COPPE UFRJ
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

SÍNTESE DE UMA NOVA POLÍTICA DE OFUSCAÇÃO QUE GARANTA DIFERENTES NÍVEIS DE PRIVACIDADE E UTILIDADE

João Manoel Costa Cardoso

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Lilian Kawakami Carvalho

Rio de Janeiro
Junho de 2022

SÍNTESE DE UMA NOVA POLÍTICA DE OFUSCAÇÃO QUE GARANTA DIFERENTES NÍVEIS DE PRIVACIDADE E UTILIDADE

João Manoel Costa Cardoso

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Lilian Kawakami Carvalho

Aprovada por: Prof. Lilian Kawakami Carvalho
Prof. João Carlos dos Santos Basílio
Prof. Patrícia Nascimento Pena

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2022

Cardoso, João Manoel Costa

Síntese de uma Nova Política de Ofuscação que Garanta Diferentes Níveis de Privacidade e Utilidade/João Manoel Costa Cardoso. – Rio de Janeiro: UFRJ/COPPE, 2022.

VIII, 48 p.: il.; 29, 7cm.

Orientador: Lilian Kawakami Carvalho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2022.

Referências Bibliográficas: p. 46 – 48.

1. Sistemas a eventos discretos. 2. Controle supervisório. 3. Ofuscação. I. Carvalho, Lilian Kawakami. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SÍNTESE DE UMA NOVA POLÍTICA DE OFUSCAÇÃO QUE GARANTA DIFERENTES NÍVEIS DE PRIVACIDADE E UTILIDADE

João Manoel Costa Cardoso

Junho/2022

Orientador: Lilian Kawakami Carvalho

Programa: Engenharia Elétrica

O mundo está cada vez mais conectado através de redes de computadores. Isso faz com que seja necessário o desenvolvimento de técnicas capazes de garantir a privacidade e a utilidade da informação transmitida. Neste trabalho consideramos o problema de ofuscação de sistemas a eventos discretos que garanta diferentes níveis tanto de privacidade quanto de utilidade da informação gerada. O sistema é modelado como um sistema de transição rotulado que apresenta estados considerados secretos. O primeiro objetivo é manter a informação gerada pelo sistema útil para que usuários legítimos possam utilizar da mesma para tomada de decisões. Ao mesmo tempo, o segundo objetivo é esconder informações críticas a respeito dos estados secretos do sistema. Porém, diferentemente das abordagens anteriores apresentadas na literatura, consideramos o caso em que é possível para o ofuscador reportar a passagem por estados secretos. Dessa forma, além de aumentar o número de sistemas capazes de serem ofuscados, esta abordagem dificulta notar a presença do ofuscador, uma vez que mantém a visão pública do sistema com o mesmo comportamento do sistema original. Para realizar a síntese do ofuscador utilizamos o controle supervisório baseado em lógica temporal, em que a especificação do supervisor é dada por uma expressão em lógica de árvore de computação.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SYNTHESIS OF A NEW OBFUSCATION POLICY THAT GUARANTEES DIFFERENT LEVELS OF PRIVACY AND UTILITY

João Manoel Costa Cardoso

June/2022

Advisor: Lilian Kawakami Carvalho

Department: Electrical Engineering

The world is increasingly connected through computer networks., which makes it necessary to develop techniques capable of guaranteeing the privacy and utility of the transmitted information. In this work we consider the problem of obfuscation of discrete event systems that guarantees different levels of both privacy and utility of the generated information. The system is modeled as a labeled transition system that has secret states. The first goal is to keep the information generated by the system useful so that legitimate users can use it for decision making. At the same time, the second goal is to hide critical information about the secret states of the system. However, unlike the previous approaches presented in the literature, we consider the case in which it is possible for the obfuscator to report the secret states. Thus, the proposed approach increases the number of systems capable of being obfuscated and, since it keeps the public view of the system with the same behavior as the original system, it makes difficult to notice the presence of the obfuscator. To perform the obfuscator synthesis we use supervisory control based on temporal logic, in which the supervisor specification is given by an expression in computation tree logic.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Muito se tem discutido, nos últimos anos, acerca do tema Industria 4.0 que remete à quarta revolução industrial e representa uma tendência atual pela utilização de tecnologias de automação e conectividade na manufatura. De acordo com [14] a Industria 4.0 busca a conectividade de maquinas, fábricas e os complexos de armazenamentos por meio de redes globais que permitirão um controle inteligente da produção e estoque através do compartilhamento de informações. Uma tendência da Industria 4.0 é a utilização de modelos computacionais para a representação e controle de sistemas físicos, os chamados *Cyber-Physical Systems* (CPS), que são sistemas capazes de unir componentes físicos e cibernéticos por meio de tecnologias como *Internet of Things* (IOT), *Big Data* e *Cloud Networks* [1]. Porém a utilização de CPS acarreta em diversos problemas de segurança e confidencialidade, já que muitas informações cruciais são compartilhadas através de redes como a internet.

A indústria 4.0 utiliza cada vez mais das tecnologias da informação para obter mais dados sobre o processo produtivo e com isso tomar decisões para melhorar a eficiência do negócio como um todo. Porém, conectar seu processo produtivo através de redes de comunicação traz à tona o problema de segurança. Esta segurança pode estar relacionada tanto com os dados do processo propriamente ditos como também com a segurança física de quem trabalha no local. Caso um agente mal intencionado invada o sistema e com isso consiga manipular o comportamento do mesmo, as consequências podem ser muito graves. A questão da segurança de dados dificulta o crescimento da indústria 4.0 pois pode limitar a quantidade de dados disponíveis para serem analisados. Dessa forma, é necessário desenvolver tecnologias que sejam capazes de garantir a privacidade da informação gerada além de manter a qualidade da mesma.

O comportamento dos sistemas em análise neste trabalho será modelado como sistema a eventos discretos (SEDs). Consideramos que cada evento gerado pelo sistema é enviado através de redes de comunicação para usuários externos, o que pode acarretar em diversos problemas relacionados à segurança da informação ge-

rada. Assim, alguns trabalhos abordam o tema de segurança contra usuários mal intencionados [2, 5, 6, 13, 19–21, 23]. Em especial, CARVALHO *et al.* [6] considera o problema de se detectar e prevenir ataques em que atuadores são acionados com o intuito de causar danos ao sistema, e LIMA *et al.* [20] aborda a detecção de intrusos em CPS para prevenir danos causados por ataques do tipo *man-in-the-middle*. Outro trabalho, ALVES *et al.* [2], considera ataque em sensores que utiliza a teoria de controle supervisório para garantir o comportamento legal do sistema, apesar da ocorrência de um conjunto de ataques conhecidos. Todos os trabalhos citados acima tentam de alguma forma mitigar o ataque ou o conjunto de ataques.

Em segurança para sistemas a eventos discretos, uma propriedade que garante a privacidade do sistema é chamada de opacidade. Um sistema é dito opaco se for impossível para um observador externo determinar que o sistema se encontra em uma situação considerada secreta com absoluta certeza. Assim, para um sistema ser opaco as sequências de eventos observadas pelos agentes externos que levam a estados considerados secretos devem ser confundidas com sequências que levam para estados considerados não secretos. Assim, existem alguns trabalhos que abordam o tema de opacidade [11, 16, 22, 25, 28] e todas as referências contidas nos artigos citados. Quando um sistema a eventos discretos não é opaco, existem técnicas para forçar a opacidade, como os trabalhos de [12, 17, 18, 26, 29], que podem modificar a saída do sistema com o intuito de manter o segredo do mesmo, ou seja, para observadores externos o sistema permanece opaco. Uma das críticas aos trabalhos de opacidade é sobre a utilidade da informação, uma vez que ao modificar a saída do sistema não existe nada que garanta um nível aceitável de utilidade na informação reportada aos usuários externos.

Uma abordagem que permite considerar tanto o caso de forçar a privacidade quanto manter a utilidade da informação é a ofuscação, uma técnica que introduz um elemento de tomada de decisão entre o sistema e seus observadores e assim pode modificar a saída do mesmo para preservar os requisitos de privacidade e utilidade da informação. A utilidade da informação gerada pelo ofuscador depende das características individuais de cada sistema a ser ofuscado. Em sistemas baseados em geolocalização, o conceito de utilidade pode estar relacionado com a distância entre a localização real do sistema e a localização reportadas pelo ofuscador para o meio externo. As observações geradas a partir dos eventos reportados pelo ofuscador devem continuar fazendo sentido para os usuários legítimos. Algumas sequências de eventos reportada pelo ofuscador podem fazer com que a utilidade da informação transmitida se perca ou até mesmo acarrete na situação em que o ofuscador não reporta a ocorrência de nenhum evento. Como não é possível diferenciar um observador legítimo de um ilegítimo, é importante desenvolver abordagens de ofuscação que sejam capazes de preservar tanto a privacidade quanto a utilidade da informação

reportada.

O tema ofuscação já foi trazido para o contexto de sistemas a eventos discretos. DUCKHAM e KULIK [9] estabelece uma estrutura de localização baseado em grafos para tratar casos de ofuscação para serviços. Essa estrutura busca equilibrar a necessidade de serviços de alta qualidade de informação com a necessidade de privacidade de localização. Para isso, é criada uma região de ofuscação, o que gera uma incerteza sobre a localização exata do sistema, mas garante que a informação gerada contínua válida. WU *et al.* [30] foi o primeiro trabalho a considerar o caso de sintetizar um ofuscador com o objetivo de garantir que a utilidade da informação entregue esteja dentro de um limite desejado. Duas abordagens para solucionar o problema de ofuscação são apresentadas. A primeira modela o sistema através de autômatos e utiliza funções de edição que são capazes de adicionar, apagar ou alterar a ocorrência de eventos para realizar a ofuscação do sistema e garantir a utilidade da informação entregue. A segunda abordagem modela o sistema como um sistema de transição rotulado e utiliza o programa SynthSMV para sintetizar o ofuscador através de especificações temporais que garantem a utilidade da informação gerada. Por sua vez, GÓES *et al.* [15], também no contexto de ofuscação, apresenta um mecanismo para forçar a privacidade e a utilidade enquanto a posição de um usuário é rastreada. O sistema é modelado através de um sistema de transição e, em seguida, também utiliza o programa SynthSMV para realizar a síntese do ofuscador. Através do resultado obtido no SynthSMV é desenvolvida uma aplicação em tempo real para realizar a ofuscação da posição de um robô móvel.

Os trabalhos sobre ofuscação em SEDs apresentados em [15, 30] consideram que um estado secreto do sistema nunca é reportado pelo ofuscador. Entretanto, em alguns casos é necessário que o estado secreto seja reportado para preservar todo o comportamento do sistema como será apresentado no Exemplo 1.

***Exemplo* 1** *Consideramos o caso de uma ferrovia em que um ou mais trens se deslocam sobre o mesmo conjunto de trilhos, como ilustrado na Figura 1.1. Através da utilização de sensores, algumas informações úteis sobre a geolocalização dos trens são enviadas para usuários externos através de uma rede de comunicação. Desta forma, é possível estimar a localização dos trens, o que permite traçar rotas mais eficientes, evitando congestionamentos. Porém, se um usuário mal intencionado conhecer a localização exata de cada trem o mesmo poderá causar danos graves ao sistema. Assim, pontos de interesse como intercessões e pontes precisam de um cuidado especial para serem reportados aos usuários externos e por isso são modeladas como estados secretos.*

O Exemplo 1 ilustra uma situação em que é necessário para o ofuscador reportar a passagem pela ponte, modelada como estado secreto, para preservar todo o

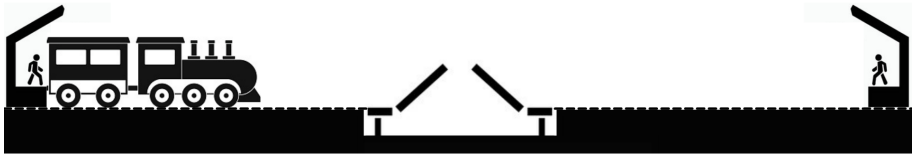

Figura 1.1: Linha de trem simplificada.

comportamento original, uma vez que o trem só é capaz de transitar entre as duas estações através da ponte. Portanto, é necessário desenvolver um ofuscador que seja capaz de reportar os estados secretos do sistema e ao mesmo tempo garantir a privacidade do mesmo. Neste trabalho será apresentado uma nova definição de ofuscação para sistemas a eventos discretos cuja principal diferença é permitir que, em determinadas situações, os estados secretos do sistema sejam reportados para os usuários externos. Além de proteger o anonimato do ofuscador, uma vez que o comportamento do sistema original é preservado, esta abordagem amplia os sistemas que podem ser ofuscados, pois é mais permissiva no tratamento dos estados secretos quando comparada a abordagens anteriores como apresentadas em [15] e [30].

O presente trabalho é organizado da seguinte forma. No capítulo 2 são apresentados os conhecimentos preliminares necessários para a compreensão deste tralhado, como sistemas a eventos discretos, lógica temporal e controle supervisório baseado em lógica temporal. A Seção 3 apresenta a formulação do problema da síntese do ofuscador; além disso apresentamos um algoritmo para a síntese do sistema aumentado, os procedimentos para sintetizar o ofuscador que garatam diferentes níveis de privacidade e utilidade, proposto neste trabalho, e diferenciamos nossa abordagem das propostas anteriormente na literatura. Por fim, a conclusão do trabalho e descrição de trabalhos futuros são apresentados no capítulo 4.

Capítulo 2

Embasamento Teórico

Neste capítulo iremos apresentar os conceitos necessários sobre SEDs, lógica temporal e controle supervisório utilizados neste trabalho. O embasamento teórico sobre SEDs e lógica temporal apresentados neste capítulo são baseados em BAIER e KATOEN [3], CASSANDRAS e LAFORTUNE [7]. A Seção 2.1 introduz o conceito de SEDs, onde são expostos o formalismo adotado para a modelagem dos sistemas a eventos discretos apresentados neste trabalho (subseção 2.1.1), formas de analisar o comportamentos destes modelos (subseção 2.1.2) e como regular o comportamento concorrente entre diversos modelos individuais (subseção 2.1.3). Por sua vez, a Seção 2.2 apresenta a ideia de lógica temporal, em que é descrita a forma como as propriedades do sistema são especificadas. Por fim, a Seção 2.3 introduz o conceito de síntese de controle supervisório a partir de especificações temporais, onde é apresento um algoritmo capaz de solucionar o problema de forma eficiente.

2.1 Sistemas a Eventos Discretos

Sistemas a eventos discretos são uma classe de sistemas cujo espaço de estados é um conjunto discreto e, além disso, a evolução temporal desses sistemas ocorre por meio de transições, as quais são associadas a eventos que são observados em pontos discretos no tempo [7].

O conjunto de estados representa uma situação específica do sistema. Por exemplo, uma válvula pode ser modelada por um sistema com dois estados, um estado representa a situação em que a válvula está fechada e o outro estado em que a válvula está aberta. Assim, SEDs podem possuir um número muito grande de estados para representar todas as possíveis situações que o sistema pode apresentar.

Por sua vez, os eventos ocorrem de forma instantânea e ativam as transições, o que leva à mudança do estado atual do sistema. Desta forma, os eventos podem ser vistos como uma ação específica (por exemplo, um botão é pressionado), também podem estar associados com algo que acontece de forma espontânea (por exemplo,

um alarme é ativado) ou podem ser o resultado da ocorrência de uma situação previamente determinada (por exemplo, a leitura de um sensor atinge um valor pré determinado). Para ilustrar, considere que existem dois botões, um responsável por abrir e outro por fechar a válvula, no caso em que a válvula esta fechada e um operário pressiona o botão para abrir, o evento abrir válvula é disparado e o sistema transita do estado válvula fechada para o estado válvula aberta.

Na próxima subseção mostraremos como modelar sistemas a eventos discretos utilizando sistemas de transição rotulados.

2.1.1 Sistemas de Transição Rotulados

Uma forma comum de modelar o comportamento de sistemas a eventos discretos é através da utilização de sistemas de transição. Um sistema de transição é composto por estados, os quais podem ou não possuir rótulos. Cada estado representa uma situação específica do sistema e a evolução do mesmo ocorre através de transições que ligam um estado a outro.

***Definição* 1** *Um sistema de transição rotulado é definido como* $TS = (S, Act, \rightarrow, I, AP, L)$*, em que:*

- S *é o conjunto de estados,*
- Act *é o conjunto de eventos,*
- $\rightarrow \subseteq S \times Act \times S$ *é uma relação de transição,*
- $I \subseteq S$ *é o conjunto dos estados iniciais,*
- AP *é o conjunto de proposições atômicas,*
- $L : S \rightarrow 2^{AP}$ *é a função de rotulação.*

No caso de um sistema de transição rotulado, o conjunto de eventos, Act, é utilizado para nomear as transições. Assim a evolução do sistema pode ser vista através de uma sequência de eventos. Por outro lado, as proposições atômicas presentes no conjunto AP são premissas a respeito dos estados e são utilizadas para definir qual situação específica do sistema cada estado representa. Portanto, a função de rotulação $L(s)$ atribui a cada estado $s \in S$ um conjunto de proposições atômicas pertencente ao conjunto 2^{AP}.

***Exemplo* 2** *Consideramos o caso de uma válvula. Modelaremos seu comportamento através de um sistema de transição rotulado* $TS_a = (S_a, Act_a, \rightarrow_a, I_a, AP_a, L_a)$*, em que* $S_a = \{s_0, s_1\}$ *é o conjunto de estados,* $Act_a = \{\alpha_1, \beta_1\}$ *o conjunto de eventos,*

$I_A = \{s_0\}$ o estado inicial, $AP_a = \{v_1\}$ o conjunto de proposições atômicas e a função de rotulação é dada por $L_a(s_0) = \{\neg v\}$ e $L_a(s_1) = \{v\}$. Já a relação de transição, $\rightarrow_a$, é definida da seguinte forma: $s_0 \xrightarrow{\alpha_1}_a s_1$ e $s_1 \xrightarrow{\beta_1}_a s_0$. O sistema de transição TS_a é apresentado na Figura 2.1.

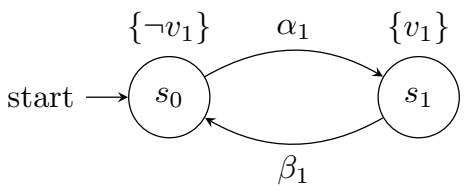

Figura 2.1: Exemplo sistema de transição rotulado abrir e fechar válvula.

Os eventos α_1 e β_1 indicam o abrir e fechar da válvula respectivamente. Por outro lado, a proposição atômica v_1 representa a condição da válvula estar aberta e a proposição atômica $\neg v_1$ a condição da válvula fechada. Assim, caso v_1 for verdadeiro no estado a válvula encontra-se aberta e, analogamente, caso v_1 for falso a válvula está fechada naquele estado. Desta forma, como $L_a(s_0) = \{\neg v\}$, no estado s_0, a válvula está fechada e, por sua vez, $L_a(s_1) = \{v\}$ indica que no estado s_1 a válvula está aberta.

Na próxima subseção são expostas formas de analisar o comportamento de sistemas de transição rotulados através de execuções, caminhos e traços.

2.1.2 Execuções, Caminhos e Traços de um Sistema de Transição Rotulado

Uma maneira de analisar o comportamento de um sistema de transição rotulado é através das possíveis execuções que este sistema pode realizar. Uma execução consiste na sequência de estados visitados pelo sistema dada uma sequência de eventos gerada que são observados [3]. Desta forma, podemos definir o conceito de execução da seguinte forma:

Definição 2 *Dado um o sistema de transição $TS = (S, Act, \rightarrow, I, AP, L)$, um fragmento de execução finito ϱ deste sistema é dado pela sequência de estados visitados, começando por um estado inicial, alternada com uma sequência de eventos observada, i.e.,*

$$\varrho = s_0\alpha_1 s_1\alpha_2 s_2\alpha_3 ... \text{ tal que } s_i \xrightarrow{\alpha_{i+1}} s_{i+1} \text{ para todo } 0 \leq i \leq n,$$

em que $n \geq 0$ é chamado de comprimento da execução. Um fragmento de execução infinito é dado pela sequência alternada de estados e eventos como definido a seguir:

$$\varrho = s_0\alpha_1 s_1\alpha_2 s_2\alpha_3 ... \text{ tal que } s_i \xrightarrow{\alpha_{i+1}} s_{i+1} \text{ para todo } 0 \leq i.$$

Uma execução é dita máxima se não pode ser prolongada. Uma execução do sistema de transição TS é o fragmento de execução inicial, ou seja, fragmentos que iniciam em algum estado inicial, e também é máximo. Além disso, definimos o conceito de estados alcançáveis de um sistema de transição

***Definição* 3** *Considere o sistema de transição $TS = (S, Act, \rightarrow, I, AP, L)$. Para que um estado $s \in S$ seja dito alcançável é necessário que exista um fragmento finito de execução inicial com o seguinte formato*

$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} s_n = s.$$

Assim, um estado é alcançável em TS se existe uma execução que termina neste estado, começando a partir de algum estado inicial qualquer. O conjunto de estados alcançáveis do sistema de transição TS é dado por $Reach(TS)$.

***Definição* 4** *Considere o sistema de transição $TS = (S, Act, \rightarrow, I, AP, L)$. Para um dado estado $s \in S$ e um dado evento $\alpha \in Act$ o conjunto dos α-sucessores diretos de s é dado por*

$$Post(s, \alpha) = \{s' \in S | s \xrightarrow{\alpha} s'\}, \; Post(s) = \bigcup_{\alpha \in Act} Post(s, \alpha).$$

Outra forma de analisar o comportamento de um sistema de transição rotulado é observar a sequência de estados visitados durante uma execução, os chamados caminhos, que carregam a mesma noção de uma execução, porém omitindo os eventos. Assim, definimos um fragmento finito de caminho como apresentado a seguir

***Definição* 5** *Dado um sistema de transição $TS = (S, Act, \rightarrow, I, AP, L)$, um fragmento finito de caminho é representado pela sequência de estados $s_0 s_1 ... s_n$ em que $s_i \in Post(s_{i-1})$ para todo $0 \leq i \leq n$, no qual $n \geq 0$. Por sua vez, um fragmento infinito de caminho é dado pela sequência infinita de estados $s_0 s_1 ...$ em que $s_i \in Post(s_{i-1})$ para todo $i > 0$. Um fragmento de caminho é inicial se o primeiro estado da sequência pertence ao conjunto de estados iniciais de TS e é dito máximo se não pode ser prolongado. Desta forma, um caminho de TS é um fragmento de caminho inicial e máximo.*

Para analisar um caminho, devemos verificar a sequência de proposições atômicas presentes nos estados deste caminho. Assim, ao invés de considerar-

mos uma execução $s_0\alpha_1 s_1\alpha_2 s_2\alpha_3 s_3...$ analisaremos os rótulos dos estados visitados $L(s_0)L(s_1)L(s_3)...$, chamado de traço, que representa a sequência de proposições atômicas válidas durante uma execução como definido a seguir

***Definição* 6** *Considere o sistema de transição $TS = (S, Act, \rightarrow, I, AP, L)$. O traço do fragmento infinito de caminho $\pi = s_0 s_1...$ é dado por $traço(\pi) = L(s_0)L(s_1)...$ já o traço do fragmento finito de caminho $\hat{\pi} = s_0 s_1 ... s_n$ é dado por $traço(\pi) = L(s_0)L(s_1)...L(s_n)$. Assim o traço de um fragmento de caminho pode ser visto como uma palavra no alfabeto 2^{AP}.*

***Exemplo* 3** *Considere o sistema de transição rotulado $TS_b = (S_b, Act_b, \rightarrow_b, I_b, AP_b, L_b)$, em que $S_b = \{s_0, s_1, s_2\}$, $Act_b = \{\alpha_1, \alpha_2, \alpha_3\}$, $I_b = \{s_0\}$, $AP = \{a, b, c\}$ e a função de transição, $\rightarrow_b$, junto à função de rotulação, L_b, são definidas como apresentado na Figura 2.2.*

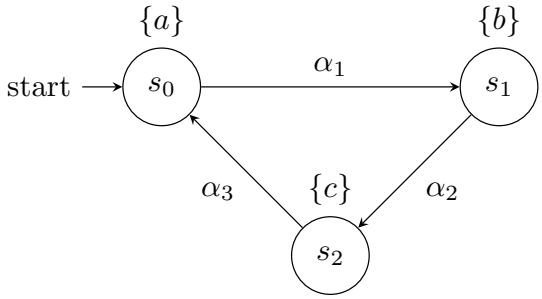

Figura 2.2: Exemplo execução, caminho e traço de um sistema de transição rotulado.

Assim, um possível fragmento inicial finito de execução deste sistema de transição rotulado é dado pela sequência alternada de estados e eventos $\varrho = s_0\alpha_1 s_1\alpha_2 s_2\alpha_3 s_0$, já um fragmento inicial infinito de execução deste sistema, ou somente execução, é dado pela sequência $s_0\alpha_1 s_1\alpha_2 s_2\alpha_3 s_0\alpha_1....$ Note que todos os estados do sistema são alcançáveis, ou seja $Reach(TS_b) = \{s_0, s_1, s_2\}$, uma vez que sempre existe uma execução na forma $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} s_n = s_i$, para $i = 1, 2, 3$. Por outro lado, a partir do fragmento inicial finito de execução apresentado anteriormente obtemos o seguinte fragmento inicial finito de caminho que foi percorrido pelo sistema $s_0 s_1 s_2 s_0$. Analogamente, para o caso da execução o seguinte caminho foi percorrido $s_0 s_1 s_2 s_0....$ Por fim, a traço gerado através do fragmento inicial finito de execução é $L(s_0)L(s_1)L(s_2)L(s_0) = abca$ e o traço da execução é $L(s_0)L(s_1)L(s_2)L(s_0) = abca....$

Notamos que existem diversas formas de analisar o comportamento que um sistema de transição rotulado pode apresentar. Assim, a forma adequada irá depender

do objetivo da análise. Por exemplo, para verificar se um modelo se comporta de uma maneira específica devemos checar se todos os possíveis traços que aquele modelo pode gerar condizem com o comportamento desejado.

Na próxima subseção apresentamos como modelar o comportamento concorrente entre sistemas de transição rotulados individuais, para isso é apresentada a operação *handshaking*.

2.1.3 Sistemas de Transição Rotulados Concorrentes

Para estudar o comportamento concorrente entre dois ou mais sistemas de transição utilizaremos o mecanismo *handshaking* [3]. Considere os sistemas de transição $TS_i = (S_i, \rightarrow_i, Act_i, I_i, AP_i, L_i)$, em que $i = 1, 2$ e seja $H = Act_1 \cap Act_2$. Com isso, é possível definir a composição síncrona $TS_1 \| TS_2$ da seguinte forma:

$$TS_1 \| TS_2 := (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

em que $L((s_1, s_2)) = L_1(s_1) \cup L_2(s_2)$ e a relação de transição $\rightarrow$ é definida como:

- Se $\alpha \in H$, $(s_1 \xrightarrow{\alpha}_1 s_1') \wedge (s_2 \xrightarrow{\alpha}_2 s_2')$ então $(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2')$,
- Se $\alpha \notin H$, $(s_1 \xrightarrow{\alpha}_1 s_1')$ (resp. $(s_2 \xrightarrow{\alpha}_2 s_2')$) então $(s_1, s_2) \xrightarrow{\alpha} (s_1', s_2)$ (resp. $(s_1, s_2) \xrightarrow{\alpha} (s_1, s_2')$).

Portanto, no mecanismo de *handshaking* a execução de um evento comum a ambos os processos, $\alpha \in Act_1 \cap Act_2$, somente poderá ser executada pelos dois processos simultaneamente. Por outro lado, um evento particular de um processo, $\alpha \in (Act_1 \setminus Act_2) \cup (Act_2 \setminus Act_1)$, pode ser executado assim que ativo de forma independente em cada processo.

***Exemplo* 4** *Considere agora que, além de modelar o comportamento de uma válvula como apresentado no Exemplo 2, desejamos modelar o comportamento concorrente entre duas válvulas. Para isso, além do sistema de transição rotulado TS_a apresentado no Exemplo 2, criamos o sistema de transição $TS_c = (S_c, Act_c, \rightarrow_c, I_c, AP_c, L_c)$, em que $S_c = \{s_0', s_1'\}$, $Act_c = \{\alpha_2, \beta_2\}$, $I_b = \{s_0'\}$, $AP_b = \{v_2\}$ e a relação de transição, $\rightarrow_c$, com a função de rotulação, L_c, são definidas como na Figura 2.3.*

O sistema de transição rotulado TS_c modela o comportamento de uma segunda válvula e, portanto, sua estrutura é análoga ao sistema de transição rotulado apresentado no Exemplo 2. Assim, de posse dos sistemas de transição TS_a e TS_c podemos realizar a operação de handshaking e obter a composição síncrona entre eles. O sistema de transição obtido foi denominado de $TS_{ac} = (S_{ac}, Act_{ac}, \rightarrow_{ac}, I_{ac}, AP_{ac}, L_{ac})$, em que $S_{ac} = S_a \times S_b = \{(s_0, s_0'), (s_0, s_1'), (s_1, s_0'), (s_1, s_1')\}$, $Act_{ac} = Act_a \cup Act_c = \{\alpha_1, \beta_1, \alpha_2, \beta_2\}$, $I_{ac} = \{(s_0, s_0')\}$, $AP_{ac} = \{v_1, v_2\}$, já a relação de transição, $\rightarrow_{ac}$,

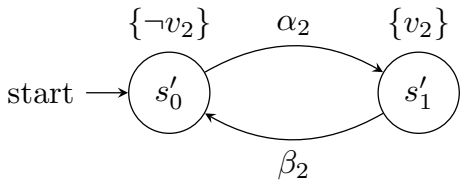

Figura 2.3: Exemplo sistema de transição rotulado abre e fecha segunda válvula.

e a função de rotulação, L_{ac}, são definidas como ilustrado na Figura 2.4. Como $L((s_0, s'_0)) = \{\neg v_1, \neg v_2\}$, no estado inicial as válvula 1 e 2 estão fechadas. Do mesmo modo, $L((s_0, s'_1)) = \{\neg v_1, v_2\}$ indica que no estado (s_0, s'_1) a válvula 1 está fechada porém a válvula 2 está aberta. Já $L((s_1, s'_0)) = \{v_1, \neg v_2\}$ mostra que em (s_1, s'_0) a válvula 1 está aberta e a válvula 2 fechada. Por fim, $L((s_1, s'_1)) = \{v_1, v_2\}$ determina que no estado (s_1, s'_1) tanto a válvula 1 quanto a 2 estão abertas.

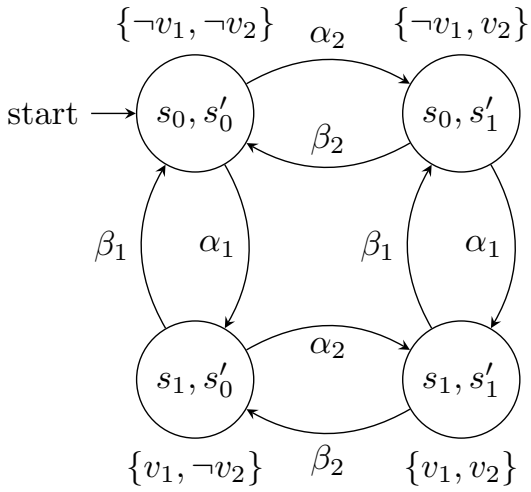

Figura 2.4: Exemplo composição síncrona entre os sistemas de transição rotulados das Figuras 2.1 e 3.3.

Portanto, através da operação de *handshaking* é possível modelar o comportamento concorrente entre diversos sistemas de transição rotulados que foram modelados individualmente.

A próxima Seção introduz o conceito de lógica temporal. Um formalismo lógico utilizado para descrever o comportamento de um sistema de acordo com a evolução temporal que o sistema está sujeito.

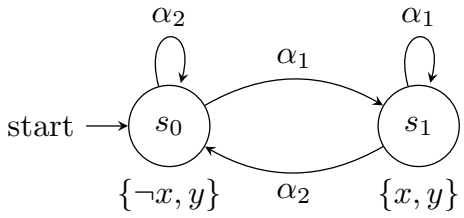

Figura 2.5: Exemplo de sistema de transição rotulado para busca em ramificações.

2.2 Lógica Temporal

A verificação de modelos, como os sistemas de transição rotulados, tem o intuito de checar se o sistema em questão satisfaz uma dada especificação através de uma busca exaustiva de todos os possíveis comportamentos que tal sistema pode apresentar [7]. Tal especificação pode ser descrita através de lógicas temporais, como lógica temporal linear e lógica de árvore de computação [3], e descrevem o comportamento do sistema enquanto navega através de seus estados. Lógica temporal pode expressar as mais diversas condições, por exemplo alcançabilidade, não bloqueio e vivacidade [7].

Lógica temporal é uma maneira eficiente de descrever o comportamento de um sistema através da evolução dos rótulos dos estados durante as execuções [3]. Uma forma usual de especificar propriedades de um sistema através de lógica temporal é lógica de árvore de computação (LAC) [3], um formalismo lógico temporal baseado em ramificações. O conceito de ramificações está ligado ao fato de que podem existir diversos futuros distintos para o mesmo instante de tempo. Assim, para os sistemas de transição rotulados uma ramificação que inicia em um determinado estado exprime todas as possíveis evoluções deste sistema que iniciam neste estado.

***Exemplo* 5** *Para ilustrar a busca através das ramificações considere o sistema de transição rotulado* $TS_d = (S_d, Act_d, \rightarrow_d, I_d, AP_d, L_d)$*, em que* $S = \{s_0, s_1\}$*,* $Act_d = \{\alpha_1, \alpha_2\}$*,* $I_d = \{s_0\}$*,* $AP_d = \{x\}$ *e a função de transição,* $\rightarrow_d$*, com a função de rotulação,* L_d*, são definidas como ilustrado na Figura 2.5.*

Para realizar a busca em ramificações devemos checar todos os possíveis futuros que iniciam no estado em análise naquele instante de tempo. Assim, considerando que a busca é iniciada a partir do estado inicial, o estado s_0 *possui dois possíveis futuros de acordo com a função de transição: é possível permanecer no estado* s_0 *através do autolaço ou evoluir para o estado* s_1*. Assim, o estado inicial,* s_0*, é ramificado em dois estados,* s_0 *e* s_1*. Esse processo é repetido de forma exaustiva, navegando através de todas as possíveis execuções que, neste caso, podem possuir comprimento infinito. Parte do resultado é apresentado na Figura 2.6.*

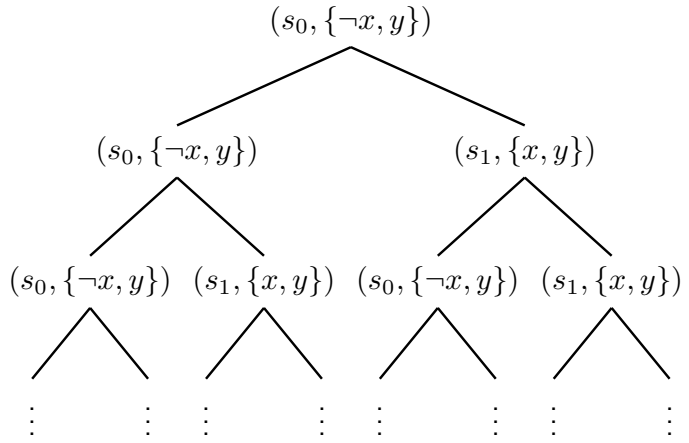

Figura 2.6: Busca em ramificações para o sistema de transição rotulado da Figura 2.5.

As expressões em LAC podem possuir diversos operadores temporais, porém para este trabalho somente os operadores temporais AG, abreviação do inglês para *always globally*, e EF, abreviação do inglês para *eventually future*, serão considerados. Dito isso, a formula em LAC $AG(\phi)$ significa que todos os estados de todos os caminhos devem satisfazer a formula ϕ. Por outro lado, a fórmula em LAC $EF(\phi)$ especifica que deve existir pelo menos um caminho capaz de alcançar um estado em que ϕ seja satisfeito. Também é valido ressaltar que $AG(\phi_1) \wedge AG(\phi_2) \iff AG(\phi_1 \wedge \phi_2)$.

Agora precisamos definir quando um sistema de transição rotulado satisfaz uma fórmula em LAC. Para isso, introduziremos o conceito de relação de satisfação. Dado um sistema de transição rotulado qualquer $TS = (S, Act, \rightarrow, I, AP, L)$, o conjunto de estados que satisfazem a fórmula LAC ϕ, denotado por $Sat(\phi)$, é definido como:

$$Sat(\phi) = \{s \in S \mid s \models \phi\}$$

Desta forma, o sistema de transição TS satisfaz a fórmula ϕ, ou seja $TS \models \phi$, se e somente se ϕ é válido para todos os estados iniciais, ou seja $I \subseteq Sat(\phi)$. Observando a Figura 2.6 percebemos que o sistema TS_d, do Exemplo 5, satisfaz a fórmula $AG(y)$, uma vez que todos os estados de todos os caminhos partindo do estado inicial satisfazem a fórmula, isto é $I_d \subseteq Sat(AG(y))$. Por outro lado, TS_d satisfaz a fórmula $EF(x)$ uma vez que a partir do estado inicial existe pelo menos um caminho em que um estado com rótulo x é alcançado, em outras palavras $I_d \subseteq Sat(EF(x))$. Existem diversas ferramentas de verificação de modelos disponíveis capazes de checar especificações em LAC, como o SMV e variações recentes como o NuSMV.

A Seção seguinte irá tratar do caso de controle supervisório com especificações temporais em LAC. Desta forma, é exposto um algoritmo computar o supervisor ótimo para um tipo especial de especificação LAC.

2.3 Controle Supervisório com Especificações Temporais

Sistemas a eventos discretos podem modelar os mais diversos tipos de sistemas físicos. Após modelar todo o comportamento de um sistema, podemos querer evitar que determinadas situações aconteçam por motivos de segurança através da desabilitação de certos eventos em determinados estados do sistema. Desta forma, o objetivo do controle supervisório neste trabalho é restringir o comportamento do sistema para evitar situações indesejadas. O problema de controle supervisório é considerado solucionado se existe um controlador capaz de forçar o sistema a satisfazer determinada especificação e ele puder ser construído [27]. Existem diversas formas de se calcular um controlador para sistemas a eventos discretos. WONHAM *et al.* [27] apresentam uma abordagem baseada na linguagem gerada através da observação dos eventos do sistema. Uma outra forma é RAWLINGS *et al.* [24]. que calcula o supervisor através de especificações em lógica de árvore de computação (LAC).

Neste trabalho utilizaremos a abordagem proposta por RAWLINGS *et al.* [24]. Desta forma, definimos as especificações do controle supervisório através de especificações em LAC. As especificações podem envolver os operadores temporais AG, que requer que a propriedade seja válida para todos os estados alcançáveis, e EF, que requer que pelo menos um estado alcançável satisfaça a propriedade. Considerando que exitem algumas transições do sistema que podem ser desabilitadas, podemos alterar o comportamento do sistema para que o mesmo satisfaça uma especificação que naturalmente não é satisfeita. Portanto, o sistema é modelado através de um sistema de transição rotulado $TS = (S, Act, \rightarrow, I, AP, L)$. Pode-se definir que algumas transições são controláveis e o conjunto de transições controláveis é representadas por $\rightarrow_c \subseteq \rightarrow$. As transições controláveis podem ou não ser desabilitadas e o conjunto de transições controláveis desabilitadas é representado por $\rightarrow_d \subseteq \rightarrow_c$. Para a síntese do controle supervisório é necessário definir uma especificação ϕ e verificar quais transições devem ser desabilitadas para que a evolução de TS sujeito a tais desabilitações garanta que o sistema satisfaça ϕ.

O controle supervisório tem como objetivo maximizar o conjunto de estados S que satisfazem uma dada especificação ϕ para um dado conjunto de transições desabilitadas $\rightarrow_d$, ou seja $Sat(\phi)_{(S,(\rightarrow \setminus \rightarrow_d))} = max(Sat(\phi)_{(S,(\rightarrow \setminus \rightarrow'_d))})$. Além disso,

como objetivo secundário, devemos minimizar o conjunto de transições desabilitadas $\rightarrow_d$, ou seja $min(\rightarrow_d)$. Desta forma, após combinar ambos os objetivos obtemos o seguinte problema de síntese de controle supervisório

$$min(\rightarrow_d) \text{ tal que } Sat(\phi)_{(S,(\rightarrow\setminus\rightarrow_d))} = max(Sat(\phi)_{(S,(\rightarrow\setminus\rightarrow'_d))}),\ (\rightarrow'_d, \rightarrow_d) \subseteq (\rightarrow_c) \tag{2.1}$$

Primeiro iremos tratar o caso de controle supervisório baseado em lógica temporal para especificações com apenas um operador temporal, AG ou EF. O algorítimo 1 foi proposto por RAWLINGS *et al.* [24] para solucionar o problema da síntese de controle supervisório ótimo, apresentado na equação 2.1, para especificações LAC com apenas um operador temporal. O algoritmo 1 verifica qual tipo de operador temporal foi passado como especificação: no passo 1, verifica EF, e no passo 3, AG. No caso de uma especificação do tipo $EF(\phi)$ nenhuma transição precisa ser desabilitada ao se maximizar o conjunto de estados $Sat(EF(\phi))$, como pode ser verificado no passos 2. Por outro lado, para especificações do tipo $AG(\phi)$ é necessário desabilitar as transições $S^* \rightarrow (S\setminus S^*)$, ou seja, é necessário desabilitar as transições que levam de um estado que satisfaz a especificação ϕ para estados que não satisfazem, como pode ser verificado nos passos 4 e 5.

Algorithm 1 Síntese de Controle Supervisório Ótimo para Sistemas de Transição Rotulados Sujeitos a um Único Operador Temporal

Input: $TS = (S, Act, \rightarrow, I, AP, L),\ \phi$

Output: $(\rightarrow_d) \subseteq (\rightarrow_c)$

1: **if** $EF(\phi)$ **then**

2: $\rightarrow_d = \emptyset$ ▷ nenhuma transição é desabilitada

3: **else if** $AG(\phi)$ **then**

4: calcule $S^* := Sat(\phi)_{(S,(\rightarrow\setminus\rightarrow_c))}$

5: em seguida desabilite as transições $\rightarrow_d = \{(s, s') \in (\rightarrow_c) | s \in S \wedge s' \notin S^*\}$

return $\rightarrow_d$

O algorítimo 1 é embasado pelo lema 1 e pelo teorema 1 [24], descritos como se segue.

***Lema* 1** *Se* $\rightarrow \subseteq \rightarrow'$, *então* $Sat(EF(\phi))_{(s,\rightarrow)} \subseteq Sat(EF(\phi))_{(S,\rightarrow')}$

Pelo lema 1, para todo estado $s \in Sat(EF(\phi))_{(S,\rightarrow)}$ existe um caminho em TS, para S e $\rightarrow$, que leva até um estado em que a proposição atômica p é verdadeira. Desta forma, pelo fato de $\rightarrow \subseteq \rightarrow'$, esse mesmo caminho existe no sistema de transição TS, para o conjunto de estados S e o novo conjunto de transições $\rightarrow'$. Portanto $s \in Sat(EF(\phi))_{(S,\rightarrow')}$.

O caso da invariância, AG, é apresentado no teorema 1.

***Teorema* 1 (Controle Supervisório para Invariância)** *O algorítimo 1 calcula a solução para a equação 2.1 para uma especificação do tipo $AG(\phi)$, em que ϕ não contém nenhum outro operador temporal.*

Agora iremos tratar o caso de controle supervisório baseado em lógica temporal para um tipo especial de combinação de operadores temporais. Especificações em LAC no formato $AG(EF(\phi))$, em que ϕ é uma proposição atômica e não contém nenhum outro operador temporal, expressam especificações de controlabilidade e de não bloqueio [10] e podem ser utilizadas para a síntese de controle supervisório. Uma extensão desse tipo considera o caso em que múltiplos conjuntos de estados sempre devem ser alcançáveis. Para isso, são utilizadas as especificações de invariância e alcançabilidade combinadas, do inglês *combined invariance and reachability* (CIR), que possuem o seguinte formato:

$$AG\left(\bigwedge_{\mathrm{i}\in\mathrm{I}} p_i \wedge \bigwedge_{\mathrm{j}\in\mathrm{J}} EF(p_j)\right) \tag{2.2}$$

Quando um sistema de transição rotulado satisfaz uma especificação do tipo CIR, significa que todos os estados de todos os caminhos partindo do conjunto de estados iniciais devem satisfazer p_i e, adicionalmente, todos os estados devem possuir pelo menos um caminho que alcance um estado com rótulo p_j. Portanto, caso um sistema de transição rotulado não satisfaça determinada especificação CIR, podemos verificar a existência de um controlador que seja capaz de alterar o comportamento do sistema para desabilitar a ocorrência de eventos que levem para estados que não satisfazem a especificação.

RAWLINGS *et al.* [24] desenvolveram um algoritmo (2) capaz de sintetizar controladores para especificações do tipo CIR. O algoritmo recebe como entrada um sistema de transição rotulado e uma especificação CIR e retorna um conjunto de transições que devem ser desabilitadas. O algoritmo é iterativo e parte da subfórmula mais interna, ϕ_{in} definida no passo 1. No passo 5, verifica o conjunto de estados que satisfazem essa sub fórmula, $Sat(\phi_{in})_{(S,\rightarrow^{k-1})}$, e estende para todos os possíveis caminhos $Sat(AG(s \in Z^k_{in})_{(S,(\rightarrow^{k-1}\setminus\rightarrow_c)}$, dado o conjunto de transições controláveis $(\rightarrow^{k-1} \setminus \rightarrow_c)$. Desta forma é possível desabilitar as transições que levam para estados que não satisfazem a especificação, $(\rightarrow^k_d) \leftarrow \{(s, s^+)|s \in Z^k \wedge s^+ \notin Z^k\}$. Por fim, o conjunto de transições desabilitadas é atualizado e o algoritmo é executado novamente até que nenhuma transição nova seja desabilitada.

***Exemplo* 6** *Para ilustrar a síntese de controle supervisório através da utilização do algorítimo 2, considere o sistema de transição rotulado $TS_e = (S_e, Act_e, \rightarrow_e, I_e, AP_e, L_e)$, em que $S_e = \{s_0, s_1, s_2, s_3\}$, $Act = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, $I = \{s_0\}$,*

Algorithm 2 Síntese de Controle Supervisório Ótimo para Sistemas de Transição Rotulados Sujeitos a Especificações do tipo CIR

Input: $TS = (S, Act, \rightarrow, I, AP, L)$, $\phi = AG\left(\bigwedge_{i \in I} p_i \wedge \bigwedge_{j \in J} EF(p_j)\right)$, $\rightarrow_c \subseteq$

Output: $(\rightarrow_d) \subseteq (\rightarrow_c)$

1: $\phi_{in} \leftarrow \bigwedge_{i \in I} p_i \wedge \bigwedge_{j \in J} EF(p_j)$
2: $k \leftarrow 0$
3: $(\rightarrow_d^k) \leftarrow \emptyset$
4: $(\rightarrow^k) \leftarrow (\rightarrow \setminus \rightarrow_d^k)$
5: **repeat**

- $k \leftarrow k + 1$
- $Z_{in}^k \leftarrow Sat(\phi_{in})_{(S, \rightarrow^{k-1})}$
- $Z^k \leftarrow Sat(AG(s \in Z_{in}^k))_{(S, (\rightarrow^{k-1} \setminus \rightarrow_c))}$
- $(\rightarrow_d^k) \leftarrow \{(s, s^+) | s \in Z^k \wedge s^+ \notin Z^k\}$
- $(\rightarrow^k) \leftarrow (\rightarrow^{k-1} \setminus \rightarrow_d^k)$

6: **until** $(\rightarrow^k) = (\rightarrow^{k-1})$
7: **return** $\rightarrow_d^k = 0$

$AP = \{\emptyset\}$ e a função de transição, $\rightarrow_e$, junto à função de rotulação, L_e, são retratadas na Figura 2.7.

Utilizamos como entrada para o algoritmo 2 o sistema de transição rotulado TS_e e a especificação CIR $AG(s \neq s_3 \wedge EF(s = s_1))$, que significa que o estado s_3 nunca poderá ser alcançado e para todos os estados deve ser possível evoluir para o estado s_1. Desta forma, na primeira iteração do algoritmo apenas o estado s_3 não satisfaz a formula interna $\phi_{in} = s \neq s_3 \wedge EF(s = s_1)$ e, portanto, a transição α_4 é desabilitada. Na segunda iteração o estado s_2 deixa de satisfazer a fórmula interna, pois o estado s_1 não é mais alcançável a partir de s_2, assim a transição α_1 também é desabilitada. Em seguida, nenhuma outra transição é desabilitada e o controlador obtido é apresentado na Figura 2.8, em que as transições desabilitadas são representadas por setas tracejadas. A saída do algoritmo 2 é o conjunto de transições desabilitadas $\rightarrow_d = \{s_0 \xrightarrow{\alpha_1}_e s_2\}$. Perceba que após desabilitar a ocorrência do evento α_1, no estado s_0, os estados s_2 e s_3 não são mais alcançáveis a partir do estado inicial, assim não é necessário desabilitar a transição α_4.

Para facilitar a utilização desta metodologia para sistemas com um número muito grande de estados, RAWLINGS *et al.* [24] desenvolveu um programa que implementa o algoritmo 2, chamado de SynthSMV. Esse programa é uma variação da ferramenta de verificação de modelos NuSMV [8] e assim utiliza sua conhecida eficiência para análise de modelos simbólicos, como sistemas de transição rotulados, para solucionar a síntese de controle supervisório para especificações CIR com múltiplos requisitos

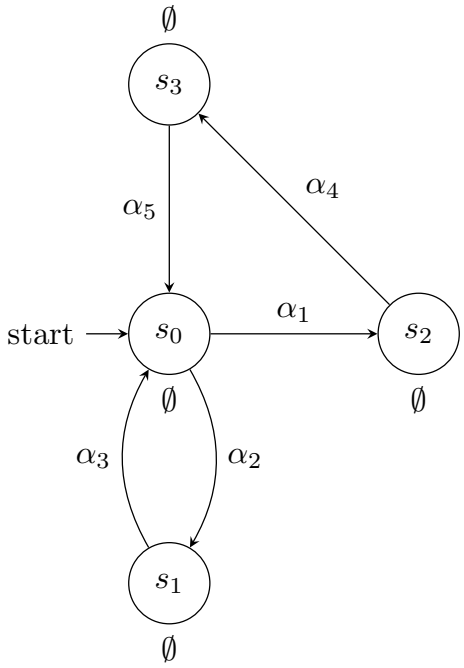

Figura 2.7: Sistema de Transição Rotulado TS_e.

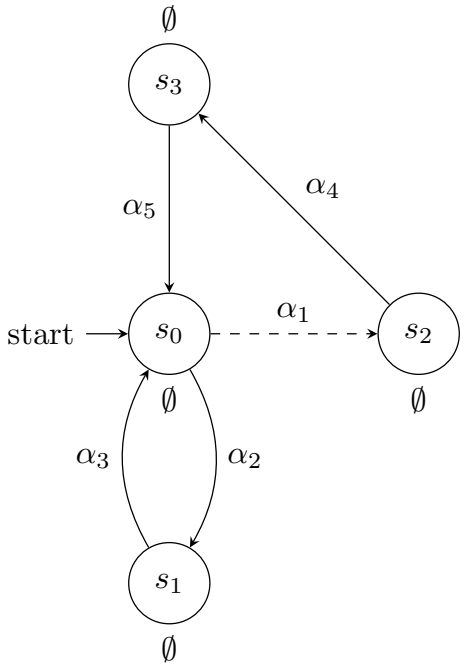

Figura 2.8: Controle Supervisório para o sistema de transição rotulado TS_e em que transições desabilitadas são representadas por setas tracejadas.

de alcançabilidade. A seguir é apresentado o código utilizado no SynthSMV para solucionar a síntese de controle supervisório do Exemplo 6.

```
MODULE main
VAR state : # {s0,s1,s2,s3};
IVAR event : # {alpha1,alpha2,alpha3,alpha4,alpha5};
INIT state = s0;
ASSIGN next(state) :=
  case
    state = s0:
      case
        event = alpha1 : s2;
        event = alpha2 : s1;
        TRUE : 0;
      esac;
    state = s1:
      case
        event = alpha3 : s0;
        TRUE : 0;
      esac;
    state = s2:
      case
        event = alpha4 : s3;
        TRUE : 0;
      esac;
    state = s3:
      case
        event = alpha5 : s0;
        TRUE : 0;
      esac;
  esac;

DEFINE
    Ss := state = s3

CTRBL event = {alpha1,alpha2,alpha3,alpha4,alpha5};
SPEC AG(!Ss & EF(state = s1)); -- falso
SYNTH AG(!Ss & EF(state = s1)); -- verdadeiro
```

O código para implementação do SynthSMV é divido em duas partes. Na pri-

meira parte devemos declarar o sistema de transição rotulado, no caso do Exemplo 6 $TS_e = (S_e, Act_e, \rightarrow_e, I_e, AP_e, L_e)$, em que o conjunto de estados S_e é declarado em "$VAR\ state$", o conjunto de eventos Act_e em "$VAR\ event$", o estado inicial I_e em "$INIT\ state$", a relação de transição $\rightarrow_e$ em "$ASSIGN\ next(state)$", e o conjunto de proposições atômicas, AP_e, junto à função de rotulação dos estados, L_e, são declarados em "$DEFINE$". Na segunda parte do código o SynthSMV utiliza da verificação de modelos presente no NuSMV para verificar se o sistema de transição rotulado TS_e satisfaz a especificação "$SPEC\ AG(!Ss \wedge EF(state = s1))$", e como o resultado é negativo realiza a síntese de controle supervisório para esta especificação dado o conjunto de transições controláveis, declarados em "$CTRBL\ event$", retornando o conjunto de transições desabilitadas. A saída do SynthSMV para a solução do Exemplo 6 foi o conjunto de transições desabilitadas $s_0 \xrightarrow{\alpha_1}_d s_2$.

O próximo capítulo introduz a síntese de uma nova política de ofuscação com garantias de diferentes níveis de privacidade e utilidade. Em primeiro lugar é apresentado um algoritmo para criar o sistema aumentado, um sistema que modela o comportamento concorrente entre o sistema a ser ofuscado e sua visão pública. Em seguida é aplicado o controle supervisório para sintetizar o ofuscador e o resultado é comparado com outras abordagens de ofuscação que garantem privacidade.

Capítulo 3

Síntese de uma Nova Política de Ofuscação

Neste capítulo consideramos a síntese de uma nova política de ofuscação para sistemas de transição rotulados que garatam diferentes níveis tanto de privacidade quanto de utilidade da informação gerada pelo sistema. A Seção 3.1 apresenta a formulação do problema. Já a Seção 3.2 apresenta um algoritmo para síntese do sistema de transição aumentado, o qual modela a iteração entre o sistema original e a visão pública do mesmo. Em seguida, a Seção 3.3 define uma nova política de ofuscação que permite, além de variar o nível de utilidade da informação gerada como em WU *et al.* [30], permita variar o nível de privacidade. Por fim, a Seção 3.4 diferencia a abordagem proposta neste trabalho das existentes na literatura.

3.1 Formulação do Problema

O sentido da palavra ofuscar remete a confundir alguém ou obscurecer o sentido de algo. Essa é justamente a ideia por trás do conceito de ofuscação que tratamos neste trabalho, uma vez que tentamos confundir os observadores de um dado sistema para que eles não consigam saber quando o sistema se encontra em um estado dito secreto. Como ilustrado na Figura 3.1, para realizar a ofuscação consideramos que todo evento, α, gerado pelo sistema seja enviado ao ofuscador que reporta ao ambiente uma palavra, α_o, de forma a manter os segredos do sistema protegido dos agentes que observam a evolução do sistema externamente. Diferentemente das abordagens anteriores de ofuscação, consideraremos neste trabalho que, em determinadas situações, os estados secretos do sistema poderão ser reportados pelo ofuscador. Desta forma, sistemas que possuem estados secretos no seu caminho principal e necessariamente precisariam ser reportados para preservar a dinâmica do sistema original poderão ser ofuscados. Como pode ser observado no Exemplo

1.1, o trem necessariamente precisa passar sobre a ponte para transitar entre as duas estações, pois é o único caminho disponível. Considerando a ponte como o estado secreto do sistema, seria impossível criar, a partir das abordagens anteriores, um ofuscador que garanta a utilidade da informação, uma vez que o ofuscador não conseguiria reportar a passagem pelo estado secreto. Portanto, a nova definição de ofuscação proposta neste trabalho amplia os sistemas que podem ser ofuscados, além de dificultar para os agentes mal-intencionados detectar a presença do ofuscador a partir de observações incoerentes. Com isso, as seguintes hipóteses são feitas sobre a estrutura do ofuscador:

- O evento gerado pelo sistema é imediatamente enviado para o ofuscador,
- O agente possui uma cópia do modelo do sistema,
- O agente somente consegue observar os eventos reportados pelo ofuscador.

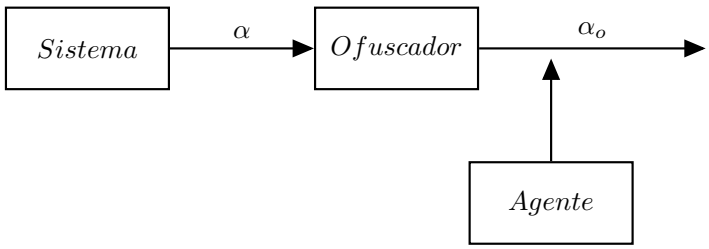

Figura 3.1: Estrutura do ofuscador.

Inspirado no problema de controle supervisório para sistemas de transição [24], nós iremos definir o problema da síntese do ofuscador como se segue.

***Definição* 7 Problema de Síntese do Ofuscador:** *Dado um sistema de transição rotulado* $TS = (S, Act, \rightarrow, I, AP, L)$*, um conjunto de estados secretos* $S_s \subseteq S$ *e uma especificação em LAC* θ*, então o ofuscador* TS_{ofusc} *deve satisfazer os seguintes critérios:*

- *maximiza o conjunto de estados que satisfazem a especificação* θ
- *minimiza o conjunto de transições desabilitadas pelo ofuscador*

Desta forma, a próxima Seção descreve o procedimento para criar o sistema aumentado, um sistema de transição rotulado que modela o comportamento concorrente entre o sistema a ser ofuscado e sua visão pública.

3.2 Síntese do Sistema Aumentado

Para realizar a síntese do ofuscador é necessário ser possível diferenciar a ocorrência de um evento na planta de sua observação por um agente externo. Para modelar a observação dos eventos por agentes externos, considere os conjuntos Act_r e $Act_{r\omega}$ como cópias de Act porém com os eventos renomeados com os subescritos r e $r\omega$ respectivamente. Desta forma, podemos definir as operações de renomeação dos eventos, R_r e R_ω, da seguinte forma.

A operação de renomeação R_r é definida como $R_r : Act^* \to Act_r^*$, em que:

- $R_r(\varepsilon) = \varepsilon$,
- $R_r(\alpha) = \alpha_r$ e $R_r(s\alpha) = R_r(s)R_r(\alpha) = R_r(s)\alpha_r$, para $\alpha \in Act$ e $s \in Act^*$.

Analogamente definimos a operação de renomeação $R_\omega : Act_r^* \to Act_{r\omega}^*$:

- $R_{r\omega}(\varepsilon) = \varepsilon$,
- $R_{r\omega}(\alpha_r) = \alpha_{r\omega}$ e $R_{r\omega}(s_r\alpha_r) = R_{r\omega}(s_r)R_{r\omega}(\alpha_r) = R_{r\omega}(s)\alpha_{r\omega}$, para $\alpha_r \in Act_r$ e $s_r \in Act_r^*$.

Uma forma de representar a concorrência entre as execuções dos eventos na planta e as possíveis observações realizadas pelo agente externo é através da criação de um sistema aumentado conforme descrito no algoritmo 3. No primeiro passo do algoritmo 3, é criado uma cópia do conjunto de estados S chamada de S_p. Já no passo 2, é criado o conjunto Act_r a partir da operação R_r que contém todos os eventos de Act renomeados com o subescrito r. O passo 3 cria o conjunto $Act_{r\omega}$ a partir da operação R_ω que contém todos os eventos de Act_r renomeados com o subescrito $r\omega$. Desta forma, no passo 5 definimos Act_p como a união de Act_r e $Act_{r\omega}$ que representam as observações dos eventos em Act por algum agente externo. No passo 5 definimos a visão pública do sistema, TS_p, que possui o mesmo comportamento do sistema de transição rotulado original e representa o sistema que os agentes externos observam. Para tanto, é necessário definir a sua relação de transição, $\to_p \subseteq S_p \times Act_p \times S_p$, em que substitui-se as transições $\to$ rotuladas por $\alpha \in Act$, para todos os estados $s_p \in S_p$, por duas novas transições rotuladas $\alpha_r \in Act_p$ e $\alpha_{r\omega} \in Act_p$.

Após computar a visão pública do sistema, precisamos definir o comportamento concorrente entre o sistema original e a visão pública do mesmo. Para isso, no passo 6 é definido o sistema de transição TS_1, que representa se é a vez do sistema original executar um evento, estado com rotulo $\neg o$, ou da visão pública do mesmo gerar uma observação, estado com rótulo o. Desta forma, como $I_1 = \{\neg o\}$, foi definido que o sistema original executará o primeiro evento no sistema aumentado. O evento ω é

Algorithm 3 Procedimento para criar o Sistema Aumentado

Input: $TS = (S, Act, \rightarrow, I, AP, L)$
Output: TS_{aug}

1: $S_p \leftarrow S$
2: $Act_r = R_r(Act)$
3: $Act_{r\omega} = R_\omega(Act_r)$
4: $Act_p \leftarrow Act_r \cup Act_{r\omega}$
5: Defina $TS_p = (S_p, Act_p, \rightarrow_p, I, AP, L)$ em que:

- $(s_{1_p} \xrightarrow{\alpha_r}_p s_{2_p}) \leftarrow (s_1 \xrightarrow{\alpha} s_2)$, em que $s_{1_p}, s_{2_p} \in S_p$, $\alpha_r = R_r(\alpha)$, $s_1, s_2 \in S$ e $\alpha \in Act$
- $(s_{1_p} \xrightarrow{\alpha_{r\omega}}_p s_{2_p}) \leftarrow (s_1 \xrightarrow{\alpha} s_2)$, em que $s_{1_p}, s_{2_p} \in S_p$, $\alpha_{r\omega} = R_\omega(R_r(\alpha))$, $s_1, s_2 \in S$ e $\alpha \in Act$

6: Defina $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1)$ em que:

- $S_1 = \{0, 1\}$
- $Act_1 = Act \cup Act_r \cup Act_{r\omega} \cup \omega$
- $0 \xrightarrow{\alpha}_1 1$ se $\alpha \in Act$
- $1 \xrightarrow{\alpha}_1 1$ se $\alpha \in Act_r$
- $1 \xrightarrow{\alpha}_1 0$ se $\alpha \in \{Act_{r\omega} \cup \omega\}$
- $AP_1 = \{o, \neg o\}$
- $L_1(0) = \{\neg o\}$
- $L_1(1) = \{o\}$
- $I_1 = \{0\}$

7: Defina $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2)$ em que:

- $S_2 = \{0, 1\}$
- $Act_2 = Act \cup Act_r \cup Act_{r\omega} \cup \omega$
- $1 \xrightarrow{\alpha}_2 1$ se $\alpha \in \{Act_r \cup Act_{r\omega} \cup \omega\}$
- $1 \xrightarrow{\alpha}_2 0$ se $\alpha \in Act$
- $0 \xrightarrow{\alpha}_2 0$ se $\alpha \in Act$
- $0 \xrightarrow{\alpha}_2 1$ se $\alpha \in \{Act_r \cup Act_{r\omega} \cup \omega\}$
- $AP_2 = \{p, \neg p\}$
- $L_2(0) = \{\neg p\}$
- $L_2(1) = \{p\}$
- $I_2 = \{1\}$

8: $TS_{aug} = TS \| TS_p \| TS_1 \| TS_2$

responsável por alternar a vez entre a visão pública e o sistema original, ou seja, ele causa a mudança do sistema do estado com rótulo o para o estado com rótulo $\neg o$. Um evento com o subescrito $r\omega$ representa a ocorrência de um evento com o subescrito r somado a ocorrência do evento ω. Assim, eventos com o subescrito $r\omega$ representam que a visão pública gerou uma observação e o sistema aumentado alternou a vez para o sistema original. Assim, toda vez que o sistema executa um evento, a visão pública do sistema tem a possibilidade de gerar uma ou mais observações até que o evento ω ocorra. No passo 7 é definido o sistema de transição TS_2, que registra se quem gerou o último evento foi o sistema, estado com rótulo $\neg p$, ou a visão pública, estado com rótulo p. Por fim, no passo 8 o sistema aumentado, TS_{aug}, é formado pelo *handshaking* entre os sistemas TS, TS_p, TS_1 e TS_2 e representa todas as possíveis execuções de eventos pelo sistema original e as possíveis observações realizadas pelos agentes externos através da visão pública.

Exemplo 7 *Para ilustrar a criação do sistema aumentado, considere o sistema de transição apresentado na Figura 3.2 que modela o comportamento de um trem que viaja da estação* 1 *para a estação* 3*. Portanto,* $TS^2 = (S^2, Act^2, \rightarrow^2, I^2, AP^2, L^2)$ *em que* $S^2 = \{1, 2, 3\}$*,* $Act^2 = \{a, b, c\}$*,* $I^2 = \{1\}$*,* $AP^2 = \emptyset$*,* $L^2(s) = \emptyset$*,* $\forall s^2 \in S^2$*, e* $\rightarrow^2$ *é definido como ilustrado na Figura 3.2.*

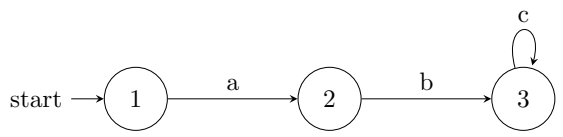

Figura 3.2: Sistema de transição TS^2 para o comportamento do trem do Exemplo 7

Para a síntese do ofuscador, precisamos primeiro obter o sistema aumentado seguindo o algoritmo 3. No primeiro passo do algoritmo criamos o conjunto S_p^2 *a partir de uma cópia de* S^2*, ou seja,* $S_p^2 = \{1, 2, 3\}$*. Em seguida criamos* Act_r^2 *através da renomeação do conjunto* Act^2 *pela função* R_r*, assim obtemos* $Act_r^2 = \{a_r, b_r, c_r\}$*. Da mesma forma, criamos o conjunto* $Act_{r\omega}^2$ *através da renomeação do conjunto* Act_r^2 *pela função* R_ω*, assim* $Act_{r\omega}^2 = \{a_{r\omega}, b_{r\omega}, c_{r\omega}\}$*. Com isso, no passo 4 definimos o conjunto* $Act_p^2 = \{a_r, a_{r\omega}, b_r, b_{r\omega}, c_r, c_{r\omega}\}$*. No passo 5, definimos a visão pública do sistema dada por* $TS_p^2 = (S_p^2, Act_p^2, \rightarrow_p^2, I^2, AP^2, L^2)$*, com isso substituíamos as transições* $\rightarrow^2$ *rotuladas por* $\sigma \in Act^2$*, para todos os estados* $s_p^2 \in S_p^2$*, por duas novas transições rotuladas* $\sigma_r \in Act_p^2$ *e* $\sigma_{r\omega} \in Act_p^2$*. Por exemplo, a transição do estado 1 para o estado 2,* $(1 \xrightarrow{a} 2)$*, é substituída pelas transições* $(1 \xrightarrow{a_r}{}_p^2\, 2)$ *e* $(1 \xrightarrow{a_{r\omega}}{}_p^2\, 2)$*. Em* TS_p^2*, a substituição das demais transições é realizado de forma análoga como pode ser visto na Figura 3.3.*

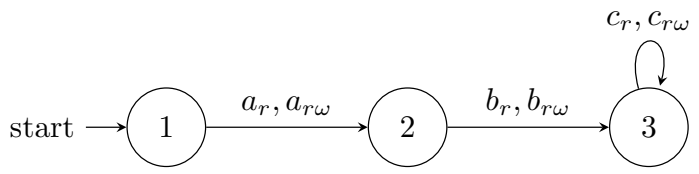

Figura 3.3: Visão pública do sistema de transição do Exemplo 2, TS_p^2

No passo 6 devemos montar o sistema de transição $TS_1^2 = (S_1^2, Act_1^2, \rightarrow_1^2, I_1^2, AP_1^2, L_1^2)$*, em que* $Act_1^2 = \{a, a_r, a_{r\omega}, b, b_r, b_{r\omega}, c, c_r, c_{r\omega}, \omega\}$*, representado na Figura 3.4.*

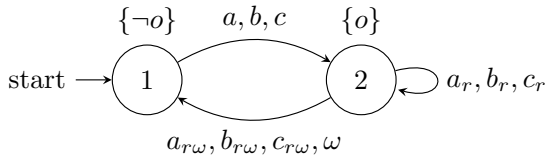

Figura 3.4: Sistema de transição que modela a alternância entre sistema e visão pública do Exemplo 2, TS_1^2

O sistema de transição TS_1^2 *representa a alternância entre o sistema original e a visão pública do mesmo. Assim, o estado 1, com rótulo* $\neg o$*, caracteriza a vez do sistema original gerar um evento e o estado 2, com rótulo o, caracteriza a vez da visão pública reportar um evento. Observe que após o sistema original gerar um evento, ou seja, a ocorrência de algum evento do conjunto* $Act^2 = \{a, b, c\}$*, a visão pública sempre tem a possibilidade de reportar um evento contido no conjunto* $Act_r^2 \cup Act_{r\omega}^2 \cup \omega$*. Por fim, é valido salientar que o estado 1 é o inicial pois possui o rótulo* $\neg o$*, o que garante que o sistema original seja o primeiro a evoluir.*

No passo 7 é definido o sistema de transição $TS_2^2 = (S_2^2, Act_2^2, \rightarrow_2^2, I_2^2, AP_2^2, L_2^2)$*, representado na Figura 3.5.Por sua vez, o sistema de transição* TS_2^2 *representa qual dos agentes envolvidos na ofuscação gerou o último evento, o sistema original ou a visão pública. Desta forma, as transições que levam para o estado 1, p, são compostas apenas por eventos contidos no conjunto* $Act_r^2 \cup Act_{r\omega}^2 \cup \omega$ *e, portanto, caracterizam que no estado 1 a visão pública gerou o último evento. Analogamente, as transições que levam para o estado 2,* $\neg p$*, são constituídas por eventos do conjunto* Act^2 *indicando, assim, que no estado 2 o sistema original gerou o último evento. Observe que neste caso o estado inicial é indiferente, pois no estado inicial nenhum dos agentes gerou o último evento, assim foi escolhido o estado 1 como inicial de*

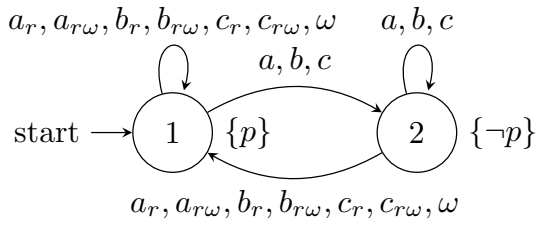

Figura 3.5: Sistema de transição que indica qual agente gerou o último evento, TS_2^2

forma arbitrária.

Por fim, para sintetizar o sistema aumentado utilizaremos a operação de handshaking. Assim $TS_{aug}^2 = TS^2 \| TS_p^2 \| TS_1^2 \| TS_2^2$. *Parte do sistema aumentado obtido é retratado na Figura 3.6.*

Observe que o conjunto de estados S_{aug}^2 *é formado pela composição* $S^2 \times S_p^2 \times S_1^2 \times S_2^2$, *porém, para simplificação da Figura 3.6, optou-se pela nomeação dos estados na forma* $S^2 \times S_p^2$. *Além disso, considera-se que a qualidade de informação gerada pelo ofuscador, ou a sua utilidade, diminui à medida que o número de transições que liga o estado real do sistema ao estado reportado pelo ofuscador aumenta. Por consequência, buscamos construir um ofuscador que, além de garantir a privacidade, mantenha a utilidade da informação gerada. Para isso, definimos a* ***função de utilidade*** $D : S \times S \to \mathbb{N}$ *que modela a distância entre dois estados do sistema. Por exemplo, para os estados* $(2, 3)$ *e* $(3, 2)$ *o valor da função de utilidade é dado por* $D(2, 3) = D(3, 2) = 1$.

A partir do resultado obtido nota-se que o sistema aumentado modela quais eventos a visão pública pode reportar em resposta a qual evento o sistema original gerou. Por exemplo, a ocorrência do evento a no sistema original acarreta nas possíveis respostas pela visão pública: $a_r, a_{r\omega}, \omega$. *O evento* a_r *significa que a visão pública reportou a ocorrência do evento a para os observadores externos e mantém a vez de evoluir com a visão pública. Já o evento* $a_{r\omega}$ *também reporta a ocorrência do evento a, mas, por conter o subescrito* ω, *alterna a vez e espera o sistema original gerar um novo evento. Por fim, o evento* ω *não reporta a ocorrência de um evento naquele intervalo e retorna a vez para o sistema original evoluir. Portanto, cada evento gerado pelo sistema original acarreta em três possíveis resposta pela visão pública para os observadores externos.*

A próxima Seção define uma nova forma de sintetizar um ofuscador que garanta diferentes níveis de privacidade e utilidade através da utilização do controle supervisório para sistemas de transição rotulados com especificações temporais do tipo CIR.

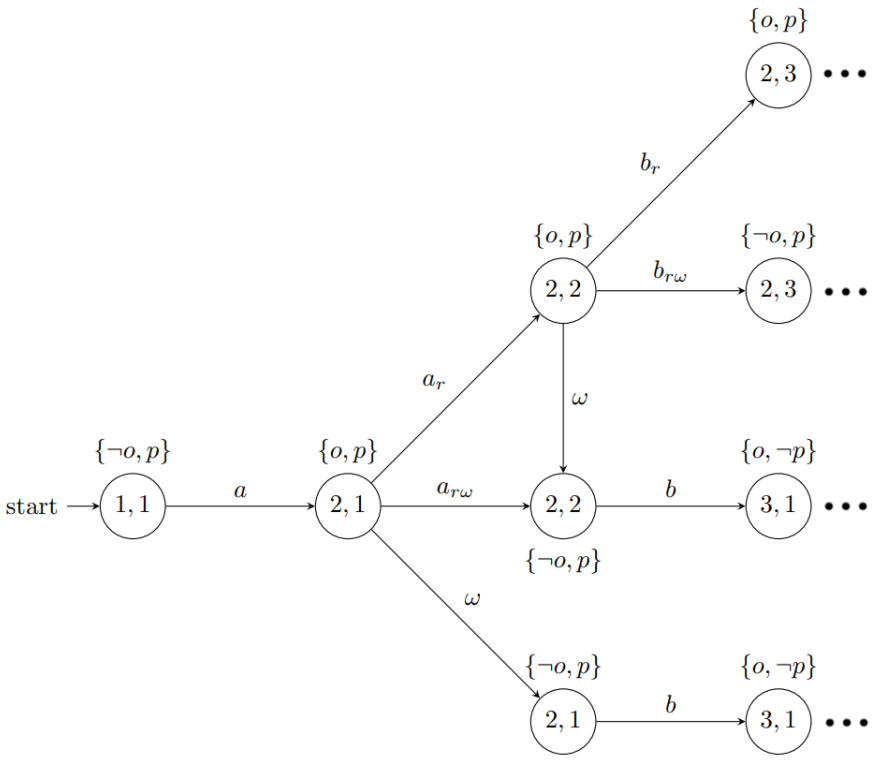

Figura 3.6: Parte do sistema aumentado TS^2_{aug}

3.3 Síntese do Novo Ofuscador com Garantias de Diferentes Níveis de Privacidade e Utilidade

A síntese do ofuscador empregada neste trabalho utiliza da teoria de controle supervisório para limitar o comportamento do sistema aumentado e, com isso, eliminar as observações indesejadas. Para resolver o problema de controle supervisório utilizamos o programa SynthSMV que implementa o algoritmo 2 apresentado em [24]. O SynthSMV utiliza como entradas um sistema de transição rotulado e uma especificação em LAC do tipo CIR e calcula o supervisor como um único *Binary Decision Diagram* (BDD) que registra quais eventos são desabilitados em cada estado do sistema. Portanto, como já definimos a criação do sistema aumentado no algoritmo 3, precisamos definir a especificação LAC que irá limitar o comportamento do mesmo de maneira adequada.

Inicialmente precisamos definir o novo critério de privacidade. Desejamos que somente seja possível para o ofuscador reportar um estado secreto caso o sistema encontre-se em uma posição considerada segura e, analogamente, caso o sistema encontre-se em um estado secreto o ofuscador somente poderá reportar estados considerados seguros. Desta forma, em LAC podemos especificar esta restrição como $AG(p \wedge ((s \vee s_p) \in S_s) \rightarrow D(s, s_p) \geq c_1)$, ou seja, se quem gerou o último evento

foi o ofuscador, p, e o sistema ou o ofuscador encontra-se em um estado secreto, $(s \vee s_p) \in S_s$, então a distância entre o estado real do sistema e o estado reportado deve ser maior ou igual a c_1, iste é, $D(s, s_p) \geq c_1$.

Por outro lado, a informação gerada pelo ofuscador deve ser útil, o que pode ser escrito em LAC como uma especificação do tipo $AG(p \rightarrow D(s, s_p) \leq c_2)$, em que um evento gerado pelo ofuscador, p, não poderá levar o sistema a um estado cuja distância entre o estado real e o estado reportado do sistema seja maior ou igual a c_2, isto é, $D(s, s_p) \leq c_2$. Por fim, é preciso que o sistema aumentado sob controle do supervisor não entre em estados que impeçam o sistema original de evoluir, o que é representado em LAC como $AG(EF(\neg o))$. Assim garantimos que existe pelo menos um caminho que leva a um estado em que é a vez da planta evoluir, o que é representado pela proposição atômica $\neg o$. Portanto, para obter o comportamento desejado pelo ofuscador devemos combinar estas três especificações da seguinte forma:

$$\begin{aligned} AG(((p \wedge (s \vee s_p) \in S_s) \rightarrow (D(s, s_p) \geq c_1)) \wedge \\ (p \rightarrow (D(s, s_p) \leq c_2)) \wedge EF(\neg o)) \end{aligned} \tag{3.1}$$

Assim, o programa SynthSMV recebe como entrada o sistema aumentado, TS_{aug} (algoritmo 3), e a especificação em LAC da equação 3.2. A saída do programa é um supervisor que possui a função ofuscador, denominado TS_{ofusc}, que garante a utilidade e um nível de privacidade variável.

***Exemplo* 8 (Ferrovia simplificada)** *Analogamente ao Exemplo 1.1, consideramos o caso de uma ferrovia simplificada. Considere um trem deslocando-se da estação 1 para 6 como apresentado na Figura 3.7. Desta forma, para este caso temos que* $TS^3 = (S^3, Act^3, \rightarrow^3, I^3, AP^3, L^3)$ *em que* $S^3 = \{1, 2, 3, 4, 5, 6\}$, $Act^3 = \{a, b, c, d, e, f\}$, $I^3 = \{1\}$, $AP^3 = \emptyset$, $L^3(s) = \emptyset$, $\forall s^3 \in S^3$, *e* $\rightarrow^3$ *é definido como apresentado na Figura 3.7. O estado 3, em vermelho, modela uma ponte e por isso é considerado secreto, portanto* $S_s^3 = \{3\}$. *Deseja-se ofuscar este sistema de modo a manter o nível de utilidade e privacidade proposto.*

O primeiro passo na síntese do ofuscador é a criação do sistema aumentado a partir do algoritmo 3 o qual retorna o sistema de transição TS_{aug}^3. *Em seguida devemos estabelecer os parâmetros para que a especificação definida na equação 3.2 satisfaça os níveis de privacidade e utilidade desejados. Para isso, definimos as constantes de privacidade* $c_1 = 1$ *e de utilidade* $c_2 = 2$. *Assim, a partir dos parâme-*

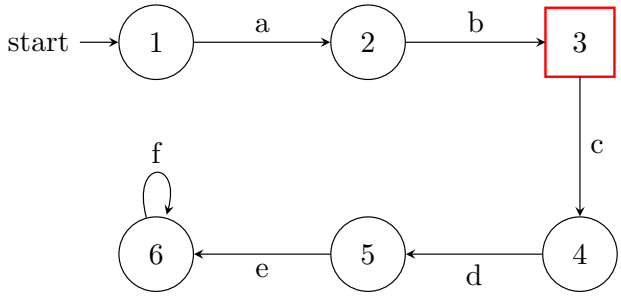

Figura 3.7: Exemplo ferrovia simplificada com ponte.

tros propostos, obtemos a seguinte especificação em LAC:

$$AG(((p \wedge (s^3 \vee s_p^3) \in S_s^3) \rightarrow (D(s^3, s_p^3) \geq 1)) \wedge \\ (p \rightarrow (D(s^3, s_p^3) \leq 2)) \wedge EF(\neg o)) \tag{3.2}$$

em que $AG((p \wedge (s^3 \vee s_p^3) \in S_s^3) \rightarrow (D(s^3, s_p^3) \geq 1))$ *representa o critério de privacidade e significa que sempre que for a vez do ofuscador evoluir e o sistema ou sua visão pública estiverem em um estado secreto então a distância entre o estado real do sistema e o estado da visão pública reportado devem estar a uma distância maior ou igual a 1. Por sua vez, o critério de utilidade é dado por* $AG(p \rightarrow (D(s^3, s_p^3) \leq 2))$ *e define que o ofuscador só poderá evoluir para estados em que a distância entre o estado real do sistema e o estado da visão pública reportado seja maior ou igual a 2. Já a especificação* $AG(EF(\neg o))$ *garante que o sistema original nunca será impedido de evoluir pelo ofuscador.*

A partir disso, o sistema aumentado, TS_{aug}^3*, junto à especificação proposta acima foram utilizadas como entradas para o SynthSMV para obter o ofuscador. A solução encontrada no programa é ilustrada na forma de um sistema de transição na Figura 3.8.*

A partir do resultado fica claro que, para este conjunto de restrições, caso o sistema original evolua para um estado secreto o ofuscador irá restringir o comportamento da visão pública para que ela somente consiga evoluir para estados que satisfazem o critério de privacidade. Esta situação é observada nos estados com nome (3,1), em que a visão pública é obrigada a permanecer no estado 1 para que o sistema original possa evoluir pelo estado secreto, pois $D(3, 1) > 2$ *satisfaz o critério de privacidade. Da mesma forma, o ofuscador somente irá permitir que a visão pública evolua para um estado secreto caso o sistema original encontre-se em um estado que satisfaça o critério de privacidade. Assim, nos estados com nome (5,3) é necessário que o sistema original evolua para uma posição segura para que a visão*

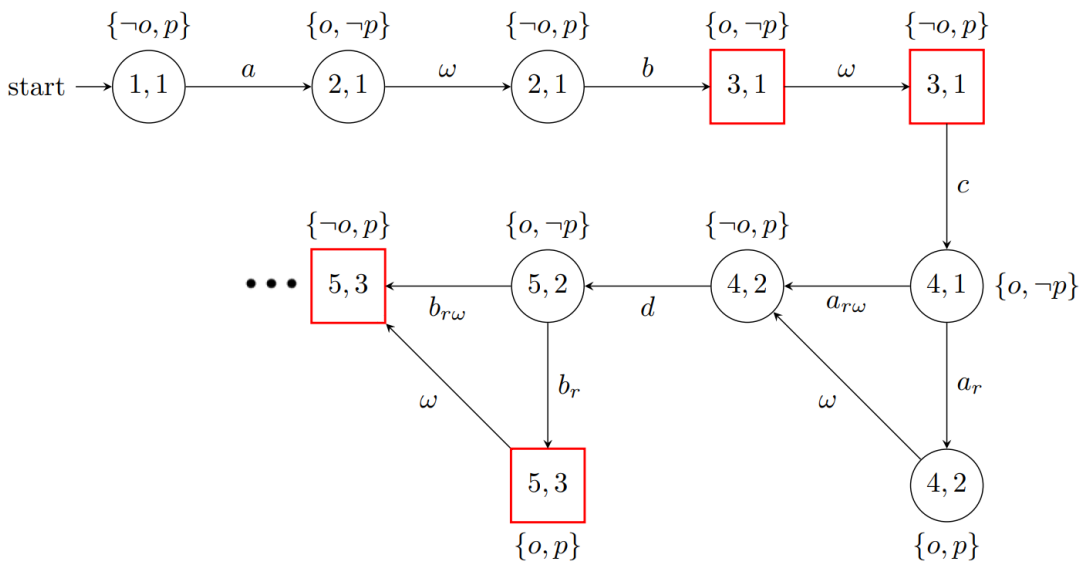

Figura 3.8: Ofuscador sintetizado no SynthSMV para o Exemplo 3.

pública possa evoluir para o estado secreto.

Também é possível observar situações como a do estado $(4, 1)$*, cujo o rótulo é* $\{o, \neg p\}$*, em que a distância entre o estado real e o estado reportado é maior que* $c_2 = 2$*, ou seja* $(D(s^3, s_p^3) \leq 2)$ *não é válido. Desta forma, neste estado o ofuscador desabilitara a ocorrência do evento* ω *e somente permitirá que o sistema transite para estados em que* $p \rightarrow (D(s^3, s_p^3) \leq 2)$ *seja verdadeiro.*

Portanto, este é um exemplo em que as abordagens anteriores, como as utilizadas por [30] e [15], não permitiriam a ofuscação preservando a utilidade da informação gerada, pois consideram que o ofuscador não pode permitir que a visão pública transite para estados secretos, ou seja $AG(s_p \notin S_S)$*, e faz com que a evolução da visão pública fique restrita aos estados 1 e 2. Além disso, pelo fato de restringir o comportamento da visão pública para que não evolua por estados secretos, as abordagens anteriores permitem que, a longo prazo, os observadores suspeitem da existência do ofuscador pois o comportamento não é totalmente preservado. Ambos os problemas são solucionados neste trabalho ao permitir que a visão pública transite pelos estados secretos, mas mantendo a privacidade do sistema como um todo.*

No Exemplo 9, apresentado a seguir, utilizamos a síntese do novo ofuscador com garantias de diferentes níveis de privacidade para um sistema cíclico, que são sistemas em que o estado inicial sempre é alcançável. O exemplo é baseado no problema de distribuição de comida proposto por BARCELOS e BASILIO [4].

***Exemplo* 9 (Distribuição de Comida)** *Considere um sistema que modela a distribuição de comida entre cinco cidades com a utilização de um caminhão, como*

apresentado na Figura 3.9. O sistema apresenta cinco cidades que são interligadas através de vias de mão única, as quais possuem sensores para detectar a passagem do caminhão de distribuição. Assuma que existe um agente mal intencionado que observa o sistema, denominado de intruso, com o objetivo de contaminar a distribuição de comida e assim envenenar a população local. O intruso somente tem a capacidade de contaminar a comida quando o caminhão de distribuição está na cidade 5, assim o objetivo da ofuscação proposta neste trabalho é garantir a privacidade do caminhão durante sua passagem pela cidade 5.

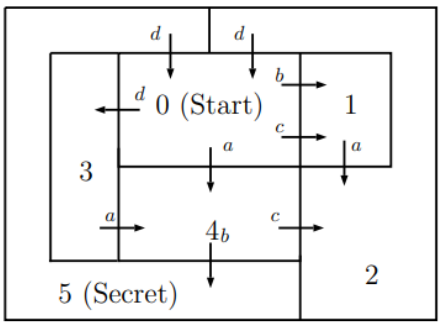

Figura 3.9: Sistema de distribuição de comida entre cinco cidades.

O sistema de transição rotulado para modelar o comportamento do sistema de distribuição proposto foi definido como $TS_{dc} = (S_{dc}, Act_{dc}, \rightarrow_{dc}, I_{dc}, AP_{dc}, L_{dc})$*, em que o conjunto de estados é* $S_{dc} = \{1, 2, 3, 4, 5\}$*, o conjunto de eventos é* $Act_{dc} = \{a, b, c, d\}$*, o conjunto de estados iniciais é* $I_{dc} = \{s_0\}$*, o conjunto de proposições atômicas é* $AP = \emptyset$ *e a função de transição,* $\rightarrow_{dc}$*, junto à função de rotulação,* L_{dc}*, são apresentados na Figura 3.10.*

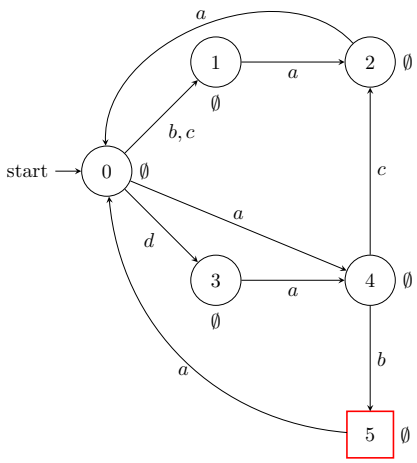

Figura 3.10: Sistema de transição rotulado para a sistema de distribuição de comida.

Como o estado 5 é passível de sofrer um ataque pelo intruso ele é considerado secreto, assim definimos que $S_s = \{5\}$. *Desta forma, nosso objetivo com a ofuscação é confundir o invasor de tal forma que ele somente estime que o caminhão de distribuição está na cidade 5 quando o mesmo estiver de fato em outra cidade. O primeiro passo para solucionar o problema de ofuscação com diferentes níveis de privacidade é construir o sistema aumentado a partir do algoritmo 3. Parte sistema aumentado obtido é apresentado na Figura 3.11, uma vez que possui 108 estados e 372 transições, e foi definido como* TS_{dcAug}.

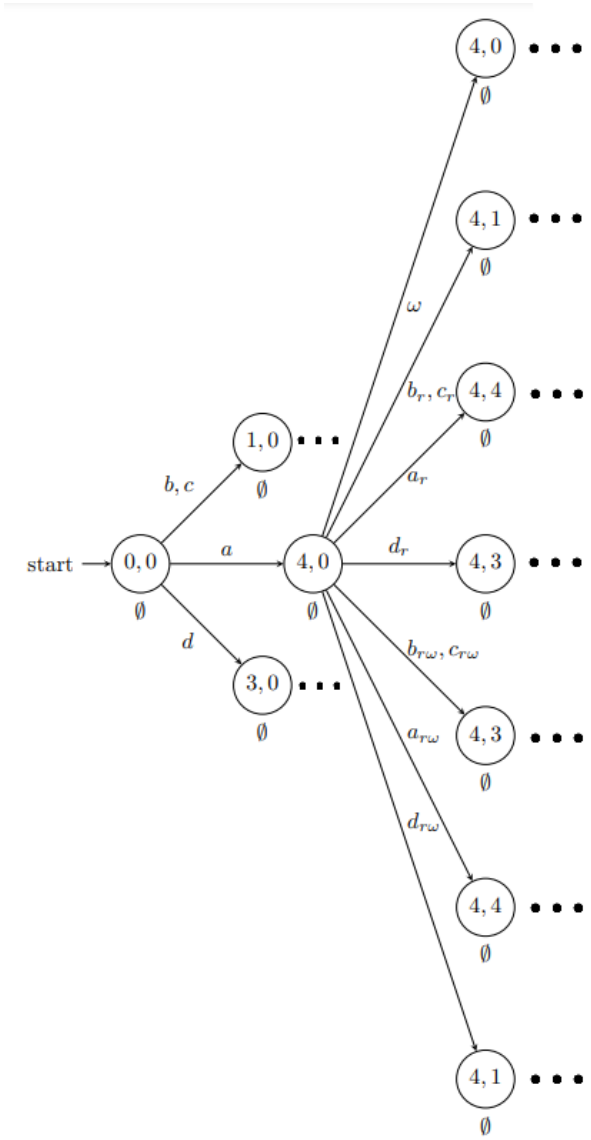

Figura 3.11: Sistema de transição aumentado para a sistema de distribuição de comida.

Assim, o próximo passo é utilizar o sistema de transição aumentado, TS_{dc}, *junto à proposição atômica LAC* $AG(((p \wedge (s \vee s_p) \in S_s) \rightarrow (D(s, s_p) \geq 1)) \wedge EF(\neg o))$ *para*

obter o ofuscador $TS_{dcOfusc}$, que possui 106 estados e 363 transições. A Figura 3.12 apresenta parte do observador do ofuscador obtido considerando apenas os eventos de $Act_r \cup Act_{r\omega}$ como observáveis. Alguns estados e transições foram omitidas, mas e válido ressaltar a partir do observador que todo o comportamento do sistema original é preservado. O intruso será capaz de estimar que o caminhão de comida está na cidade 5 pois é permitida a observação das execuções $s_0as_4bs_5$ e $s_0ds_3as_4bs_5$, porém o ofuscador, através de sua especificação $AG((p \wedge (s \vee s_p) \in S_s) \rightarrow (D(s, s_p) \geq 1))$, garante que fisicamente o caminhão estará em outra cidade o que preserva a privacidade do sistema de distribuição de comida.

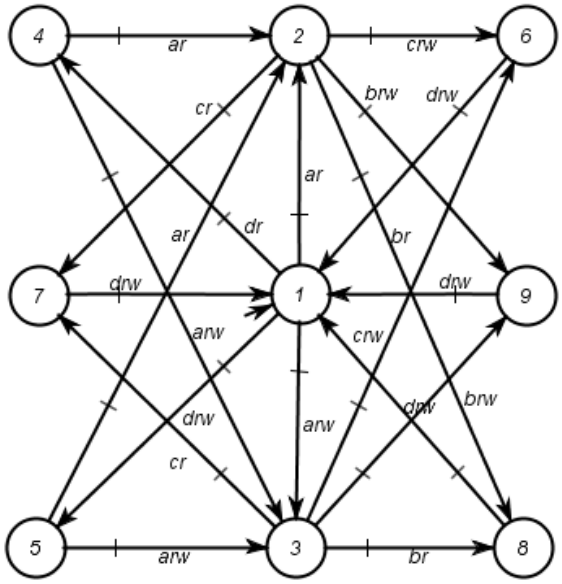

Figura 3.12: Sistema de transição aumentado para a sistema de distribuição de comida.

A próxima Seção realiza uma comparação entre os resultados obtidos com a síntese da nova política de ofuscação com garantias de diferentes níveis de privacidade com técnicas de ofuscação usuais para privacidade.

3.4 Comparação entre a Síntese da Nova Política de Ofuscação que Garante Diferentes Níveis de Privacidade com o Forçamento da Opacidade

Uma noção de privacidade para sistemas a eventos discretos é dada pela opacidade. Opacidade é uma característica do sistema que garante que um determinado segredo sobre o comportamento do sistema seja escondido de um agente externo mal

intencionado. Assim, no contexto de sistemas de transição rotulados, toda execução que termina em um estado secreto do sistema deve possuir uma observação igual a outra execução que termina em um estado não secreto do sistema. Para que esta situação ocorra naturalmente em um sistema de transição rotulado, é necessário que existam eventos que o agente externo não é capaz de observar, os chamados eventos não observáveis, por exemplo a falta de sensores para registrar que determinada transição do sistema ocorreu. Para exemplificar a opacidade, considere o sistema de transição rotulado $TS_{op} = (S_{op}, Act_{op}, \rightarrow_{op}, I_{op}, AP_{op}, L_{op})$, em que o estado $s_1 \in S_{op}$ é considerado secreto, como apresentado na Figura 3.13.

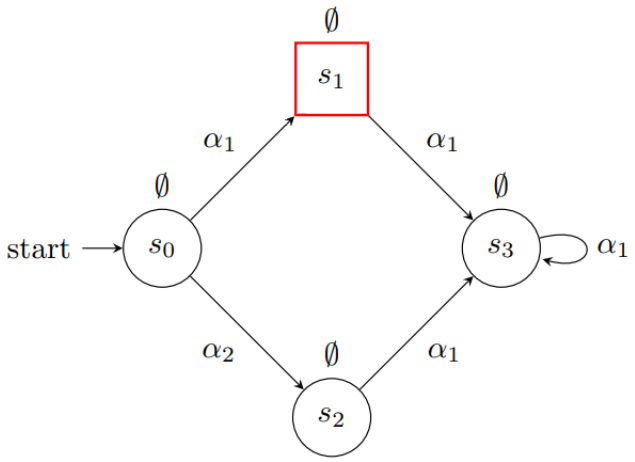

Figura 3.13: Exemplo de sistema com eventos não observáveis.

Assuma que o evento α_2 não é observável, portanto, não é possível determinar com exatidão em qual estado o sistema se encontra, por isso é necessário construir um estimador de estados neste caso chamado de observador. O observador faz uma estimativa de quais estados o sistema original pode estar com base na sequência de eventos observada. Para ilustrar o observador do sistema de transição rotulado TS_{op}, denominado $TS_{obs} = (S_{obs}, Act_{obs}, \rightarrow_{obs}, I_{obs}, AP_{obs}, L_{obs})$, é apresentado na Figura 3.14.

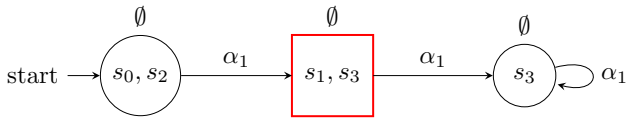

Figura 3.14: Observador do sistema de transição rotulado da Figura 3.13

A partir do observador TS_{obs} notamos que um agente externo nunca será capaz

de deduzir que o sistema TS_{op} está com certeza no estado secreto s_1. Isso se dá pelo fato que não existe um estado no observador em que todos os estados estimados são secretos. Essa característica indica que o sistema de transição rotulado TS_{op} é opaco.

Infelizmente nem todos os sistemas são naturalmente opacos, para isso existem técnicas de ofuscação que verificam a possibilidade de modificar a sequência de eventos observada pelo agente externo de modo que o sistema permaneça opaco. Para demonstrar a ofuscação considere o sistema de transição rotulado $TS_{nop} = (S_{nop}, Act_{nop}, \rightarrow_{nop}, I_{nop}, AP_{nop}, L_{nop})$, em que o estado $s_1 \in S_{nop}$ é secreto e somente os eventos α_1 e α_3 são observáveis, como apresentado na Figura 3.15.

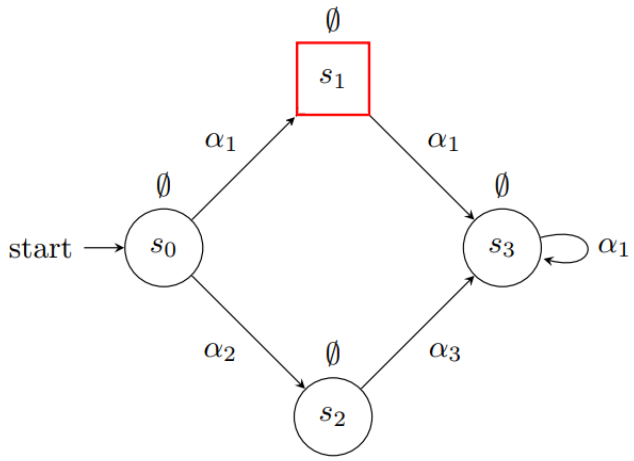

Figura 3.15: Exemplo de sistema para forçar a opacidade

Para verificar se TS_{nop} é opaco foi construído seu observador TS_{nobs}, o qual é exposto na Figura 3.16. A partir de TS_{nobs} notamos que o sistema Ts_{nop} não é opaco, uma vez que existe um estado do observador composto somente por estados secretos, o estado s_1.

É fácil notar que existe pelo menos um caminho de comprimento máximo não bloqueante, $((s_0, s_2), s_3)$, em que os agentes que observam externamente não estimam um estado secreto. Portanto, para forçar a opacidade através da ofuscação, devemos primeiramente criar o sistema aumentado para o sistema de transição TS_{nobs} a partir do algoritmo 3, denominado $TS_{nAug} = (S_{nAug}, Act_{nAug}, \rightarrow_{nAug}, I_{nAug}, AP_{nAug}, L_{nAug})$, parte do sistema aumentado é apresentado na Figura 3.17.

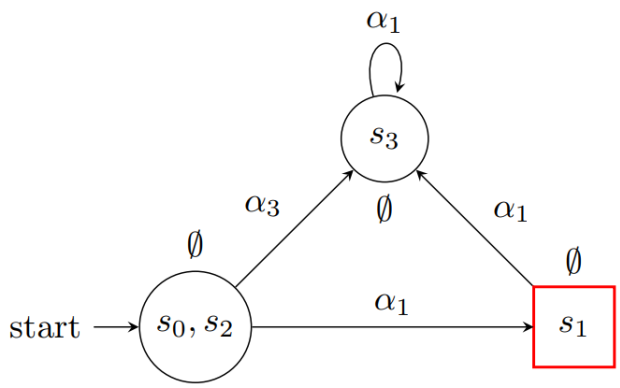

Figura 3.16: Observador do sistema de transição rotulado apresentado na Figura 3.15

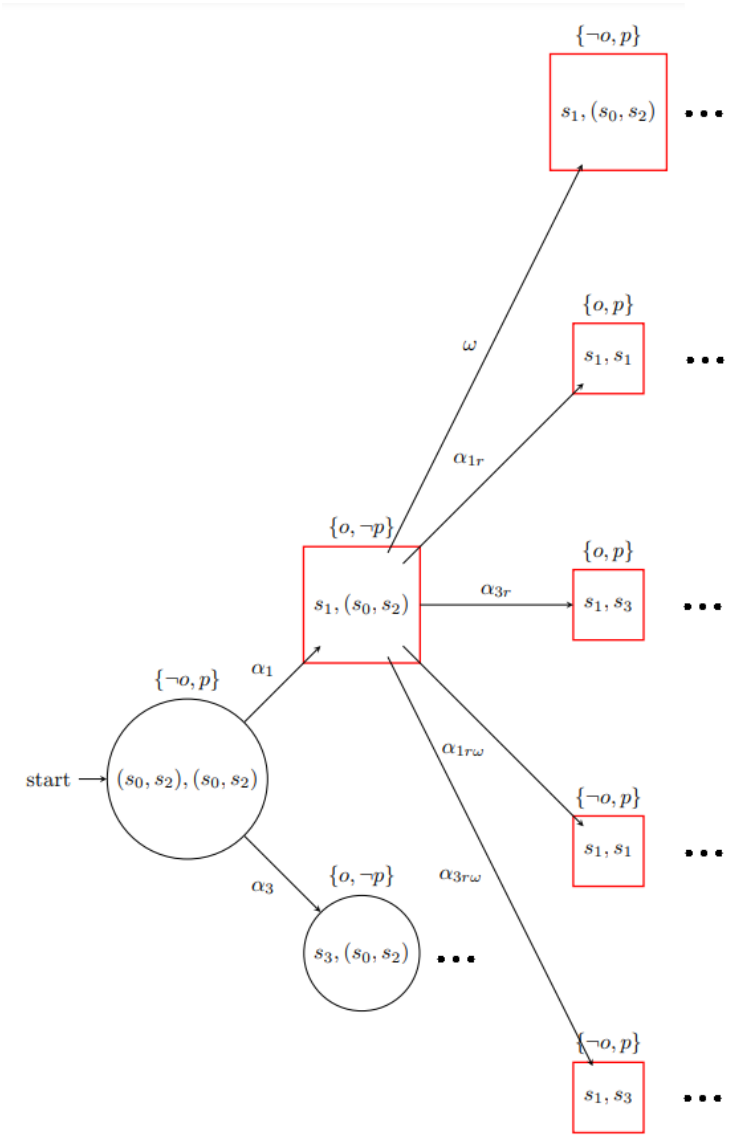

Figura 3.17: Sistema de transição aumentado para o sistema de transição rotulado apresentado na Figura 3.16

Em seguida utilizamos o SynthSMV para forçar a opacidade através da especificação LAC $AG(s_p \notin S_s \wedge EF(\neg o))$. Desta forma, não é possível para o ofuscador permitir que a visão pública do sistema reporte para usuários externos que o sistema TS_{nop} transitou por um estado secreto do sistema. Parte do resultado obtido para o ofuscador deste sistema, denominado de $TS_{nOfusc} = (S_{nOfusc}, Act_{nOfusc}, \rightarrow_{nOfusc}, I_{nOfusc}, AP_{nOfusc}, L_{nOfusc})$, é apresentado na figura 3.18.

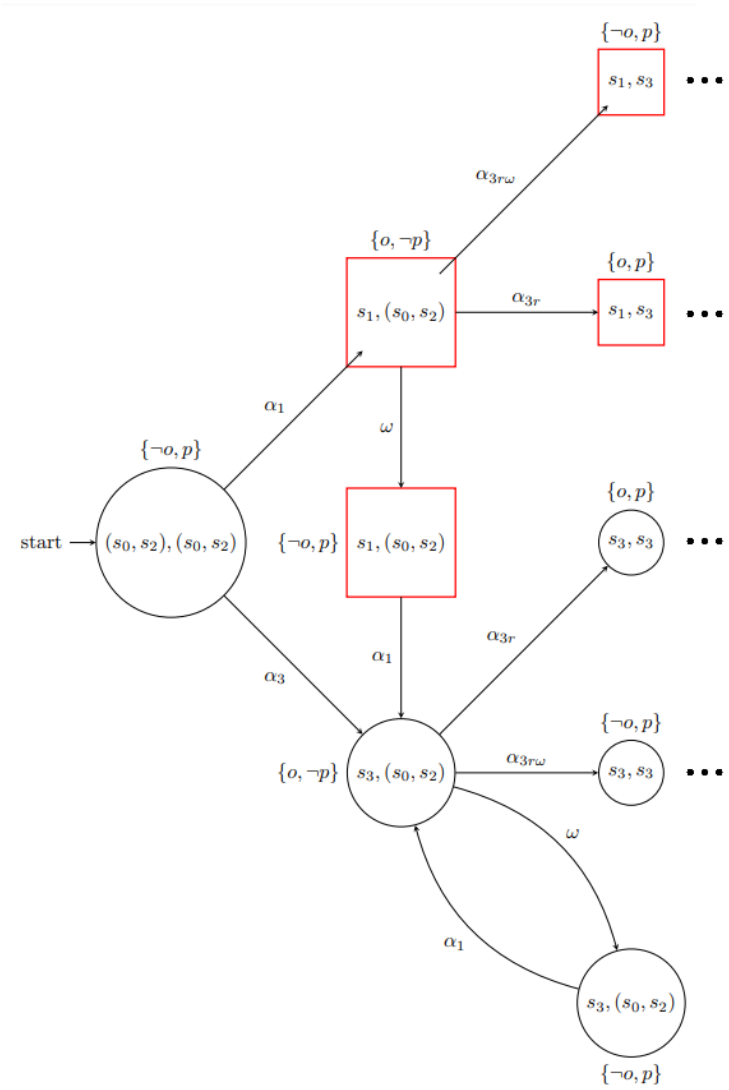

Figura 3.18: Ofuscador para o sistema de transição aumentado apresentado na Figura 3.17

.

Para simular a visualização da saída do ofuscador pelos agentes externos através do resultado da síntese, construímos o observador para o sistema de transição rotulado TS_{nOfusc}, apresentado na Figura 3.19, considerando apenas os eventos do conjunto $Act_r \cup Act_{r\omega}$ como observáveis, uma vez que são os eventos da visão pública

do sistema. Podemos observar que o ofuscador não permite que a visão pública do sistema reporte a ocorrência das transições α_{1r} e $\alpha_{1r\omega}$ no estado inicial, que levam para o estado secreto s_1 no sistema original. Porém, como existe outro um caminho não bloqueante em que o estado secreto não é alcançável, através da transição α_3, o ofuscador é capaz de garantir a opacidade do sistema.

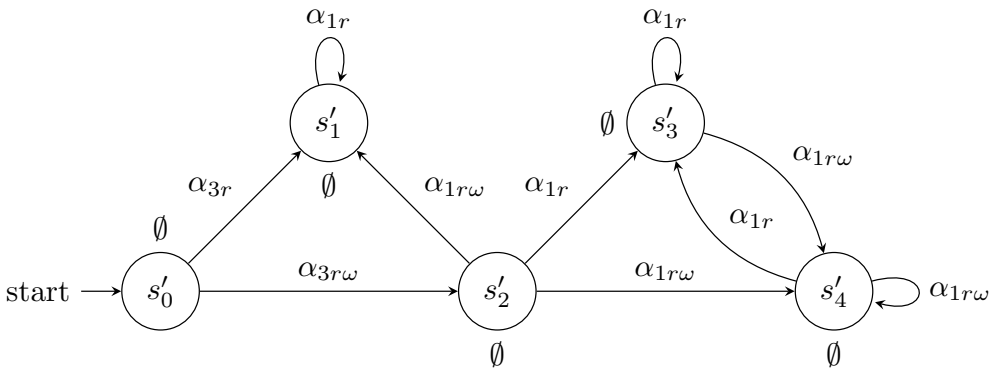

Figura 3.19: Observador do ofuscador apresentado na Figura 3.18

.

Por outro lado, também existem os sistemas que são impossíveis de serem ofuscados para garantir a opacidade e que seja possível preservar todo o comportamento para quem observa externamente, como é o caso do sistema de transição TS_{nop2}, uma variação do caso anterior, em que novamente o estado $s_1 \in S_{nop2}$ é secreto e os eventos α_1 e α_2 são os únicos observáveis, como ilustrado na 3.20.

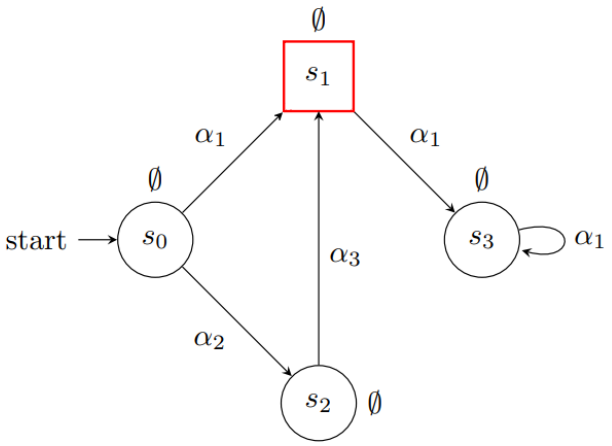

Figura 3.20: Exemplo de sistema em que não é possível forçar a opacidade

Percebemos que o sistema de transição rotulado TS_{nop2} não é opaco,

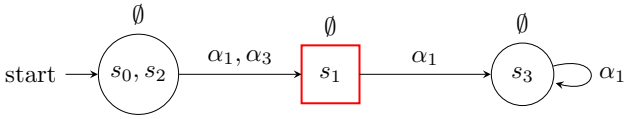

Figura 3.21: Observador do sistema de transição da Figura 3.20.

pois existe um estado no observador $TS_{nObs2} = (S_{nObs2}, Act_{nObs2}, \rightarrow_{nObs2}, I_{nObs2}, AP_{nObs2}, L_{nObs2})$, apresentado na Figura 3.21, que contém somente o estado secreto s_1.

Podemos então tentar forçar a opacidade como no caso anterior. Desta forma, primeiramente devemos criar o sistema sistema aumentado denominado $TS_{nAug2} = (S_{nAug2}, Act_{nAug2}, \rightarrow_{nAug2}, I_{nAug2}, AP_{nAug2}, L_{nAug2})$, parte deste sistema aumentado é apresentado na Figura 3.22.

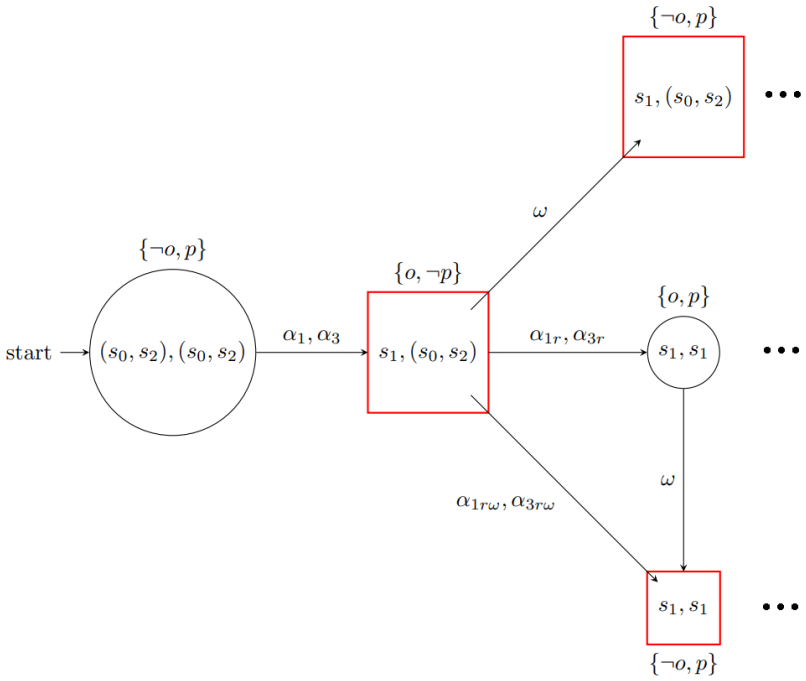

Figura 3.22: Sistema aumentado do sistema de transição rotulado da Figura 3.21.

Em seguida, utilizamos o SynthSMV para forçar a especificação $AG(s_p \notin S_s \wedge EF(\neg o))$ neste sistema aumentado e assim obtemos o ofuscador $TS_{nOfusc2} = (S_{nOfusc2}, Act_{nOfusc2}, \rightarrow_{nOfusc2}, I_{nOfusc2}, AP_{nOfusc2}, L_{nOfusc2})$, apresentado na Figura 3.23. A partir do resultado obtido percebemos que o sistema é forçado a ser opaco, porém o ofuscador é incapaz de manter o comportamento original do sistema através da visão pública, uma vez que é impossível para a visão pública sair do estado inicial

(s_0, s_2) através das transições α_{1r} e $\alpha_{1r\omega}$ sem violar a especificação do supervisor.

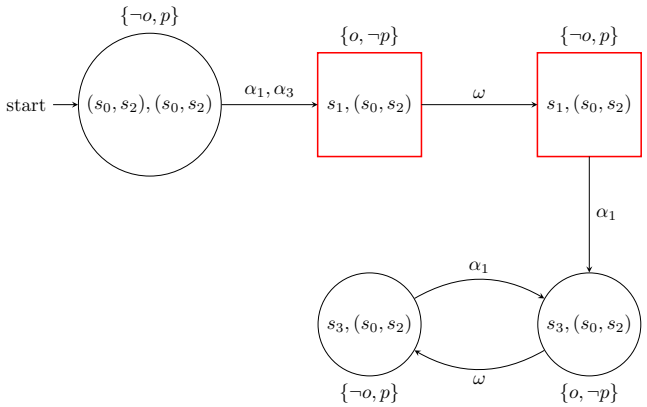

Figura 3.23: Ofuscador para garantir a opacidade no sistema de transição rotulado da Figura 3.21.

Portanto, utilizamos da abordagem proposta neste trabalho para variar o nível de privacidade em determinados estados do sistema aumentado afim de manter o mesmo comportamento do sistema original na visão pública do mesmo. Assim, utilizamos do SynthSMV, junto à especificação do em LAC $AG(((p \wedge (s \vee s_p) \in S_s) \rightarrow (D(s, s_p) \geq 1)) \wedge EF(\neg o))$ para realizar a ofuscação. Parte do resultado obtido é apresentado na Figura 3.24.

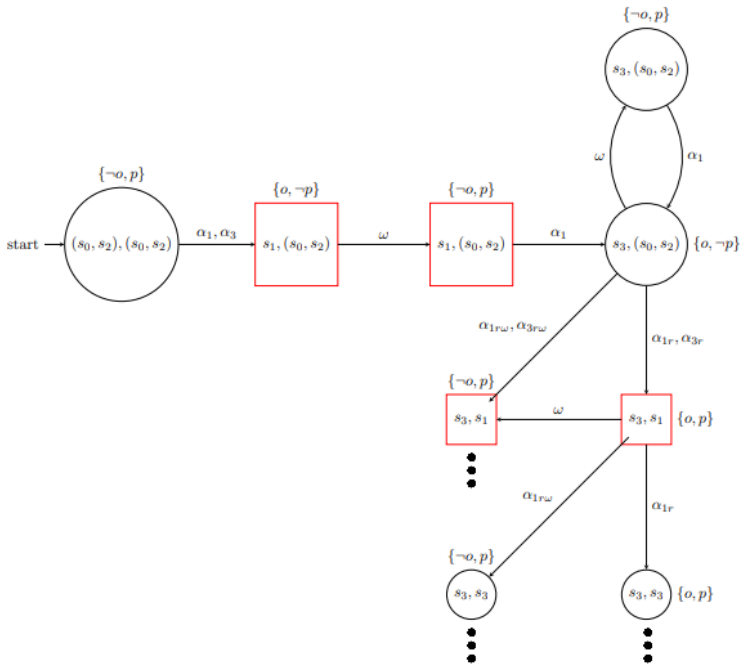

Figura 3.24: Ofuscador com diferentes níveis de privacidade para o sistema de transição rotulado da Figura 3.21.

Desta forma, além de manter a privacidade do sistema, uma vez que a visão pública somente poderá reportar um estado secreto caso o sistema original encontre-se em um estado não secreto, o ofuscador é capaz de manter o comportamento total do sistema. Essa característica pode ser verificada através do observador do ofuscador $TS_{nOfusc2}$, apresentado na Figura 3.25, considerando apenas os eventos do conjunto $Act_r \cup Act_{r\omega}$ como observáveis, uma vez que são os eventos da visão pública do sistema. Como os eventos α_{1r}, α_{3r} e $\alpha_{1r\omega}$, $\alpha_{3r\omega}$, representam a visualização dos eventos α_1 e α_3, respectivamente, pelos agentes externos verificamos que todo o comportamento do sistema de transição TS_{nop2} é preservado. Assim, além de aumentar o número de sistemas que podem ser ofuscador de forma não trivial, esta abordagem dificulta que usuários externos percebam a interferência do ofuscador a partir da visão pública do sistema por manter todo comportamento do sistema original.

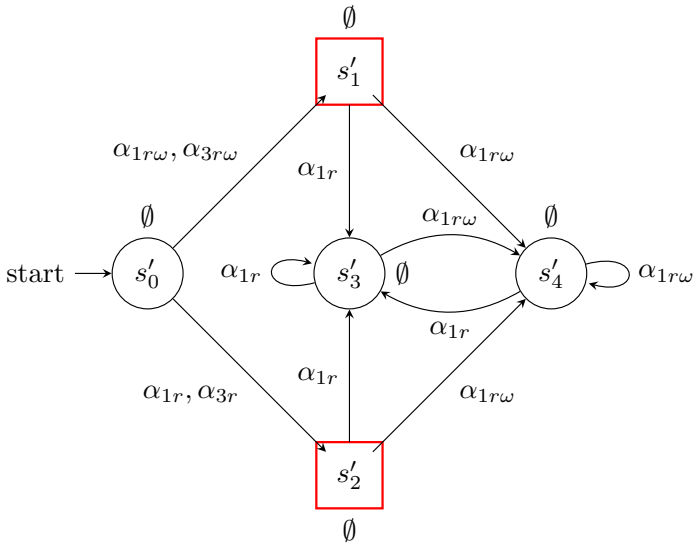

Figura 3.25: Observador do ofuscador da Figura 3.24

Além disso, podemos comparar a abordagem de ofuscação com diferentes níveis de privacidade proposta neste trabalho com o forçamento de opacidade em sistemas cíclicos através do resultado obtido no Exemplo 9. Considere o caso de forçar a opacidade através da ofuscação para o sistema de transição aumentado apresentado na Figura 3.11. Assim, utilizamos o sistema de transição aumentado TS_{dcAug} junto à especificação LAC $AG(s_p \notin S_s \wedge EF(\neg o))$ para obter o ofuscador com garantia de opacidade TS_{dcOpac}, que possui 90 estados e 302 transições. Observamos então, que a abordagem proposta neste trabalho restringe menos o comportamento do sistema aumentado TS_{dcAug}, pois mais estados serão alcançados, 106 no caso anterior, e

menos transições serão desabilitadas, 363 transições permaneceram habilitadas no caso anterior.

Por fim, para comparar a diferença no comportamento observado pelo intruso, na Figura 3.26 é apresentada parte do observador para o ofuscador com garantia de opacidade, TS_{dcOpac}, considerando apenas os eventos no conjunto $Act_r \cup Act_{r\omega}$ como observáveis. Assim, é possível notar através da comparação entre o observador obtido aqui com o observador obtido no Exemplo 9 que ao forçar a opacidade o intruso nunca será capaz de deduzir que o caminhão de distribuição passou pela cidade 5, ou seja as execuções $s_0as_4bs_5$ ou $s_0ds_3as_4bs_5$ nunca serão observadas, esse fato pode levá-lo a inferir que existe algum mecanismo para enganá-lo. Portanto, a abordagem proposta neste trabalho torna mais difícil a detecção da presença do ofuscador por algum intruso.

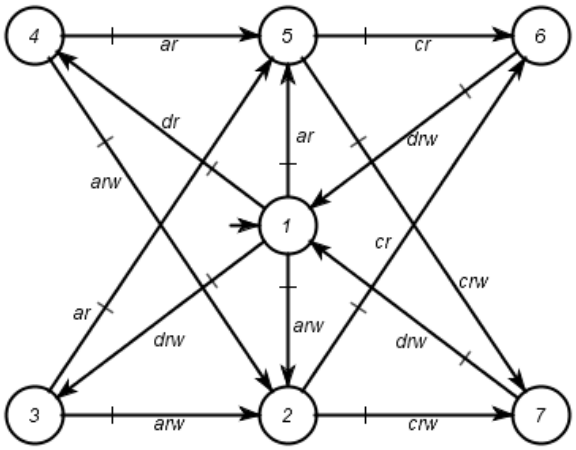

Figura 3.26: Parte do observador para o ofuscador com garantia de opacidade

Capítulo 4

Conclusões

Este trabalho apresenta uma nova abordagem de ofuscação que garante diferentes níveis de privacidade e utilidade para sistemas a eventos discretos utilizando controle supervisório baseado em lógica temporal. Pelo nosso conhecimento, este é primeiro trabalho a tratar de casos de ofuscação em que é possível reportar aos usuários externos que o sistema se encontra em um estado considerado secreto. A síntese do ofuscador é dividida em duas etapas, na primeira etapa é necessário criar um sistema aumentado e na segunda etapa utiliza-se do controle supervisório para limitar seu comportamento.

Para facilitar o procedimento de construção do sistema aumentado, na Seção 3.2 foi desenvolvido um algoritmo capaz de construir o sistema aumentado através da análise do comportamento concorrente entre o sistema original e sua visão pública utilizando a operação de *handshaking*. Este algoritmo permite construir os sistemas de transição aumentado de forma mais automatizada, o que permite trabalhar com sistemas maiores e diminui a chance de erro durante a construção.

Por sua vez, para solucionar o problema da síntese do ofuscador foi utilizado o procedimento proposto por RAWLINGS *et al.* [24] para construir o supervisor ótimo que, por sua vez, tem o papel de restringir o comportamento do sistema aumentado a partir de uma especificação do tipo CIR. Foi proposta uma nova especificação CIR na equação 3.2, esta especificação permite que, em determinadas situações, seja possível para o ofuscador reportar a evolução do sistema por um estado secreto. Desta forma é possível ofuscar uma variedade maior de sistemas, uma vez que problemas específicos podem exigir que um estado secreto seja reportado. Na Seção 3.3 apresentamos o Exemplo 8 da ferrovia simplificada que ilustra uma situação que a abordagem usual de ofuscação não seria capaz de garantir a privacidade e ao mesmo tempo manter a qualidade da informação gerada pois o ofuscador não consegue reportar que o trem não alcançou sua estação de destino. Além disso, ainda na Seção 3.3 a proposta de ofuscação deste trabalho é utilizada para ofuscar um sistema de distribuição de comida, classificado como sistema cíclico .

Além disso, permitir que o ofuscador reporte os estados secretos dificulta para os usuários mal intencionados a identificar a presença do ofuscador uma vez que toda a dinâmica do sistema será preservada. Para ilustrar essa situação, na Seção 3.4 mostramos a diferença de forçar a opacidade em sistemas a eventos discretos e a abordagem de ofuscação proposta neste trabalho. Desta forma, são apresentados exemplos que ilustram que o forçamento de opacidade modifica a visão pública do sistema e permite que um determinado invasor suspeite da presença do ofuscador.

4.0.1 Propostas de Trabalhos Futuros

Como primeira proposta de trabalho futuro, pretende-se buscar outros exemplos em que seja possível implementar esta nova abordagem que garante diferentes níveis de privacidade e utilidade. Além de investigar novos critérios de utilidade diferente da distância física ou número de transições entre estados.

Uma segunda linha de trabalho futuro busca considerar múltiplos conjuntos de estados secretos com diferentes requisitos de privacidade. Assim, podemos considerar a possibilidade de existir um conjunto de estados secretos que o ofuscador não poderá reportar através da visão pública, porém existe outro conjunto de estados que o ofuscador poderá reportar sob certas condições.

Referências Bibliográficas

[1] ALGULIYEV, R., IMAMVERDIYEV, Y., SUKHOSTAT, L., 2018, "Cyber-physical systems and their security issues", *Computers in Industry*, v. 100, pp. 212–223.

[2] ALVES, M. R., PENA, P. N., RUDIE, K., 2022, "Discrete-event systems subject to unknown sensor attacks", *Discrete Event Dynamic Systems*, v. 32, n. 1, pp. 143–158.

[3] BAIER, C., KATOEN, J.-P., 2008, *Principles of model checking*. MIT press.

[4] BARCELOS, R. J., BASILIO, J. C., 2021, "Enforcing current-state opacity through shuffle and deletions of event observations", *Automatica*, v. 133, pp. 109836.

[5] BASILIO, J. C., HADJICOSTIS, C. N., SU, R., et al., 2021, "Analysis and Control for Resilience of Discrete Event Systems: Fault Diagnosis, Opacity and Cyber Security", *Foundations and Trends® in Systems and Control*, v. 8, n. 4, pp. 285–443.

[6] CARVALHO, L. K., WU, Y.-C., KWONG, R., et al., 2018, "Detection and mitigation of classes of attacks in supervisory control systems", *Automatica*, v. 97, pp. 121–133.

[7] CASSANDRAS, C. G., LAFORTUNE, S., 2008, *Introduction to discrete event systems*. Springer.

[8] CIMATTI, A., CLARKE, E., GIUNCHIGLIA, E., et al., 2002, "Nusmv 2: An opensource tool for symbolic model checking". In: *International conference on computer aided verification*, pp. 359–364. Springer.

[9] DUCKHAM, M., KULIK, L., 2005, "A formal model of obfuscation and negotiation for location privacy". In: *International conference on pervasive computing*, pp. 152–170. Springer.

[10] EHLERS, R., LAFORTUNE, S., TRIPAKIS, S., et al., 2017, "Supervisory control and reactive synthesis: a comparative introduction", *Discrete Event Dynamic Systems*, v. 27, n. 2, pp. 209–260.

[11] FALCONE, Y., MARCHAND, H., 2010, *Various notions of opacity verified and enforced at runtime.* Tese de Doutorado, INRIA.

[12] FALCONE, Y., MARCHAND, H., 2015, "Enforcement and validation (at runtime) of various notions of opacity", *Discrete Event Dynamic Systems*, v. 25, n. 4, pp. 531–570.

[13] FRITZ, R., ZHANG, P., 2018, "Modeling and detection of cyber attacks on discrete event systems", *IFAC-PapersOnLine*, v. 51, n. 7, pp. 285–290.

[14] GILCHRIST, A., 2016, *Industry 4.0: the industrial internet of things.* Apress Berkeley, CA, Springer.

[15] GÓES, R. M., RAWLINGS, B. C., RECKER, N., et al., 2018, "Demonstration of indoor location privacy enforcement using obfuscation", *IFAC-PapersOnLine*, v. 51, n. 7, pp. 145–151.

[16] JACOB, R., LESAGE, J.-J., FAURE, J.-M., 2016, "Overview of discrete event systems opacity: Models, validation, and quantification", *Annual reviews in control*, v. 41, pp. 135–146.

[17] JI, Y., WU, Y.-C., LAFORTUNE, S., 2018, "Enforcement of opacity by public and private insertion functions", *Automatica*, v. 93, pp. 369–378.

[18] JI, Y., YIN, X., LAFORTUNE, S., 2019, "Enforcing opacity by insertion functions under multiple energy constraints", *Automatica*, v. 108, pp. 108476.

[19] LAFORTUNE, S., LIN, F., HADJICOSTIS, C. N., 2018, "On the history of diagnosability and opacity in discrete event systems", *Annual Reviews in Control*, v. 45, pp. 257–266.

[20] LIMA, P. M., ALVES, M. V. S., CARVALHO, L. K., et al., 2019, "Security against communication network attacks of cyber-physical systems", *Journal of Control, Automation and Electrical Systems*, v. 30, n. 1, pp. 125–135.

[21] LIMA, P. M., ALVES, M. V. S., CARVALHO, L. K., et al., 2021, "Security of Cyber-Physical Systems: Design of a Security Supervisor to Thwart Attacks", *IEEE Transactions on Automation Science and Engineering*, pp. 1–12. doi: 10.1109/TASE.2021.3076697.

[22] LIN, F., 2011, "Opacity of discrete event systems and its applications", *Automatica*, v. 47, n. 3, pp. 496–503.

[23] MEIRA-GÓES, R., LAFORTUNE, S., MARCHAND, H., 2021, "Synthesis of Supervisors Robust Against Sensor Deception Attacks", *IEEE Transactions on Automatic Control*, v. 66, n. 10, pp. 4990–4997. doi: 10.1109/TAC.2021.3051459.

[24] RAWLINGS, B. C., LAFORTUNE, S., YDSTIE, B. E., 2018, "Supervisory control of labeled transition systems subject to multiple reachability requirements via symbolic model checking", *IEEE Transactions on Control Systems Technology*, v. 28, n. 2, pp. 644–652.

[25] SABOORI, A., HADJICOSTIS, C. N., 2007, "Notions of security and opacity in discrete event systems". In: *2007 46th IEEE Conference on Decision and Control*, pp. 5056–5061. IEEE.

[26] TONG, Y., LI, Z., SEATZU, C., et al., 2018, "Current-state opacity enforcement in discrete event systems under incomparable observations", *Discrete Event Dynamic Systems*, v. 28, n. 2, pp. 161–182.

[27] WONHAM, W. M., CAI, K., OTHERS, 2019, "Supervisory control of discrete-event systems", .

[28] WU, Y.-C., LAFORTUNE, S., 2013, "Comparative analysis of related notions of opacity in centralized and coordinated architectures", *Discrete Event Dynamic Systems*, v. 23, n. 3, pp. 307–339.

[29] WU, Y.-C., LAFORTUNE, S., 2014, "Synthesis of insertion functions for enforcement of opacity security properties", *Automatica*, v. 50, n. 5, pp. 1336–1348.

[30] WU, Y.-C., RAMAN, V., RAWLINGS, B. C., et al., 2018, "Synthesis of obfuscation policies to ensure privacy and utility", *Journal of Automated Reasoning*, v. 60, n. 1, pp. 107–131.