation reduced*: Modelo reduzido da rede regulatória e vias de sinalização que controlam a diferenciação das células Th.

32. *Trp Regulatory*: Modelagem qualitativa da rede controlando a biossíntese de Trp.

33. *Controlling the lysis-lysogeny decision - 2 genes*: Rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CI e Cro.

34. *Controlling the lysis-lysogeny decision - 4 genes*: Rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CII, N, CI e Cro.

35. *Controlling the lysis-lysogeny decision - 4 genes (modified)*: Versão modificada da rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CII, N, CI e Cro.

36. *Flowering of arabidopsis*: Modelo para controle da morfogênese da flor na planta modelo Arabidopsis thaliana.

37. *Neuron differentiation in Zebra Fish*: Este modelo booleano mostra o papel sutil de miR-9 no curso de especificação neural no peixe-zebra.

38. DP: *Dining Philosophers* apesar de não ser um problema muito realístico, contém um bloqueio não trivial.

39. *ELEVATOR*: Modela o controle para uma construção com elevador, usando tarefas para modelar os mesmos.

40. HARTSTONE: *Hartstone Program* é o esqueleto de comunicação de um programa em Ada, onde cada tarefa que começa pára tarefas em progresso.

41. MMGT: *Distributed Memory Manager* é um esqueleto de comunicação de um programa Ada que implementa um esquema de gerenciamento de memória com usuários.

42. Q: User Interface é o esqueleto Ada de uma interface de usuário *RPC client/server based* com 18 tarefas que são usadas por diversas aplicações reais.

43. SENTEST: Sensor *Test Program* é o esqueleto de comunicação de um programa Ada que inicia tarefas para testar sensores.

44. SPEED: *Speed Regulation Program* é o esqueleto de comunicação de um programa Ada com 10 tarefas que monitoram e regulam a velocidade de um carro.

Todas as redes utilizadas precisaram ser convertidas para os formatos utilizados pelos programas. Os programas implementados utilizam o formato "net"padrão da ferramenta TINA [12], uma *toobox* para edição e análise de redes de Petri. Este formato foi escolhido pela sua facilidade de entendimento e por ter vários programas que convertem redes para este formato. O sistema Lola utiliza um formato "net"que contém regras específicas. Para a conversão utilizou-se o programa Snoopy [10], capaz de converter as redes para os dois formatos.

Os programas implementados recebem a rede no formato "net"utilizado pela *toobox* TINA e aplicando as regras apresentadas no Capítulo 4 a transforma em uma instância no padrão SMT-Lib [43], que é o formato utilizado nas competições, como a SMTEXEC [42]. Os detalhes de implementações de cada programa é apresentado na seção 5.2.

5.2 Programas Implementados

O programa implementado neste trabalho foi nomeado "net2smt"por se tratar de um programa que faz a conversão das propriedades de redes de Petri para instâncias SMT, foram implementados quatro versões que serão apresentadas nesta seção.

A primeira versão foi implementada em linguagem de programação C e denominada "net2smt_v1". Esta versão implementa o método apresentado na seção 4.1, que posteriormente é utilizado como base para as novas melhorias. Este programa é capaz de ler uma rede no formato "net"e converter os problemas de alcançabilidade e bloqueio para uma instância SMT. No caso de alcançabilidade, um arquivo extra também é passado para o

conversor, que descreve o estado que se deseja alcançar. Dois módulos extras foram inseridos neste programa, um para gerar a instância em ISCAS ,um formato utilizado para representar fórmulas não clausais, e a inserção da biblioteca ABC [45] para conversão da instância ISCAS para uma instância em CNF. Estes dois módulos não foram implementados nas novas versões do programa, uma vez que apresentaram tempos maiores que SMT, como pode ser visto nas Tabelas 5.4 e 5.5.

A segunda versão foi implementada em C++, devido ao fato das bibliotecas de programação do resolvedor SMT utilizado terem somente a versão em C++ atualizada. Esta versão do programa foi denominada "net2smt_v2.0"e implementa o método apresentado com as melhorias propostas na subseção 4.2.1 e na subseção 4.2.2. Estas melhorias apresentaram menor tempo gasto para resolução, comparado com a versão anterior. As melhorias inseridas no "net2smt_v2.0"estarão presentes nas próximas versões do programa implementado.

Uma nova versão foi implementada e denominada "net2smt_v2.1", incorporando o método apresentado na subseção 4.2.3 que permitiu ao resolvedor disparar uma transição mais de uma vez a cada expansão. Estas alterações são necessárias para que a próxima versão do programa funcione.

A ultima versão implementada é denominada "net2smt_v2.2"e implementa as funções para disparar mais de uma transcrição em paralelo, descritas na subseção 4.2.4. Este método não pode ser executado por completo, pois a verificação de todos os conjuntos paralelos de transições é muito custoso e a implementação corta a busca para conjuntos de no máximo três transições paralelas. Além da necessidade de gerar os conjunto de transições paralelas, esse programa tem que gerar os lugares que devem ser propagados para cada conjunto de transições paralelas. Na seção 5.3 são apresentados os resultados para os programas apresentados e para o sistema Lola. É apresentado ainda uma análise destes resultados.

5.3 Testes

Foi realizada uma rodada de testes preliminares, afim de escolher um resolvedor com tempos de resolução baixos e capacidade de resolver as instâncias geradas pelos programas apresentados na seção 5.2. Para isto escolhemos os resolvedores que tiveram os menores tempos de resolução na competição SMT-COMP [3]. Os resolvedores testados foram o Z3 [35], CVC4 [5] e Matsat5 [18]. Os resolvedores CVC4 e Matsat5 foram escolhidos por apresentarem um baixo tempo de resolução na competição de 2012 e o Z3 foi escolhido por ter um tempo de resolução baixo na competição de 2011. Os resolvedores foram testados utilizando o net2smt_v1.

Para fazer a análise de qual resolvedor mais se adaptaria às instâncias geradas foram utilizadas algumas redes com bloqueio. Essas redes foram convertidas para instâncias e submetidas a cada um dos resolvedores com suas opções padrões. O tempo que cada um dos resolvedores demorou para encontrar o bloqueio pode ser visto na Tabela 5.1.

Os testes com os resolvedores foram executados em uma máquina com processador Intel Centrino Duo de 2.1 GHZ com 3 GB de memória, com sistema operacional Linux distribuição Ubuntu, versão 12.04.

Tabela 5.1: Comparação de tempo entre os resolvedores SMT para verificação de bloqueio. Tempo expresso em segundos.

Problema	Z3	Matsat5	CVC4
DP(6)	0.948	3.654	15.348
DP(8)	11.575	115.522	135.419
SENT(25)	0.436	1.910	8.667
SENT(50)	0.826	2.644	19.134
SENT(75)	1.388	4.096	37.283
SPD(1)	0.351	0.689	5.128

Após a análise preliminar, optou-se pelo o uso do resolvedor Z3, que obteve tempos menores no conjunto de testes. Todos os resultados apresentados nas Tabelas 5.2, 5.3, 5.4 e 5.5 utilizam este resolvedor.

A Tabela 5.2 apresenta os tempos de execução de algumas redes utilizadas para as diferentes implementações do método desenvolvido. Os resultados de tempo para todas as redes utilizadas para testes de alcançabilidade estão disponíveis no Apêndice C. Este

tempo compreende o tempo de leitura e conversão, além do tempo para resolução. Os tempos expressos estão em segundos. Os "N"representam que não foi possível chegar a um resultado, pois o programa excedeu o tempo limite de cinco minutos para resolver o problema. O termo "Memória"representa falha ocorrida por estouro de memória.

Tabela 5.2: Resultados dos programas testados para alcançabilidade, com tempo expresso em segundos.

Nome	**v2.0**	**v2.1**	**v2.2**	**Lola**	**v1**
Adaptation Davidich & Bornholdt 2008 Cdc2-ON	0.02	0.05	0.15	0.02	Memória
Controlling the lysis-lysogeny decision - 4 genes	0.03	0.06	0.06	0.03	0.20
Controlling the lysis-lysogeny decision - 2 genes	0.03	0.03	0.03	0.01	0.06
Gap model C	0.04	0.11	0.11	0.03	0.80
MCP model	0.03	0.06	0.10	0.01	N
MCP model Wild-type	0.05	0.16	0.54	0.01	N
MCP model swe1-delta	0.03	0.06	0.34	0.01	0.23
MCP model mih1-delta	0.03	0.06	0.10	0.01	0.27
Direct transcription Davidich & Bornholdt 2008	0.02	0.08	0.12	0.01	Memória
Adaptation Davidich & Bornholdt 2008 Cln3-ON_Ste9-ON	0.04	0.11	0.24	0.01	Memória
Gap model B	0.07	0.09	0.12	0.03	N
Gap model D	0.03	0.08	0.06	0.01	0.44
Pair Rule	0.02	0.07	0.07	0.01	0.57
Apterous dependent network	0.04	0.09	0.30	0.01	0.22
Controlling the lysis-lysogeny decision - 4 genes N_KO	0.07	0.09	0.08	0.03	1.22
HARTSTONE(25)	1.53	4.52	27.34	1.21	1.44

A Tabela 5.3 apresenta o tempo para resolução do problema de bloqueio em rede de Petri para algumas redes. Os resultados de tempo para todas as redes utilizadas para testes de bloqueio estão disponíveis no Apêndice D. Os dados são apresentados como na Tabela 5.2.

Net2smt não implementa nenhum método de redução para rede de Petri, que o faz trabalhar em um espaço de estados muito maior que os programas específicos para rede de Petri. Não utilizou-se um resolvedor SMT com funções especificas capazes de tirar proveito da estrutura das instâncias geradas. As instâncias SMT geradas têm uma hierarquia

Tabela 5.3: Resultados dos programas testados para bloqueio, com tempo expresso em segundos.

Nome	**v2.0**	**v2.1**	**v2.2**	**Lola**	**v1**
MCP model Wild-type	0.03	0.06	0.33	0.01	0.29
MCP model swe1-delta	0.03	0.05	0.34	0.01	0.27
MCP model mih1-delta	0.03	0.05	0.11	0.01	0.28
Direct transcription Davidich & Bornholdt 2008	0.03	0.07	0.13	0.01	Memória
Direct transcription Li et al. 2004	0.03	0.08	0.22	0.01	Memória
Adaptation Li et al. 2004	0.03	0.08	0.27	0.01	Memória
Gap model A	0.07	0.08	0.05	0.02	0.54
Gap model B	0.07	0.08	0.11	0.02	N
Gap model C	0.06	0.16	0.16	0.03	N
Gap model D	0.02	0.04	0.04	0.01	0.08
Controlling the lysis-lysogeny decision - 2 genes	0.02	0.04	0.03	0.01	0.08
Controlling the lysis-lysogeny decision - 4 genes	0.04	0.05	0.05	0.02	0.21
Controlling the lysis-lysogeny decision - 4 genes N_KO	0.04	0.06	0.04	0.02	0.17
Controlling the lysis-lysogeny decision - 4 genes (modified)	0.04	0.05	0.05	0.01	0.22
Flowering of arabidopsis	0.03	0.05	0.10	0.01	0.13
Neuron differentiation in Zebra Fish miR9_Etc	0.03	0.09	0.10	0.01	0.05
SPEED(1)	0.23	0.68	1.84	0.01	0.27

entre as variáveis, ou seja, as variáveis de uma expansão são dependentes das variáveis da expansão anterior. Esta característica pode ser utilizada para agilizar a resolução. Porém os resolvedores SMT não estão preparados para levar em conta estas características. O resolvedor Z3, utilizado neste trabalho, não permite edição para inserção destas funcionalidades.

A partir dos resultados apresentados na Tabela 5.2 e Tabela 5.3 pode-se ver que o net2smt_v2.1 obteve tempos bem próximos ao do sistema Lola, que implementa várias técnicas de redução. Pode-se ver na Tabela 5.2 que os tempos para a rede "Adaptation Davidich & Bornholdt 2008 Cdc2-ON"e "Controlling the lysis-lysogeny decision - 4 genes"tiveram tempos de resolução iguais entre o sistema Lola e o net2smt_v2.1.

O método proposto para verificação de propriedades de rede de Petri utilizando instâncias SMT consegue encontrar o resultado para a maioria dos problemas. O método apresenta

na maior parte dos casos um tempo menor quando as instâncias são mais simples, como nas duas primeiras colunas da Tabela 5.2 e Tabela 5.3, onde os programas geram instâncias menores e com um menor número de variáveis.

Para os testes com redes 1-limitadas o sistema Lola não tem configuração específica, neste caso a limitação é tratada internamente e seus resultados podem ser conferidos nas Tabelas 5.2 e 5.3. Para os testes em redes 1-limitadas as versões do net2smt geram a instâncias SMT booleanas e o Z3 é executado. A exceção é o net2smt_v1 CNF que gera uma instância em ISCAS. O formato ISCAS foi escolhido por ser mais próximo da representação de uma instância SMT booleana. A instância em ISCAS é posteriormente convertida para CNF utilizando a biblioteca ABC [45]. A instância CNF é resolvida pelo resolvedor Lingelinge [19]. As Tabelas 5.4 e 5.5 apresentam os resultados para as redes 1-limitadas.

Tabela 5.4: Resultados para alcançabilidade para redes 1-limitadas em instâncias booleanas. Tempo expresso em segundos.

Nome	**v2.1**	**v2.2**	**v2.3**	**v1**	**v1 CNF**
DP(6)	0.80	1.62	2.77	1.07	7.82
DP(8)	0.30	0.46	1.19	0.37	2.15
DP(10)	0.65	0.91	3.42	1.18	6.34
DP(12)	0.50	1.08	3.64	1.29	5.09
ELEVATOR(1)	12.00	23.15	175.99	9.31	44.11
ELEVATOR(2)	N	N	N	N	N
HARTSTONE(25)	0.91	1.46	19.04	1.55	5.71
HARTSTONE(50)	1.17	1.63	132.63	1.59	5.39
HARTSTONE(75)	2.62	3.36	N	3.55	13.31
HARTSTONE(100)	5.06	5.90	N	6.52	18.34
MMGT(3)	8.07	10.64	152.25	10.41	52.65
Q(1)	N	N	N	N	N
SENTEST(25)	52.27	70.89	N	N	N
SENTEST(50)	4.22	5.28	148.39	N	N
SENTEST(75)	63.21	86.07	N	N	N
SENTEST(100)	120.57	139.42	N	N	N
SPEED(1)	0.72	1.01	5.64	0.71	2.36

Pode-se notar que fazer a conversão da instância para CNF demanda maior tempo para resolução devido ao tempo de conversão e tamanho das instâncias resultantes serem maiores. As instâncias booleanas para as redes 1-limitadas são mais fáceis de resolver

Tabela 5.5: Resultados para bloqueio em redes 1-limitadas em instâncias booleanas. Tempo expresso em segundos.

Nome	**v2.1**	**v2.2**	**v2.3**	**v1**	**v1 CNF**
DP(6)	0.29	0.56	0.45	0.39	2.82
DP(8)	6.26	8.16	2.62	5.84	10.78
DP(10)	242.25	N	30.70	N	66.26
DP(12)	N	N	N	N	N
ELEVATOR(1)	4.63	8.77	31.14	3.96	15.03
ELEVATOR(2)	284.15	N	N	N	N
HARTSTONE(25)	N	N	N	N	N
HARTSTONE(50)	N	N	N	N	N
HARTSTONE(75)	N	N	N	N	N
HARTSTONE(100)	N	N	N	N	N
MMGT(3)	223.31	231.26	N	285.59	N
Q(1)	N	N	N	N	N
SENTEST(25)	0.26	0.34	4.44	0.27	0.58
SENTEST(50)	0.58	0.75	18.77	0.58	1.12
SENTEST(75)	0.98	1.13	52.79	1.87	2.68
SENTEST(100)	1.56	1.85	118.77	1.71	4.14
SPEED(1)	0.16	0.19	0.65	0.12	0.43

pelo fato da teoria utilizada ser mais simples que a teoria de números inteiros utilizada nos testes anteriores.

5.4 Considerações

Este capítulo apresenta os programas e as melhorias que foram implementadas em cada um dos programas. Após a descrição dos programas é feita a descrição do conjunto de redes foram utilizadas nos testes.

Os experimentos foram realizados utilizando uma grande quantidade de redes. Os resultados destes testes mostram que o método funciona para verificar as propriedades de bloqueio e alcançabilidade. O tempo de verificação das propriedades mostram que o método tem possibilidade de competir com métodos tradicionais de verificação de propriedades das redes de Petri.

CAPÍTULO 6

CONCLUSÃO

Este trabalho apresenta uma abordagem para resolver o problema de alcançabilidade e bloqueio em redes de Petri lugar/transição utilizando um método incremental de tradução da rede para instâncias SMT e posterior checagem por um resolvedor.

A transformação das propriedades das redes de Petri para instância SMT apresenta um novo método para resolver o problema de alcançabilidade e de bloqueio uma vez que não foram encontrados na literatura até o momento nenhum método de transformação de propriedades de redes de Petri em instâncias SMT. Em Redes de Petri o método e disponibiliza novas instâncias SMT com características específicas que podem ajudar na melhoria dos resolvedores existentes.

Para fim de testes, utilizou-se um conjunto de cem redes, sendo 83 redes de sistemas biológicos e 17 redes 1-limitadas. O resolvedor SMT foi escolhido considerando os melhores resolvedores participantes da SMT-COMP, ocorridas em 2011 e 2012. O Z3 foi o resolvedor que apresentou menor tempo de resolução das instâncias testadas, sendo o escolhido para este trabalho.

A modelagem apresenta algumas vantagens, visto que o resultado da resolução da instância apresenta as transições disparadas, o número de vezes que as mesmas são disparadas, bem como o todas as marcações intermediárias que levaram do estado inicial ao estado final. São apresentadas não somente a sequência de transições que devem ser disparadas para atingir o estado desejado, mas também todos os estados da rede para cada um dos disparos necessários.

É possível definir um estado final onde um lugar contenha um número de marcas dentro de um intervalo definido e não somente com número de marcas igual ao solicitado, como apresentado nos programas de rede de Petri. Esta funcionalidade é implementada substituindo o símbolo de "="pelos símbolos de "<"ou ">"no estado final da instância SMT, assim como é feito para encontrar bloqueio. Também é possível fazer busca por

estados finais onde somente alguns lugares são importantes, omitindo do estado final as variáveis que representam os lugares aos quais o valor não interfere para o resultado, deixando assim para o resolvedor decidir o valor destas variáveis. Também é possível definir que uma transição tenha que obrigatoriamente ser disparada para chegar a determinada marcação final, inserindo uma igualdade que force a variável que representa a transição ter um resultado esperado, como por exemplo, $T1\_e1 = 2$, que força a transição $T1$ disparar duas vezes na expansão um da instância.

A desvantagem do net2smt é o tempo necessário para a resolução que em somente dois dos cem casos teve tempo igual ao do sistema Lola. Esta diferença de tempo pode ser causada pelo fato do sistema Lola utilizar vários métodos de simplificação da rede e algoritmos específicos para tratar redes de Petri. Também pelo fato de resolver o problema diretamente, sendo desnecessária qualquer conversão. No entanto o método proposto tem que converter a rede para uma instância SMT, ocasionando em uma perda de tempo.

O resolvedor SMT utilizado não considera as características específicas do tipo de instância resultante da transformação da rede de Petri, talvez por isso tenha gasto maior tempo para resolução comparado ao sistema Lola. O método apresentou melhores tempos em redes com maior densidade de arcos, possivelmente pelo aumento de estados alcançáveis a cada disparo.

Para trabalhos futuros, é interessante o uso de um resolvedor capaz de resolver instâncias contendo variáveis com hierarquia, ou modificar um resolvedor para que este possa tirar proveito desta característica. Outra alteração que pode apresentar um bom desempenho é a inserção de uma lógica no resolvedor SMT que defina um tipo de inteiro sem sinal, pois o método utiliza várias verificações para evitar que as variáveis inteiras assumam valores negativos, o que ocasiona um crescimento da instância que pode ser desnecessário, além do resolvedor trabalhar com um espaço de estados com o dobro do tamanho necessário.

Sugere-se ainda, aplicar as simplificações nas redes de Petri aplicáveis em conjunto com a conversão para a instâncias SMT. Fazer com que a conversão da rede gere instâncias mais simples, com menos variáveis, considerando que as simplificações na representação das propriedades realizadas durante o trabalho tiveram bom resultado. Por fim, utilizar a equação fundamental para determinar qual a quantidade mínima de expansões necessárias

para atingir a marcação desejada é sugerida, isto eliminaria a necessidade de verificar as instâncias com menos expansões que o mínimo necessário para atingir a marcação desejada.

APÊNDICE A

LISTA COMPLETA DE REDES DE PETRI

A lista completa de redes utilizadas apresentando o nome do problema a partir do qual a rede foi modelada e uma pequena descrição do problema, alguns destas redes são baseadas no mesmo problema mas com a utilização de mutações diferentes.

1. Core engine: Modelagem lógica do mecanismo principal para controle do ciclo celular do brotamento de levedura.

2. Core engine cln1_delta_cln2_delta: Modelagem lógica do mecanismo principal para controle do ciclo celular do brotamento de levedura com a mutação cln1_delta_cln2_delta.

3. Core engine cln1_delta_cln2_delta_sic1_delta: Modelagem lógica do mecanismo principal para controle do ciclo celular do brotamento de levedura com a mutação cln1_delta_cln2_delta_sic1_delta.

4. Core engine cln1_delta_cln2_delta_cdh1_delta: Modelagem lógica do mecanismo principal para controle do ciclo celular do brotamento de levedura com a mutação cln1_delta_cln2_delta_cdh1_delta.

5. Core engine cln1_delta_cln2_delta_bck2_delta: Modelagem lógica do mecanismo principal para controle do ciclo celular do brotamento de levedura com a mutação cln1_delta_cln2_delta_bck2_delta.

6. MCP model: Modelagem lógica do ponto de verificação morfogenética do brotamento da levedura.

7. MCP model Wild-type: Modelagem lógica do ponto de verificação morfogenética do brotamento da levedura com a mutação Wild-type.

8. MCP model swe1-delta: Modelagem lógica do ponto de verificação morfogenética do brotamento da levedura com a mutação swe1-delta.

9. MCP model mih1-delta: Modelagem lógica do ponto de verificação morfogenética do brotamento da levedura com a mutação mih1-delta.

10. Exit model: Modelagem lógica do módulo de saída do brotamento de levedura.

11. Exit model MEN_inactive: Modelagem lógica do módulo de saída do brotamento de levedura com a mutação MEN_inactive.

12. Exit model separase_inactivation: Modelagem lógica do módulo de saída do brotamento de levedura com a mutação separase_inactivation.

13. Exit model bub2_delta: Modelagem lógica do módulo de saída do brotamento de levedura com a mutação bub2_delta.

14. Coupled model: Modelo lógico combinando o controle do ciclo celular do brotamento de levedura com a verificação morfogenética e um módulo de saída detalhado.

15. Coupled model cln1_delta_cln2_delta: Modelo lógico combinando o controle do ciclo celular do brotamento de levedura com a verificação morfogenética e um módulo de saída detalhado com a mutação cln1_delta_cln2_delta.

16. Coupled model cln1_delta_cln2_delta_sic1_delta: Modelo lógico combinando o controle do ciclo celular do brotamento de levedura com a verificação morfogenética e um módulo de saída detalhado com a mutação cln1_delta_cln2_delta_sic1_delta.

17. Coupled model cln1_delta_cln2_delta_cdh1_delta: Modelo lógico combinando o controle do ciclo celular do brotamento de levedura com a verificação morfogenética e um módulo de saída detalhado com a mutação cln1_delta_cln2_delta_cdh1_delta.

18. Mammalian cell cycle: Modelo genérico booleano para o controle de pontos de restrição do ciclo celular de mamífero.

19. ErbB2 model: Definição e simulação do modelo lógico para o controle de ativação da rede ErbB2.

20. ErbB2 model AKT1: Definição e simulação do modelo lógico para o controle de ativação da rede ErbB2 com a mutação AKT1.

21. ErbB2 model MEK1: Definição e simulação do modelo lógico para o controle de ativação da rede ErbB2 com a mutação MEK1.

22. ErbB2 model CDK2: Definição e simulação do modelo lógico para o controle de ativação da rede ErbB2 com a mutação CDK2.

23. Drosophila cell cycle: Ciclo celular da Drosophila.

24. Drosophila cell cycle mutant_1: Ciclo celular da Drosophila com a mutação mutant_1.

25. Drosophila cell cycle CycD-ON: Ciclo celular da Drosophila com a mutação CycD-ON.

26. Drosophila cell cycle CycD-ON_Fzr-ON: Ciclo celular da Drosophila com a mutação CycD-ON_Fzr-ON.

27. Direct transcription Orlando 2008: Transcrição direta do modelo booleano do ciclo celular do brotamento da levedura publicado por Orlando et al. 2008.

28. Direct transcription Irons 2009: Transcrição direta do modelo booleano do ciclo celular do brotamento da levedura publicado por Irons 2009.

29. Direct transcription Irons 2009 Cln3_KO: Transcrição direta do modelo booleano do ciclo celular do brotamento da levedura publicado por Irons 2009 com a mutação Cln3_KO

30. Direct transcription Irons 2009 SMBF_KO: Transcrição direta do modelo booleano do ciclo celular do brotamento da levedura publicado por Irons 2009 com a mutação SMBF_KO.

31. Direct transcription Irons 2009 Cln2_KO: Transcrição direta do modelo booleano do ciclo celular do brotamento da levedura publicado por Irons 2009 com a mutação Cln2_KO.

32. Direct transcription Davidich e Bornholdt 2008: Transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008.

33. Direct transcription Davidich e Bornholdt 2008 SK-ON: Transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008 com a mutação SK-ON.

34. Adaptation Davidich e Bornholdt 2008: Adaptação da transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008.

35. Adaptation Davidich e Bornholdt 2008 Cln3-ON: Adaptação da transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008 com a mutação Cln3-ON.

36. Adaptation Davidich e Bornholdt 2008 Cdc2-ON Adaptação da transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008 com a mutação Cdc2-ON.

37. Adaptation Davidich e Bornholdt 2008 Cln3-ON_Ste9-ON: Adaptação da transcrição do ciclo celular de levedura de fissão publicado por Davidich e Bornholdt 2008 com a mutação Cln3-ON_Ste9-ON.

38. Direct transcription Li et al. 2004: Transcrição direta do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004.

39. Direct transcription Li et al. 2004 mutant_1: Transcrição direta do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004 com a mutação mutant_1.

40. Adaptation Li et al. 2004: Adaptação da transcrição do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004.

41. Adaptation Li et al. 2004 Cln3-ON: Adaptação da transcrição do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004 com a mutação Cln3-ON.

42. Adaptation Li et al. 2004 Cln3-ON_Cgh1-ON: Adaptação da transcrição do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004 com a mutação Cln3-ON_Cgh1-ON.

43. Adaptation Li et al. 2004 Clb2-ON: Adaptação da transcrição do modelo do ciclo celular do brotamento da levedura publicado por Li et al. 2004 com a mutação Clb2-ON.

44. Gap model A: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região A do embrião.

45. Gap model B: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região B do embrião.

46. Gap model C: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região C do embrião.

47. Gap model D: Análise formal da rede gap-gene envolvida na segmentação Drosophila. Grafo do estado da transição da região D do embrião.

48. Pair Rule: Análise dinamica do par de regra do módulo de controle cross-regulatory na segmentação do Drosophila melanogaster.

49. Segment-Polarity model 1 cell: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Grafo regulamentar de uma celula encapsulada.

50. Segment-Polarity model 1 cell Wg_KO: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Grafo regulamentar de uma celula encapsulada com a mutação Wg_KO.

51. Segment-Polarity model 1 cell En_KO: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Grafo regulamentar de uma celula encapsulada com a mutação En_KO.

52. Segment-Polarity model 1 cell Hh_KO: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Grafo regulamentar de uma celula encapsulada com a mutação Hh_KO.

53. Segment-Polarity model 6 cells: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Modelo com seis células.

54. Segment-Polarity model 6 cells Wg_KO: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Modelo com seis células e mutação Wg_KO.

55. Segment-Polarity model 6 cells En_KO: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Modelo com seis células e mutação En_KO.

56. Segment-Polarity model 6 cells Hh_KO: Modelo lógico centrado em componentes moleculares da rede reguladora do segmento de polaridade. Modelo com seis células e mutação Hh_KO.

57. Detailed Segment-Polarity model 1: Modelo celular estendido para considerar 18 componentes.

58. Detailed Segment-Polarity model 2: Modelo multi-celular abrangendo duas celulas.

59. Reduced Segment-Polarity model 2: Versão reduzida de um modelo multi-celular abrangendo duas celulas.

60. Ap Boundary: Modelo lógico de valores múltiplos do processo de padronização que define o limite anterior-posterior celular.

61. Ap Boundary Hh4_KO: Modelo lógico de valores múltiplos do processo de padronização que define o limite anterior-posterior celular com a mutação Hh4_KO.

62. Ap Boundary Ci1_KO: Modelo lógico de valores múltiplos do processo de padronização que define o limite anterior-posterior celular com a mutação Ci1_KO.

63. Ap Boundary Ci23_KO: Modelo lógico de valores múltiplos do processo de padronização que define o limite anterior-posterior celular com a mutação Ci23_KO.

64. Apterous dependent network: Modelo booleano envolvendo 10 variáveis booleanas. Rede ápteras dependente.

65. Dorsal-ventral boundary wing imaginal disc: Formação do limite dorso-ventral do disco imaginal da asa de Drosophila.

66. Th 17: Modelo qualitativo do controle da diferenciação de células Th1/Th2 com 17 componentes reguladores.

67. Th Boolean: Modelo booleano simplificado do controle da diferenciação de células Th.

68. TCR Signalisation Pathway: Modelo booleano da via de sinalização do TCR.

69. Th differentiation full: Modelo completo e abrangente da rede regulatória e vias de sinalização que controlam a diferenciação das células Th.

70. Th differentiation reduced: Modelo reduzido da rede regulatória e vias de sinalização que controlam a diferenciação das células Th.

71. Trp Regulatory: Modelagem qualitativa da rede controlando a biossíntese de Trp.

72. Controlling the lysis-lysogeny decision - 2 genes: Rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CI e Cro.

73. Controlling the lysis-lysogeny decision - 4 genes: Rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CII, N, CI e Cro.

74. Controlling the lysis-lysogeny decision - 4 genes Cl_KO: Rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CII, N, CI e Cro com a mutação Cl_KO.

75. Controlling the lysis-lysogeny decision - 4 genes N_KO: Rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CII, N, CI e Cro com a mutação N_KO.

76. Controlling the lysis-lysogeny decision - 4 genes Cro_Etc: Rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CII, N, CI e Cro com a mutação Cro_Etc.

77. Controlling the lysis-lysogeny decision - 4 genes (modified): Versão modificada da rede regulamentar para controle da decisão entre lise e lysogenisation. Genes reguladores CII, N, CI e Cro.

78. Flowering of arabidopsis: Modelo para controle da morfogênese da flor na planta modelo Arabidopsis thaliana.

79. Neuron differentiation in Zebra Fish: Este modelo booleano mostra o papel sutil de miR-9 no curso de especificação neural no peixe-zebra.

80. Neuron differentiation in Zebra Fish miR9_KO: Este modelo booleano mostra o papel sutil de miR-9 no curso de especificação neural no peixe-zebra com a mutação miR9_KO.

81. Neuron differentiation in Zebra Fish miR9_Etc: Este modelo booleano mostra o papel sutil de miR-9 no curso de especificação neural no peixe-zebra com a mutação miR9_Etc.

82. Neuron differentiation in Zebra Fish Her6_KO: Este modelo booleano mostra o papel sutil de miR-9 no curso de especificação neural no peixe-zebra com a mutação Her6_KO.

83. Neuron differentiation in Zebra Zic5_KO: Este modelo booleano mostra o papel sutil de miR-9 no curso de especificação neural no peixe-zebra com a mutação Zic5_KO.

84. DP(6): Dining Philosophers apesar de não ser um problema muito realístico, contém um bloqueio não trivial. Esta rede conté 6 filósofos.

85. DP(8): Dining Philosophers apesar de não ser um problema muito realístico, contém um bloqueio não trivial. Esta rede conté 8 filósofos.

86. DP(10): Dining Philosophers apesar de não ser um problema muito realístico, contém um bloqueio não trivial. Esta rede conté 10 filósofos.

87. DP(12): Dining Philosophers apesar de não ser um problema muito realístico, contém um bloqueio não trivial. Esta rede conté 12 filósofos.

88. ELEVATOR(1): Modela o controle para uma construção com 1 elevador, usando tarefas para modelar os mesmos.

89. ELEVATOR(2): Modela o controle para uma construção com 2 elevadores, usando tarefas para modelar os mesmos.

90. HARTSTONE(25): Hartstone Program é o esqueleto de comunicação de um programa em Ada, onde cada tarefa que começa pára 25 tarefas em progresso.

91. HARTSTONE(50): Hartstone Program é o esqueleto de comunicação de um programa em Ada, onde cada tarefa que começa pára 50 tarefas em progresso.

92. HARTSTONE(75): Hartstone Program é o esqueleto de comunicação de um programa em Ada, onde cada tarefa que começa pára 75 tarefas em progresso.

93. HARTSTONE(100): Hartstone Program é o esqueleto de comunicação de um programa em Ada, onde cada tarefa que começa pára 100 tarefas em progresso.

94. MMGT(3): Distributed Memory Manager é um esqueleto de comunicação de um programa Ada que implementa um esquema de gerenciamento de memória com 3 usuários.

95. Q(1): User Interface é o esqueleto Ada de uma interface de usuário RPC client/server based com 18 tarefas que são usadas por diversas aplicações reais.

96. SENTEST(25): Sensor Test Program é o esqueleto de comunicação de um programa Ada que inicia 25 tarefas para testar sensores.

97. SENTEST(50): Sensor Test Program é o esqueleto de comunicação de um programa Ada que inicia 50 tarefas para testar sensores.

98. SENTEST(75): Sensor Test Program é o esqueleto de comunicação de um programa Ada que inicia 75 tarefas para testar sensores.

99. SENTEST(100): Sensor Test Program é o esqueleto de comunicação de um programa Ada que inicia 100 tarefas para testar sensores.

100. SPEED(1): Speed Regulation Program é o esqueleto de comunicação de um programa Ada com 10 tarefas que monitoram e regulam a velocidade de um carro.

APÊNDICE B

DESCRIÇÃO DO TAMANHO DAS REDES DE PETRI UTILIZADAS

A Tabela B.1 apresenta o tamanho das redes utilizadas com o seus respectivos número de lugares, transições e arcos.

Tabela B.1: Tamanho das redes utilizadas com número de lugares de transcrições e arcos.

Nome	**Autor**	**Transcrições**	**Lugares**	**Arcos**
Core engine	GINsim	757	54	12647
Core engine cln1_delta_cln2_delta	GINsim	756	54	12639
Core engine cln1_delta_cln2_delta_sic1_delta	GINsim	809	54	13365
Core engine cln1_delta_cln2_delta_cdh1_delta	GINsim	817	54	13739
Core engine cln1_delta_cln2_delta_bck2_delta	GINsim	754	54	12627
MCP model	GINsim	39	16	302
MCP model Wild-type	GINsim	39	16	302
MCP model swe1-delta	GINsim	36	16	270
MCP model mih1-delta	GINsim	36	16	276
Exit model	GINsim	65	24	576
Exit model MEN_inactive	GINsim	65	24	574
Exit model separase_inactivation	GINsim	64	24	572
Exit model bub2_delta	GINsim	65	24	574
Coupled model	GINsim	953	64	15475
Coupled model cln1_delta_cln2_delta	GINsim	952	64	15467
Coupled model cln1_delta_cln2_delta_sic1_delta	GINsim	1084	64	17733
Continua na Próxima Página				

Nome	**Autor**	**Transcrições**	**Lugares**	**Arcos**
Coupled model cln1_delta_cln2_delta_cdh1_delta	GINsim	1050	64	17255
Mammalian cell cycle	GINsim	48	20	390
ErbB2 model	GINsim	71	40	458
ErbB2 model AKT1	GINsim	70	40	444
ErbB2 model MEK1	GINsim	70	40	444
ErbB2 model CDK2	GINsim	70	40	448
Drosophila cell cycle	GINsim	68	28	540
Drosophila cell cycle mutant_1	GINsim	68	28	540
Drosophila cell cycle CycD-ON	GINsim	68	28	540
Drosophila cell cycle CycD-ON_Fzr-ON	GINsim	63	28	480
Direct transcription Orlando 2008	GINsim	33	18	210
Direct transcription Irons 2009	GINsim	85	36	636
Direct transcription Irons 2009 Cln3_KO	GINsim	84	36	630
Direct transcription Irons 2009 SMBF_KO	GINsim	83	36	618
Direct transcription Irons 2009 Cln2_KO	GINsim	84	36	630
Direct transcription Davidich & Bornholdt 2008	GINsim	35	20	252
Direct transcription Davidich & Bornholdt 2008 SK-ON	GINsim	34	20	246
Continua na Próxima Página				

Nome	**Autor**	**Transcrições**	**Lugares**	**Arcos**
Adaptation Davidich & Bornholdt 2008	GINsim	32	18	212
Adaptation Davidich & Bornholdt 2008 Cln3-ON	GINsim	31	18	206
Adaptation Davidich & Bornholdt 2008 Cdc2-ON	GINsim	29	18	172
Adaptation Davidich & Bornholdt 2008 Cln3-ON_Ste9-ON	GINsim	30	18	196
Direct transcription Li et al. 2004	GINsim	60	22	538
Direct transcription Li et al. 2004 mutant_1	GINsim	60	22	538
Adaptation Li et al. 2004	GINsim	53	22	394
Adaptation Li et al. 2004 Cln3-ON	GINsim	53	22	394
Adaptation Li et al. 2004 Cln3-ON_Cgh1-ON	GINsim	51	22	374
Adaptation Li et al. 2004 Clb2-ON	GINsim	49	22	346
Gap model A	GINsim	14	8	100
Gap model B	GINsim	15	8	104
Gap model C	GINsim	18	8	126
Gap model D	GINsim	11	8	58
Pair Rule	GINsim	34	14	248
Segment-Polarity model 1 cell	GINsim	50	28	342
Segment-Polarity model 1 cell Wg_KO	GINsim	49	28	314
Continua na Próxima Página				

Nome	**Autor**	**Transcrições**	**Lugares**	**Arcos**
Segment-Polarity model 1 cell En_KO	GINsim	49	28	334
Segment-Polarity model 1 cell Hh_KO	GINsim	50	28	336
Segment-Polarity model 6 cells	GINsim	307	144	2146
Segment-Polarity model 6 cells Wg_KO	GINsim	295	144	1966
Segment-Polarity model 6 cells En_KO	GINsim	301	144	2098
Segment-Polarity model 6 cells Hh_KO	GINsim	301	144	2098
Detailed Segment-Polarity model 1	GINsim	67	36	444
Detailed Segment-Polarity model 2	GINsim	150	72	1056
Reduced Segment-Polarity model 2	GINsim	106	36	844
Ap Boundary	GINsim	232	56	2076
Ap Boundary Hh4_KO	GINsim	230	56	2046
Ap Boundary Ci1_KO	GINsim	227	56	2032
Ap Boundary Ci23_KO	GINsim	222	56	1988
Apterous dependent network	GINsim	26	20	136
Dorsal-ventral boundary wing imaginal disc	GINsim	186	56	1784
Th 17	GINsim	58	34	349
Th Boolean	GINsim	32	24	164
Continua na Próxima Página				

Nome	**Autor**	**Transcrições**	**Lugares**	**Arcos**
TCR Signalisation Pathway	GINsim	95	80	490
Th differentiation full	GINsim	247	130	1946
Th differentiation reduced	GINsim	146	68	1270
Trp Regulatory	GINsim	11	8	62
Controlling the lysis-lysogeny decision - 2 genes	GINsim	5	4	21
Controlling the lysis-lysogeny decision - 4 genes	GINsim	13	8	71
Controlling the lysis-lysogeny decision - 4 genes Cl_KO	GINsim	14	8	75
Controlling the lysis-lysogeny decision - 4 genes N_KO	GINsim	12	8	63
Controlling the lysis-lysogeny decision - 4 genes Cro_Etc	GINsim	16	8	90
Controlling the lysis-lysogeny decision - 4 genes (modified)	GINsim	12	8	63
Flowering of arabidopsis	GINsim	18	20	80
Neuron differentiation in Zebra Fish	GINsim	17	12	88
Neuron differentiation in Zebra Fish miR9_KO	GINsim	16	12	80
Neuron differentiation in Zebra Fish miR9_Etc	GINsim	16	12	80
Neuron differentiation in Zebra Fish Her6_KO	GINsim	16	12	80
Neuron differentiation in Zebra Zic5_KO	GINsim	16	12	80
DP(6)	Heljanko, K. and Niemelä, I.	24	36	95

Continua na Próxima Página

Nome	**Autor**	**Transcrições**	**Lugares**	**Arcos**
DP(8)	Heljanko, K. and Niemelä, I.	32	48	127
DP(10)	Heljanko, K. and Niemelä, I.	40	60	159
DP(12)	Heljanko, K. and Niemelä, I.	48	72	191
ELEVATOR(1)	Heljanko, K. and Niemelä, I.	99	63	373
ELEVATOR(2)	Heljanko, K. and Niemelä, I.	299	146	1163
HARTSTONE(25)	Heljanko, K. and Niemelä, I.	77	127	255
HARTSTONE(50)	Heljanko, K. and Niemelä, I.	152	252	505
HARTSTONE(75)	Heljanko, K. and Niemelä, I.	227	377	755
HARTSTONE(100)	Heljanko, K. and Niemelä, I.	302	502	1005
MMGT(3)	Heljanko, K. and Niemelä, I.	172	122	669
Q(1)	Heljanko, K. and Niemelä, I.	194	163	731
SENTEST(25)	Heljanko, K. and Niemelä, I.	55	104	207
SENTEST(50)	Heljanko, K. and Niemelä, I.	80	179	307
SENTEST(75)	Heljanko, K. and Niemelä, I.	105	254	407
SENTEST(100)	Heljanko, K. and Niemelä, I.	130	329	507
SPEED(1)	Heljanko, K. and Niemelä, I.	39	33	137

APÊNDICE C

RESULTADOS DOS TESTES PARA ALCANÇABILIDADE

A Tabela C.1 apresenta os tempos de execução para as diferentes implementações do método desenvolvido para todas as instâncias geradas. Este tempo compreende o tempo de leitura e conversão, além do tempo para resolver cada instâncias SMT gerada. Os tempos expressos estão em segundos. Os "N"representam que não foi possível chegar a um resultado, pois o programa excedeu o tempo limite de cinco minutos para resolver o problema. O termo "Memória"representa falha ocorrida por estouro de memória. O termo "Falha"seguido do tempo representam que o sistema Lola teve problema ao gerar a sequência de transcrições que levam ao estado buscado.

Tabela C.1: Resultados dos programas testados para alcançabilidade, com tempo expresso em segundos.

Nome	**net2smt_v2.0**	**net2smt_v2.1**	**net2smt_v2.2**	**Lola**	**net2smt_v1**
Core engine	2.68	83.89	N	0.06	Memória
Core engine cln1_delta_cln2_delta	8.17	229.78	N	0.07	Memória
Core engine cln1_delta_cln2_delta_sic1_delta	0.91	19.82	228.28	0.06	Memória
Core engine cln1_delta_cln2_delta_cdh1_delta	0.40	3.60	229.95	0.07	Memória
Core engine cln1_delta_cln2_delta_bck2_delta	4.14	78.64	N	0.07	Memória
MCP model	0.03	0.06	0.10	0.01	N
MCP model Wild-type	0.05	0.16	0.54	0.01	N
MCP model swe1-delta	0.03	0.06	0.34	0.01	0.23
MCP model mih1-delta	0.03	0.06	0.10	0.01	0.27
Exit model	0.67	3.14	10.40	0.01	Memória
Exit model MEN_inactive	1.32	7.14	15.94	0.01	Memória
Exit model separase_inactivation	0.54	2.29	5.30	0.01	Memória
Exit model bub2_delta	0.07	0.26	0.39	0.01	Memória
Coupled model	1.24	28.70	N	0.09	Memória
Coupled model cln1_delta_cln2_delta	1.20	28.15	N	0.09	Memória
Coupled model cln1_delta_cln2_delta_sic1_delta	1.46	97.51	N	0.12	Memória
Continua na Próxima Página					

Nome	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	Lola	net2smt_v1
Coupled model cln1_delta_cln2_delta_cdh1_delta	2.82	34.79	N	0.11	Memória
Mammalian cell cycle	0.10	0.38	0.94	0.01	N
ErbB2 model	0.19	0.74	3.23	0.01	Memória
ErbB2 model AKT1	0.31	1.36	9.08	0.01	Memória
ErbB2 model MEK1	0.50	2.04	9.67	0.01	Memória
ErbB2 model CDK2	0.21	0.74	5.93	0.01	Memória
Drosophila cell cycle	0.04	0.07	0.65	0.01	2.75
Drosophila cell cycle mutant_1	0.08	0.29	1.43	0.01	N
Drosophila cell cycle CycD-ON	1.07	6.29	14.57	0.01	N
Drosophila cell cycle CycD-ON_Fzr-ON	0.58	3.20	5.84	0.01	N
Direct transcription Orlando 2008	0.15	0.64	2.32	0.01	249.98
Direct transcription Irons 2009	0.18	0.92	3.30	0.01	52.97
Direct transcription Irons 2009 Cln3_KO	N	N	N	0.01	N
Direct transcription Irons 2009 SMBF_KO	0.09	0.39	3.26	0.01	12.72
Direct transcription Irons 2009 Cln2_KO	0.18	0.93	6.26	0.01	104.34
Direct transcription Davidich & Bornholdt 2008	0.02	0.08	0.12	0.01	Memória
Direct transcription Davidich & Bornholdt 2008 SK-ON	0.76	3.83	8.14	0.01	Memória

Continua na Próxima Página

Nome	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	Lola	net2smt_v1
Adaptation Davidich & Bornholdt 2008	0.22	0.85	1.66	0.01	Memória
Adaptation Davidich & Bornholdt 2008 Cln3-ON	0.35	1.51	2.69	0.01	Memória
Adaptation Davidich & Bornholdt 2008 Cdc2-ON	0.02	0.05	0.15	0.02	Memória
Adaptation Davidich & Bornholdt 2008 Cln3-ON_Ste9-ON	0.04	0.11	0.24	0.01	Memória
Direct transcription Li et al. 2004	0.17	0.93	1.16	0.01	Memória
Direct transcription Li et al. 2004 mutant_1	N	N	N	N	Memória
Adaptation Li et al. 2004	1.50	6.84	20.52	0.01	Memória
Adaptation Li et al. 2004 Cln3-ON	0.10	0.42	1.41	0.01	Memória
Adaptation Li et al. 2004 Cln3-ON_Cgh1-ON	61.06	272.99	103.36	0.01	Memória
Adaptation Li et al. 2004 Clb2-ON	0.17	0.72	2.64	0.01	Memória
Gap model A	0.09	0.14	0.13	0.02	40.60
Gap model B	0.07	0.09	0.12	0.03	N
Gap model C	0.04	0.11	0.11	0.03	0.80
Gap model D	0.03	0.08	0.06	0.01	0.44
Pair Rule	0.02	0.07	0.07	0.01	0.57
Segment-Polarity model 1 cell	0.07	0.22	1.10	0.01	2.31
Segment-Polarity model 1 cell Wg_KO	0.20	0.73	3.16	0.01	85.79
Continua na Próxima Página					

Nome	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	Lola	net2smt_v1
Segment-Polarity model 1 cell En_KO	0.11	0.41	2.08	0.01	N
Segment-Polarity model 1 cell Hh_KO	0.18	0.84	3.57	0.01	163.26
Segment-Polarity model 6 cells	0.29	0.84	N	0.01	92.75
Segment-Polarity model 6 cells Wg_KO	0.85	3.87	302.53	0.01	N
Segment-Polarity model 6 cells En_KO	0.29	0.77	331.08	0.01	51.82
Segment-Polarity model 6 cells Hh_KO	0.29	0.81	316.87	0.01	40.39
Detailed Segment-Polarity model 1	0.18	0.72	2.66	0.01	44.37
Detailed Segment-Polarity model 2	0.10	0.30	19.71	0.01	Memória
Reduced Segment-Polarity model 2	0.12	0.56	5.03	0.01	Memória
Ap Boundary	0.11	0.41	42.37	0.01	11.41
Ap Boundary Hh4_KO	0.11	0.40	42.27	0.01	N
Ap Boundary Ci1_KO	0.11	0.41	39.09	0.01	N
Ap Boundary Ci23_KO	0.11	0.38	37.23	0.01	N
Apterous dependent network	0.04	0.09	0.30	0.01	0.22
Dorsal-ventral boundary wing imaginal disc	N	N	N	0.01	N
Th 17	N	N	N	0.01	N
Th Boolean	N	N	N	0.01	N
Continua na Próxima Página					

Nome	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	Lola	net2smt_v1
TCR Signalisation Pathway	0.07	0.17	7.35	0.01	Memória
Th differentiation full	0.68	2.80	142.92	0.01	N
Th differentiation reduced	N	N	N	0.01	N
Trp Regulatory	0.05	0.11	0.63	0.01	N
Controlling the lysis-lysogeny decision - 2 genes	0.03	0.03	0.03	0.01	0.06
Controlling the lysis-lysogeny decision - 4 genes	0.03	0.06	0.06	0.03	0.20
Controlling the lysis-lysogeny decision - 4 genes Cl_KO	0.39	0.07	0.06	0.01	3.30
Controlling the lysis-lysogeny decision - 4 genes N_KO	0.07	0.09	0.08	0.03	1.22
Controlling the lysis-lysogeny decision - 4 genes Cro_Etc	0.10	0.17	0.15	0.01	39.80
Controlling the lysis-lysogeny decision - 4 genes (modified)	0.06	0.08	0.07	0.01	1.03
Flowering of arabidopsis	N	N	N	Falha 0.01	N
Neuron differentiation in Zebra Fish	N	N	N	Falha 0.01	N
Neuron differentiation in Zebra Fish miR9_KO	0.13	0.30	0.26	0.01	96.17
Neuron differentiation in Zebra Fish miR9_Etc	0.06	0.19	0.22	0.01	3.07
Neuron differentiation in Zebra Fish Her6_KO	N	N	N	Falha 0.01	N
Neuron differentiation in Zebra Zic5_KO	0.12	0.31	0.29	0.01	29.11
DP(6)	1.63	4.61	6.03	0.01	6.18
Continua na Próxima Página					

Nome	**net2smt_v2.0**	**net2smt_v2.1**	**net2smt_v2.2**	**Lola**	**net2smt_v1**
DP(8)	0.61	1.41	2.69	0.02	0.62
DP(10)	0.96	2.85	7.44	0.10	3.00
DP(12)	0.91	2.70	6.71	0.11	2.98
ELEVATOR(1)	14.15	103.12	N	0.01	88.52
ELEVATOR(2)	N	N	N	0.04	N
HARTSTONE(25)	1.53	4.52	27.34	1.21	1.44
HARTSTONE(50)	1.80	5.19	152.27	0.40	1.29
HARTSTONE(75)	4.09	12.03	N	0.52	2.79
HARTSTONE(100)	7.23	19.59	N	0.42	5.29
MMGT(3)	10.91	71.26	236.09	0.05	26.81
Q(1)	N	N	N	0.32	N
SENTEST(25)	73.43	260.39	N	0.01	N
SENTEST(50)	7.43	20.70	251.88	0.01	N
SENTEST(75)	121.18	N	N	0.01	N
SENTEST(100)	N	N	N	0.01	N
SPEED(1)	1.01	3.96	16.89	0.04	2.17

APÊNDICE D

RESULTADOS DOS TESTES PARA BLOQUEIO

A Tabela D.1 apresenta o tempo para resolução do problema de bloqueio para todas as redes de Petri utilizadas. Os dados são apresentados como na Tabela C.1 apresentada no Apêndice C. O único dado diferente é o termo "Sem"seguido do tempo levado para confirmar que a rede não contêm bloqueio.

Tabela D.1: Resultados dos programas testados para bloqueio, com tempo expresso em segundos.

NOME	**net2smt_v2.0**	**net2smt_v2.1**	**net2smt_v2.2**	**LOLA**	**net2smt_v1**
Core engine	N	N	N	N	Memória
Core engine cln1_delta_cln2_delta	N	N	N	N	Memória
Core engine cln1_delta_cln2_delta_sic1_delta	N	N	N	N	Memória
Core engine cln1_delta_cln2_delta_cdh1_delta	N	N	N	N	Memória
Core engine cln1_delta_cln2_delta_bck2_delta	N	N	N	N	Memória
MCP model	0.03	0.06	0.11	0.01	0.28
MCP model Wild-type	0.03	0.06	0.33	0.01	0.29
MCP model swe1-delta	0.03	0.05	0.34	0.01	0.27
MCP model mih1-delta	0.03	0.05	0.11	0.01	0.28
Exit model	8.38	24.90	29.00	0.01	Memória
Exit model MEN_inactive	0.44	2.52	5.54	0.01	Memória
Exit model separase_inactivation	0.31	1.65	2.38	0.01	Memória
Exit model bub2_delta	6.40	7.32	5.12	0.03	Memória
Coupled model	N	N	N	N	Memória
Coupled model cln1_delta_cln2_delta	N	N	N	N	Memória
Coupled model cln1_delta_cln2_delta_sic1_delta	N	N	N	N	Memória

Continua na Próxima Página

NOME	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	LOLA	net2smt_v1
Coupled model cln1_delta_cln2_delta_cdh1_delta	N	N	N	N	Memória
Mammalian cell cycle	0.11	0.40	0.99	0.01	N
ErbB2 model	N	N	107.61	0.03	Memória
ErbB2 model AKT1	138.49	N	109.85	0.02	Memória
ErbB2 model MEK1	141.35	N	116.75	0.02	Memória
ErbB2 model CDK2	N	N	64.40	0.03	Memória
Drosophila cell cycle	0.05	0.09	0.80	0.01	3.34
Drosophila cell cycle mutant_1	0.46	2.30	5.04	0.01	N
Drosophila cell cycle CycD-ON	N	N	N	Sem 0.01	N
Drosophila cell cycle CycD-ON_Fzr-ON	0.63	3.34	6.08	0.01	N
Direct transcription Orlando 2008	1.28	4.75	2.36	0.01	N
Direct transcription Irons 2009	N	N	N	Sem 6.69	N
Direct transcription Irons 2009 Cln3_KO	0.05	0.12	1.57	0.01	8.66
Direct transcription Irons 2009 SMBF_KO	0.69	3.79	9.84	0.01	N
Direct transcription Irons 2009 Cln2_KO	N	N	N	Sem 2.73	N
Direct transcription Davidich & Bornholdt 2008	0.03	0.07	0.13	0.01	Memória
Direct transcription Davidich & Bornholdt 2008 SK-ON	N	N	N	Sem 0.01	Memória

Continua na Próxima Página

NOME	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	LOLA	net2smt_v1
Adaptation Davidich & Bornholdt 2008	0.03	0.05	0.37	0.01	Memória
Adaptation Davidich & Bornholdt 2008 Cln3-ON	N	N	N	Sem 0.01	Memória
Adaptation Davidich & Bornholdt 2008 Cdc2-ON	0.28	0.61	0.59	0.01	Memória
Adaptation Davidich & Bornholdt 2008 Cln3-ON_Ste9-ON	0.08	0.24	0.27	0.01	Memória
Direct transcription Li et al. 2004	0.03	0.08	0.22	0.01	Memória
Direct transcription Li et al. 2004 mutant_1	N	N	N	N	Memória
Adaptation Li et al. 2004	0.03	0.08	0.27	0.01	Memória
Adaptation Li et al. 2004 Cln3-ON	N	N	N	Sem 0.01	Memória
Adaptation Li et al. 2004 Cln3-ON_Cgh1-ON	N	N	N	Sem 0.02	Memória
Adaptation Li et al. 2004 Clb2-ON	0.42	1.87	3.92	0.01	Memória
Gap model A	0.07	0.08	0.05	0.02	0.54
Gap model B	0.07	0.08	0.11	0.02	N
Gap model C	0.06	0.16	0.16	0.03	N
Gap model D	0.02	0.04	0.04	0.01	0.08
Pair Rule	0.06	0.15	0.15	0.01	2.94
Segment-Polarity model 1 cell	1.07	4.17	7.16	0.02	N
Segment-Polarity model 1 cell Wg_KO	0.13	0.21	0.41	0.01	2.04

Continua na Próxima Página

NOME	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	LOLA	net2smt_v1
Segment-Polarity model 1 cell En_KO	0.89	3.80	6.89	0.01	N
Segment-Polarity model 1 cell Hh_KO	1.11	4.54	8.26	0.01	N
Segment-Polarity model 6 cells	N	N	N	N	N
Segment-Polarity model 6 cells Wg_KO	N	N	N	N	N
Segment-Polarity model 6 cells En_KO	N	N	N	N	N
Segment-Polarity model 6 cells Hh_KO	N	N	N	N	N
Detailed Segment-Polarity model 1	19.08	20.78	12.38	0.07	N
Detailed Segment-Polarity model 2	N	N	N	N	Memória
Reduced Segment-Polarity model 2	0.42	2.36	9.28	0.01	Memória
Ap Boundary	N	N	N	N	N
Ap Boundary Hh4_KO	15.43	94.05	N	0.30	N
Ap Boundary Ci1_KO	N	N	N	N	N
Ap Boundary Ci23_KO	N	N	N	281.60	N
Apterous dependent network	0.23	0.67	1.84	0.01	N
Dorsal-ventral boundary wing imaginal disc	N	N	N	0.01	N
Th 17	N	N	N	0.01	N
Th Boolean	N	N	N	0.01	N
Continua na Próxima Página					

NOME	net2smt_v2.0	net2smt_v2.1	net2smt_v2.2	LOLA	net2smt_v1
TCR Signalisation Pathway	0.23	0.71	8.17	0.01	Memória
Th differentiation full	N	N	N	4.12	N
Th differentiation reduced	N	N	N	0.01	N
Trp Regulatory	N	N	N	Sem 0.01	N
Controlling the lysis-lysogeny decision - 2 genes	0.02	0.04	0.03	0.01	0.08
Controlling the lysis-lysogeny decision - 4 genes	0.04	0.05	0.05	0.02	0.21
Controlling the lysis-lysogeny decision - 4 genes Cl_KO	N	N	N	Sem 0.01	N
Controlling the lysis-lysogeny decision - 4 genes N_KO	0.04	0.06	0.04	0.02	0.17
Controlling the lysis-lysogeny decision - 4 genes Cro_Etc	N	N	N	Sem 0.01	N
Controlling the lysis-lysogeny decision - 4 genes (modified)	0.04	0.05	0.05	0.01	0.22
Flowering of arabidopsis	0.03	0.05	0.10	0.01	0.13
Neuron differentiation in Zebra Fish	N	N	N	Falha 0.01	N
Neuron differentiation in Zebra Fish miR9_KO	0.05	0.09	0.10	0.01	0.60
Neuron differentiation in Zebra Fish miR9_Etc	0.03	0.09	0.10	0.01	0.05
Neuron differentiation in Zebra Fish Her6_KO	N	N	N	Falha 0.01	N
Neuron differentiation in Zebra Zic5_KO	0.03	0.07	0.07	0.01	0.15
DP(6)	0.61	1.62	1.90	0.01	2.28

Continua na Próxima Página

NOME	**net2smt_v2.0**	**net2smt_v2.1**	**net2smt_v2.2**	**LOLA**	**net2smt_v1**
DP(8)	10.68	22.03	6.62	0.04	36.48
DP(10)	N	N	70.35	0.43	N
DP(12)	N	N	N	5.81	N
ELEVATOR(1)	9.39	47.46	68.51	0.01	33.52
ELEVATOR(2)	211.66	N	N	0.02	N
HARTSTONE(25)	N	N	N	N	N
HARTSTONE(50)	N	N	N	N	N
HARTSTONE(75)	N	N	N	N	N
HARTSTONE(100)	N	N	N	N	N
MMGT(3)	199.07	N	N	0.08	N
Q(1)	N	N	N	0.07	N
SENTEST(25)	0.41	1.01	7.38	0.01	0.36
SENTEST(50)	0.94	2.22	24.89	0.01	0.84
SENTEST(75)	1.62	3.92	64.86	0.01	1.48
SENTEST(100)	2.61	5.97	143.16	0.01	2.34
SPEED(1)	0.23	0.68	1.84	0.01	0.27

BIBLIOGRAFIA

[1] Cryptominisat. `http://www.msoos.org/cryptominisat2/`, 2010.

[2] Sat-race. `http://baldur.iti.uka.de`, 2010.

[3] Satisfiability modulo theories competition smt-comp. `http://smtcomp.sourceforge.net/2012/index.shtml`, 2012.

[4] Alessandro Armando, Claudio Castellini, e Enrico Giunchiglia. Sat-based procedures for temporal reasoning. *Proceedings of the 5th European Conference on Planning: Recent Advances in AI Planning*, ECP '99, páginas 97–108, London, UK, UK, 2000. Springer-Verlag.

[5] Kshitij Bansal, Clark Barrett, François Bobot, Christopher Conway, Morgan Deters, Yeting Ge, Liana Hadarean, Dejan Jovanovic̀, Timothy King, Andrew Reynolds, e Cesare Tinelli. Cvc4. `http://www.cs.nyu.edu/acsys/cvc4/`.

[6] R. S. Boyer e J. S. Moore. Machine intelligence 11. J. E. Hayes, D. Michie, e J. Richards, editors, *Machine intelligence 11*, capítulo Integrating decision procedures into heuristic theorem provers: a case study of linear arithmetic, páginas 83–124. Oxford University Press, Inc., New York, NY, USA, 1988.

[7] Roberto Bruttomesso. Satisfiability modulo theories lezione 4 - the lazy approach, 2011.

[8] Robert Cardoso, Janette Valette. *REDES DE PETRI*. UFSC, 1997.

[9] David R. Cok. *The SMT-LIBv2 Language and Tools: A Tutorial*. SMT-LIB, 2011.

[10] computer science department brandenburg university of technology cottbus. Snoopy. `http://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Snoopy/`.

[11] Stephen A. Cook. The complexity of theorem-proving procedures. *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, páginas 151–158, New York, NY, USA, 1971. ACM.

[12] Laboratoire d´Analyse et d´Architecture des Systèmes (LAAS). Time petri net analyzer. `http://projects.laas.fr/tina//`.

[13] Martin Davis, George Logemann, e Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, julho de 1962.

[14] Martin Davis e Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, julho de 1960.

[15] Leonardo de Moura e Nikolaj Bjorner. Z3: An efficient smt solver. C. Ramakrishnan e Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, páginas 337–340. Springer Berlin / Heidelberg, 2008.

[16] Leonardo de Moura e Nikolaj Bjorner. Satisfiability modulo theories: introduction and applications. *ACM Communications*, 54(9):69–77, setembro de 2011.

[17] Javier Esparza, Stefan Römer, e Walter Vogler. An improvement of mcmillan s unfolding algorithm. *Formal Methods in System Design*, páginas 87–106. Springer-Verlag, 1996.

[18] FBK-IRST e DISI-University of Trento. Mathsat 5 an smt solver for formal verification. `http://mathsat.fbk.eu/`.

[19] Institute for Formal Models e Verification. Lingeling. `http://fmv.jku.at/lingeling/`.

[20] John Franco e John Martin. *HANDBOOK OF SATISFIABILITY Frontiers in Artificial Intelligence and Applications*. IOS Press., 2009.

[21] Fausto Giunchiglia e Roberto Sebastiani. Building decision procedures for modal logics from propositional decision procedures – the case study of modal k(m), 1996.

[22] Keijo Heljanko. Mcsmodels is a deadlock and reachability checker. `http://users.ics.aalto.fi/kepa/tools/`.

[23] Joao P. Marques-silva e Karem A. Sakallah. Grasp: a search algorithm for propositional satisfiability. *Computers, IEEE Transactions on*, 48(5):506 –521, may de 1999.

[24] K. L. Mcmillan e D. K. Probst. A technique of state space search based on unfolding. *Formal Methods in System Design*, páginas 45–65, 1992.

[25] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541 –580, apr de 1989.

[26] Greg Nelson e Derek C. Oppen. Fast decision algorithms based on union and find. *Foundations of Computer Science, 1977., 18th Annual Symposium on*, páginas 114 –119, 31 1977-nov. 2 de 1977.

[27] Greg Nelson e Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.*, 1(2):245–257, outubro de 1979.

[28] Greg Nelson e Derek C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2):356–364, abril de 1980.

[29] Robert Nieuwenhuis, Albert Oliveras, e Cesare Tinelli. Abstract dpll and abstract dpll modulo theories. *In LPARâ04, LNAI 3452*, páginas 36–50. Springer, 2005.

[30] RICARDO TAVARES DE OLIVEIRA. Reduções de problemas em grafos com soluçes conexas para (max)sat e adaptação de um resolvedor sat e maxsat não clausal para as instâncias obtidas. Dissertação de Mestrado, Universidade Federal do Paraná., 2013.

[31] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[32] Carl Adam Petri. *Communication with automata*. Tese de Doutorado, Universität Hamburg, 1966.

[33] Franciele Carla Petry. Planejamento aplicado a verificação bloqueio em redes depetri. Dissertação de Mestrado, Universidade Federal do Paraná., 2008.

[34] Amir Pnueli, Yoav Rodeh, Ofer Shtrichman, e Michael Siegel. Deciding equality formulas by small domains instantiations. *Proceedings of the 11th International Conference on Computer Aided Verification*, CAV '99, páginas 455–469, London, UK, UK, 1999. Springer-Verlag.

[35] Microsoft Research. Z3 - an efficient theorem prover. `http://research.microsoft.com/en-us/um/redmond/projects/z3/`.

[36] Bruno César Ribas. Satisfatibilidade não clausal restrita às variáveis de entrada. Dissertação de Mestrado, Universidade Federal do Paraná, 2011.

[37] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, 4th edition, 1998.

[38] Jorge Luis Salvi. Relacionamento temporais entre redes e petri e planejamento automÃ¡tico. Dissertação de Mestrado, Universidade Federal do ParanÃ¡, 2009.

[39] Bart Selman, Hector Levesque, e David Mitchell. A new method for solving hard satisfiability problems. *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, páginas 440–446. AAAI Press, 1992.

[40] Sanjit A. Seshia, Shuvendu K. Lahiri, e Randal E. Bryant. A hybrid sat-based decision procedure for separation logic with uninterpreted functions. *Proceedings of the 40th annual Design Automation Conference*, DAC '03, páginas 425–430, New York, NY, USA, 2003. ACM.

[41] Robert E. Shostak. Deciding combinations of theories. *J. ACM*, 31(1):1–12, janeiro de 1984.

[42] SMT-Exec. Smt-exec. `http://www.smtexec.org/`.

[43] SMT-LIB. The satisfiability modulo theories library. `http://www.smtlib.org`, 2003.

[44] Yujin Song e Jongkun Lee. Deadlock analysis of petri nets using the transitive matrix. *SICE 2002. Proceedings of the 41st SICE Annual Conference*, volume 2, páginas 689 – 694 vol.2, aug. de 2002.

[45] Berkeley Logic Synthesis e Verification Group. Abc. `http://www.eecs.berkeley.edu/~alanmi/abc/`.

[46] Karsten Wolf. Ginsim gene interaction network simulation. `http://gin.univ-mrs.fr/GINsim/`.

[47] Karsten Wolf. Lola a low level petri net analyser. `http://www.informatik.uni-rostock.de/tpp/lola/`.

ANDERSON PEREIRA DAS NEVES

MÉTODO PARA VERIFICAÇÃO DE PROPRIEDADES DE REDES DE PETRI UTILIZANDO RESOLVEDORES SMT.

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Fabiano Silva

CURITIBA

2013