VERIFICAÇÃO FORMAL DE SISTEMAS INSTRUMENTADOS DE SEGURANÇA NA INDÚSTRIA DE PETRÓLEO E GÁS NATURAL

LUIZ PAULO ENADIO DOS REIS[*], MAX HERING DE QUEIROZ[*], JEAN-MARIE FARINES[*], MARCELO LOPES DE LIMA[†], MARIO CESAR MELLO MASSA DE CAMPOS[†]

[*]*Departamento de Automação e Sistemas Universidade Federal de Santa Catarina Florianópolis, Brasil*

[†]*CENPES, Petrobras Rio de Janeiro, Brasil*

Emails: lpa.reis@gmail.com, max.queiroz@ufsc.br, j.m.farines@ufsc.br, marceloll@petrobras.com.br, mariocampos@petrobras.com.br

Abstract— This work presents an automated method for formal verification of PLC programs integrated to the development methodology of Safety Instrumented Systems (SIS) in oil and gas industry. The safety specifications relating inputs and outputs are standardized by Cause and Effect Matrices (CEM), which are translated into LTL formulas. This work presents rules for translating PLC programs in Ladder diagrams to a formal model intermediate language called FIACRE that make the model checking through the TINA/SELT verification chain. To handle the complexity of real SIS formal models, it is presented a method based on cone of influence to breakdown the model checking process and simplify FIACRE models. The resulting counterexamples are converted in signal diagrams or in commands for the PLC simulator to make analysis easier. The method was applied to verify an illustrative case and the PLC code of a real offshore oil platform.

Keywords— PLC, SIS, model checking, formal verification, Cause and Effect Matrix, Ladder Diagram.

Resumo— Este trabalho apresenta um método automatizável para verificação formal de programas de CLP integrado à metodologia de desenvolvimento de Sistemas Instrumentados de Segurança (SIS) na indústria de petróleo e gás. As especificações de segurança relacionando os diversos sensores e atuadores de SIS são padronizadas em Matrizes de Causa e Efeito (MCE), que são sistematicamente traduzidas em fórmulas LTL. São apresentadas regras para tradução do programa de CLP em diagrama *Ladder* para um modelo formal em linguagem intermediária FIACRE que serve de entrada para realização de *model checking* através da cadeia de verificação TINA/SELT. A fim de lidar com a complexidade dos modelos formais para SIS reais, apresenta-se um método baseado em cone de influência para decomposição do processo de *model checking* e simplificação dos modelos em FIACRE. Os contraexemplos resultantes são convertidos em diagramas de sinal ou em comandos para o simulador do CLP, que facilitam a interpretação das falhas identificadas. O método foi aplicado para verificação de um caso ilustrativo e do código do SIS de uma plataforma de petróleo *offshore* real.

Palavras-chave— CLP, SIS, *model checking*, Verificação formal, Matriz Causa e Efeito, Diagrama Ladder.

1 INTRODUÇÃO

A Indústria de Petróleo e Gás (IP&G) trabalha com operações críticas cujas falhas podem acarretar em perigo a vidas humanas, desastres ambientais e prejuízos à produção, ao patrimônio e à imagem da empresa. Para evitar acidentes, são implementados Sistemas Instrumentados de Segurança (SIS), compostos por sensores, atuadores e Controlador Lógico Programável (CLP), cuja responsabilidade é levar a planta a um estado seguro sempre que for detectada alguma situação de risco no processo. O desenvolvimento de programas de CLP para SIS segue uma metodologia rigorosa baseada em normas para especificação, implementação e testes. No entanto, testes podem ser realizados apenas para uma quantidade limitada de cenários que ajudam a encontrar erros, mas não asseguram exaustivamente a conformidade do sistema.

Dentre os vários métodos para validação de CLPs (Gergely et al., 2011), a verificação formal por *model checking* permite a validação automática e exaustiva a partir da modelagem matemática do sistema e da formalização dos requisitos em lógica temporal, retornando contra-exemplos em casos de falha que podem ajudar na identificação do erro (Clarke et al., 1999). Ovatman et al. (2016) apresentam uma comparação entre os diversos trabalhos na literatura sobre verificação por *model checking* de programas de CLP. Esses trabalhos diferem pelo domínio de aplicação, pela linguagem programação de CLP, pelos métodos para tratar a complexidade do modelo e pelas ferramentas de *model checking*. O presente trabalho apresenta um método para desenvolvimento de SIS assistido por *model checking* que se adeque às linguagens de programação e aos documentos de especificação adotados pelos engenheiros da IP&G.

Em (Farines et al., 2011), com o intuito de realizar verificação formal em linguagem de usuário, foi adotada uma abordagem baseada em Engenharia Dirigida a Modelos (MDE) para transformar programas de CLP que controlam sistemas pneumáticos na linguagem intermediária FIACRE. A MDE garante o rigor nas transformações. A linguagem FIACRE (Farail et al., 2008) é uma linguagem intermediária que permite utilizar várias linguagens de entrada e utiliza diversas ferramentas de verificação. A partir do modelo em FIACRE e de fórmulas LTL para as propriedades desejadas, realiza-se model checking através da cadeia de verificação TINA/SELT (Berthomieu and Vernadat, 2006) .

Baseada na cadeia de verificação proposta em (Farines et al., 2011) e tendo em vista que a verificação formal é um método exaustivo e automático, o presente trabalho construiu uma cadeia de verificação com intuito de verificar os programas de CLP que

controlam os SIS da IP&G. Esta dissertação evolui o trabalho citado a partir do ponto de que para diminuir o número de estados do modelo formal, as regras para tradução do diagrama Ladder para linguagem FIACRE foram aprimoradas. Outro fator importante é que o método proposto neste trabalho é focado nos SIS que protegem sistemas de produção da IP&G; à vista disso, foi possível explorar as características conhecidas do sistema e adicionar a cadeia de verificação, uma etapa de redução de sistema através do método Cone de Influência.

Outro fator que torna esta técnica muito assertiva para aplicação direcionada a IP&G deriva do fato que as MCEs nestas indústrias são construídas de acordo com norma (ET-3000.00-1200-800-PGT-006, 2000). Posto isto a partir do conhecimento da norma, foi possível criar uma ferramenta para geração automática das propriedades de segurança na forma de fórmulas temporais do tipo LTL a partir das Matrizes de Causa e Efeito. Diferente das outra ferramentas que traduzem as propriedades da MCE para lógica temporal, este trabalho gera as regras na forma de LTL. Outra vantagem observada na ferramenta de tradução das propriedades desta dissertação advêm do ao fato que devido a norma (ET-3000.00-1200-800-PGT-006, 2000), as *TAGS* das entradas são todas padronizadas sendo assim, automaticamente o nome das entradas no modelo formal nas propriedades é igual. Outra evolução apresentada foi a criação de uma ferramenta que reorganiza o contraexemplo em um diagrama de sinais, para que estes fique(m) mais entendíveis a engenheiros não acostumados a trabalhar com métodos formais.

A Seção 2 apresenta uma visão geral da metodologia de desenvolvimento de SIS assistida por *model checking*. Na seção 3 são apresentadas as regras de transformação de LD para FIACRE, as regras para representação das propriedades de segurança em LTL e a estratégia de redução de modelo. Na seção 4 é explicada a cadeia de verificação formal. Um estudo de caso e resultados são apresentados na seção 5 e a seção 6 finaliza o artigo com uma breve conclusão.

2 DESENVOLVIMENTO DE SIS ASSISTIDO POR VERIFICAÇÃO FORMAL

Os projetos de automação na IP&G de petróleo são desenvolvidos por múltiplas equipes responsáveis pela criação de uma sequência de documentos padronizados. O fluxograma de processos especifica as definições básicas da planta, incluindo a instrumentação do SIS. A partir deste documento são criados a Matriz Causa e Efeito (MCE), que especifica as relações lógicas entre entradas e saídas do sistema, e o memorial descritivo, com as demais especificações funcionais da planta. Esses são os principais documentos de referência para especificação e implementação do programa do CLP, que é verificado através de simulação e testes não exaustivos, devendo ser corrigido caso seja alguma inconformidade seja diagnosticada. Uma simplificação da atual metodologia é representada na parte não destacada da figura 1.

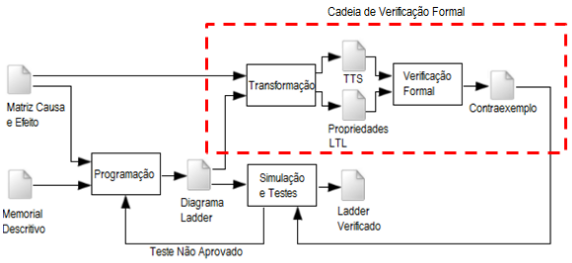

Figura 1: Metodologia de verificação formal proposta

O presente artigo propõe um método de verificação formal que se integra à metodologia vigente, adicionando uma fase de verificação formal por *model checking* complementar à etapa de simulação e teste, para verificação exaustiva da conformidade entre o código de CLP e as propriedades de segurança da MCE. Para realizar o *model checking*, é necessário que o código em linguagem LD seja traduzido para um modelo formal e as propriedades sejam descritas por fórmulas de lógica temporal. Para que a verificação formal possa ser realizada a nível de usuário, foi criada uma cadeia de verificação formal onde o código LD e as propriedades de segurança expressas na Matriz de Causa e Efeito são transformados em Sistema de Transição Temporizada (TTS) e lógica temporal do tipo LTL respectivamente. A próxima etapa da cadeia é a verificação formal, onde as ferramentas TINA/SELT (Berthomieu and Vernadat, 2006) são usadas para realizar *model checking*. Em caso de inconformidade, um contraexemplo é apresentado na forma de um diagrama temporal, que pode ser visualizado pelo programador ou executado automaticamente no simulador para identificação do erro. O método proposto corresponde à região pontilhada da figura 1.

3 ETAPA DE TRANSFORMAÇÃO

A primeira etapa desta cadeia de verificação formal é a etapa de transformação, ilustrada na figura 2, na qual os programas e as especificações em linguagens de usuários sofrem sucessivas traduções para serem modelados em linguagens de entrada para a ferramenta de *model checking*. Para transformação do código LD em modelo formal utiliza-se a linguagem intermediária FIACRE (Farail et al., 2008), que permite representar tanto os aspectos comportamentais como temporais do sistema usando elementos de linguagem de alto nível. O modelo em FIACRE pode ser traduzido automaticamente para um modelo de baixo nível através da ferramenta FRAC, cujo processo de compilação inclui uma fase de otimização de modelo que ajuda a reduzir o tamanho do espaço de estados para *model cheching*. A subseção 3.1 explica as regras para tradução de LD para FIACRE e a subseção 3.2 apresenta o método para representação das propriedades de segurança expressas na MCE em lógica temporal do tipo

LTL.

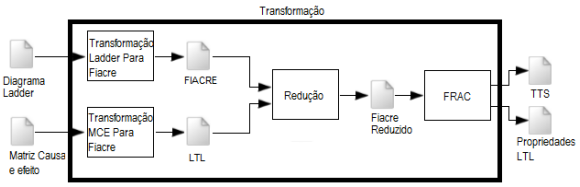

Figura 2: Etapa de Transformação

Para evitar explosão combinacional, a etapa de transformação inclui um método preliminar ao FRAC para reduzir o modelo de alto nível em FIACRE explorando a estrutura de acoplamentos na Matriz de Causa e Efeito. A subseção 3.3 apresenta o método de redução proposto, baseado em cone de influência (Clarke et al., 1999), que basicamente identifica qual parte do LD e quais entradas são relevantes para o *model checking* das propriedades de cada saída do CLP, de forma que as partes desnecessárias podem ser abstraídas do modelo sem afetar o resultado.

3.1 Transformação de LD para FIACRE

Dentre as 5 linguagens definidas pela norma IEC 61131-3 (2003), este trabalho foca na linguagem LD por ser muito usada para programação dos CLPs que controlam os SIS na IP&G. No entanto, os métodos apresentados nesta seção podem ser adaptados para as demais linguagens. A estrutura de um programa em LD baseia-se no conceito de circuitos de relés em que as saídas são ativadas por bobinas em função estado de contatos associados a entradas ou memórias. Os programas em LD são estruturados em uma sequência de *rungs* (degraus), cada qual definindo a lógica de ativação de uma bobina em função da composição de contatos em série ou em paralelo. Funções mais complexas como contadores e temporizadores, representadas como blocos de função são também utilizadas. A Figura 3 contém um programa escrito em LD que será utilizado para exemplificar todas as etapas da cadeia de verificação formal.

A linguagem FIACRE (Formato Intermediário para Arquiteturas de Componentes Distribuídos Embarcados) incorpora duas noções sintáticas básicas: processos *(process)* que descrevem o comportamento do sistema modelado nessa linguagem e componentes *(component)* que constroem o sistema como uma composição hierárquica de processos. O comportamento do *process* é definido por um conjunto de estados de controle e um conjunto de transições. As transições são construídas a partir de construções determinísticas disponíveis na linguagem de programação clássica (*assignments, conditionals, while loops, sequential compositions, ...*), construções não deterministas (escolhas e tarefas não deterministas) e um conjunto de portas de comunicação (Farail et al., 2008). Um *component* é definido como a composição paralela de *component* e/ou *process* que se comunicam por meio de portas e ou variáveis compartilhadas. A sintaxe dos

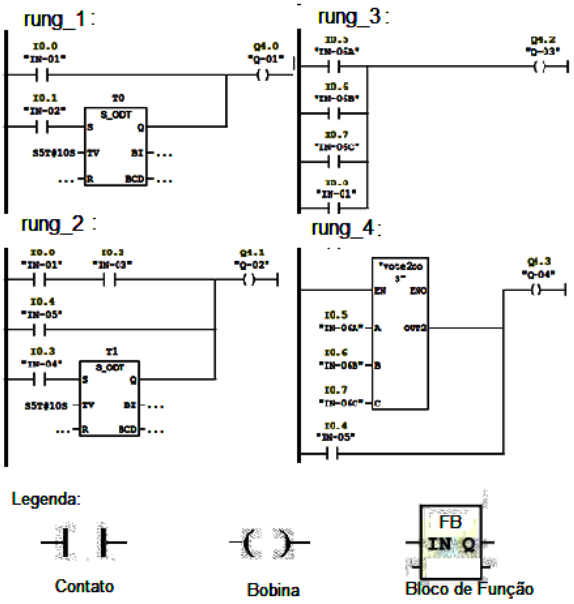

Figura 3: Exemplo de programa em LD

component permite restringir o modo de acesso e a visibilidade das variáveis e portas de comunicação, associar restrições de tempo às comunicações nas portas e definir prioridade entre os eventos de comunicação (Farail et al., 2008).

Tendo em vista essas características, o modelo que representa o comportamento do CLP é estruturado em FIACRE por intermédio do *component PLC*, que define todas as portas, hierarquias e restrições temporais e restrições de comunicação, além de especificar a composição paralela entre o *process scan_cycle*, que modela do ciclo de execução do CLP de acordo com a sequência de *rungs* do código LD principal, e os processos dos blocos de função temporizados que são definidos no código LD. Os blocos de função sem temporização são implementados como funções globais em FIACRE. A Figura 3.1 ilustra o *component PLC* para o exemplo de LD da Figura 3.

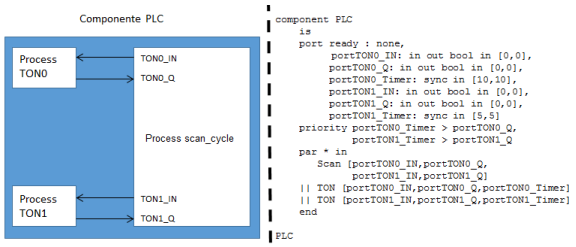

Figura 4: *Component PLC* em FIACRE do programa em LD ilustrado na Figura 3

A modelagem dos *rungs* dentro do *process scan_cycle* é realizada de três formas distintas, dependendo da presença e do tipo de bloco de função. De acordo com as regras de transformação definidas por Farines et al. (2011), quando um *rung* contém apenas elementos básicos de LD, como bobinas e contatores,

a representação em FIACRE é uma operação de atribuição simples em que a variável que é ativada pela bobina recebe o resultado da combinação lógica das variáveis associadas aos contatos, de acordo com a estrutura de ligações em série (E) ou em paralelo (OU) no *rung*. Por exemplo, o *rung_3* da Figura 3 pode ser modelado por:

```
Q_03 := IN_06A or IN_06B or IN_06C or IN_01;
```

Quando um *rung* contém um bloco de função sem temporização, os blocos de função são programados como funções globais do modelo em FIACRE e no *process scan_cycle* a variável de saída recebe o resultado da função global aplicada sobre a lógica de entrada. Por exemplo, o *rung_4* da Figura 3 pode ser modelado por:

```
Q_04 := vote2oo3(IN_06A, IN_06B, IN_06C) or IN_05;
```

onde *vote2oo3* é uma função global definida em FIACRE de acordo com a lógica programada em LD no bloco de função correspondente.

Já quando o *rung* possui um bloco de função temporizado, é necessário criar uma instância de um processo concorrente ao *process scan_cycle*. de Souza et al. (2010) apresentam modelos FIACRE para os principais blocos temporizados, como TON e TOF. Como apenas uma comunicação por transição é permitida em FIACRE, a modelagem do *rung* implica a criação de um estado auxiliar no *process scan_cycle* para realizar a sincronização com o processo que modela o bloco de função temporizado. Por exemplo, a tradução para FIACRE do rung_2 da Figura 3 corresponderia ao seguinte trecho do *process scan_cycle*:

```
from State1_W
   portTON1_IN! IN_04;
   to State1_R
from State1_R
   portTON1_Q? TON1_Q;
   Q_02 := (I_01 and I_03) or I_05 or TON1_Q;
   to State2_W
```

O *process scan_cycle* representa o ciclo de execução do código LD principal, a partir do qual é executada toda a lógica de intertravamentos do CLP. O modelo em FIACRE segue o comportamento do ciclo de varredura de CLP, que: inicia pela atualização dos valores das variáveis de entrada de acordo com o estado do sinal na entrada física correspondente; atualiza sequencialmente as variáveis de memória e de saída de cada *rung* de acordo com o código LD; atualiza o valor das saídas físicas conforme o estado das variáveis de saída ao final do ciclo; e reinicia o ciclo após decorrido o tempo de varredura. A implementação do ciclo de varredura em FIACRE corresponde a uma máquina de estados cíclica em que as lógicas dos rungs são executadas sequencialmente nas transições de modo que um novo par de estados e transições de leitura e escrita é acrescentado a cada bloco de função temporizado no LD, conforme as regras definidas acima. Considera-se que todas as transições são instantâneas, a menos da transição de reinício do ciclo cuja duração é especificada para corresponder ao tempo de varredura do CLP (comando *wait*[t,t]).

Como as especificações de segurança para SIS devem ser atendidas em qualquer situação da planta, o modelo formal deve prever todas as possibilidades de valores de entrada a cada reinício do ciclo de varredura. Assim, a transição do último estado Writing para o estado inicial atribui o valor *"any"* para cada variável de entrada. Desta forma, o modelo formal para *model checking* leva em conta todos os valores possíveis de entrada a cada novo ciclo de varredura.

A Figura 5 apresenta uma visão simplificada do comportamento do *process scan_cycle* para o LD da Figura 3.

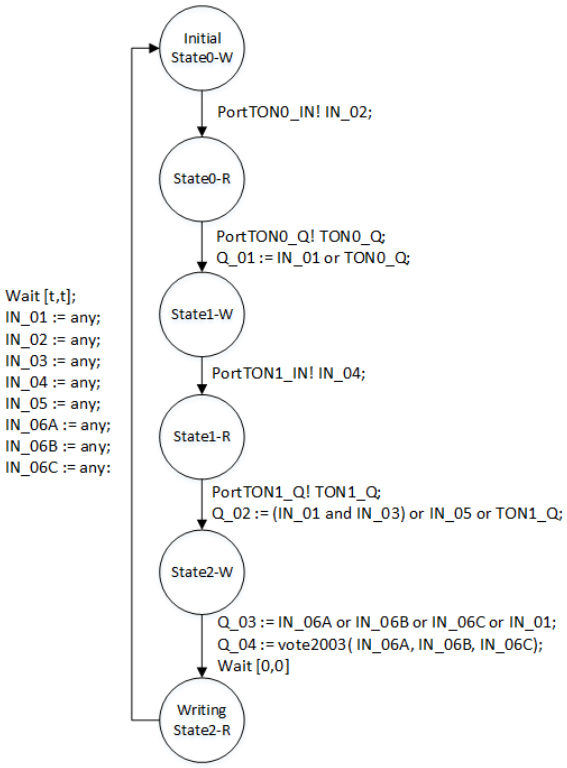

Figura 5: Visão simplificada do *process scan_cycle* para o LD da Figura 3

3.2 Transformação de MCE para LTL

A Matriz Causa e Efeito (MCE) define a semântica que representa a lógica de dependências entre os sensores (causas) e atuadores (efeitos) de SIS. A norma ET-3000.00-1200-800-PGT-006 (2000) especifica o padrão para elaboração de MCE em projetos de instalações marítimas da Petrobras. Os símbolos usados como entradas de MCE são:

- X: As causas nas linhas relacionadas são combinadas pelo operador lógico *OR* para ativar o efeito na coluna correspondente.
- A*i*: Para i = 1,2,..., as causas relacionadas pela entrada A*i* são combinadas pelo operador *AND* para ativar o efeito na coluna correspondente.
- T*x*: As causas relacionadas devem se manter ativas por x segundos para ativar o efeito.

Tabela 1: Exemplo de Matriz Causa e Efeito

		EFFECT: TAG	Q_01	Q_02	Q_03	Q_04
CAUSE: Equip.	Voting	Equip.	Output 01	Output 02	Output 03	Output 04
Input 01		IN_01	**X**	**A1**	**X**	
Input 02		IN_02	**T10**			
Input 03		IN_03		**A1**		
Input 04		IN_04		**T10**		
Input 05		IN_05		**X**		**X**
Input 06A	1oo3	IN_06A			**X**	
Input 06B	2oo3	IN_06B				**X**
Input 06C		IN_06C				

- X*oo*Y: A linha na MCE representa a votação de X elementos do grupo de Y causas.

Por exemplo, a MCE da Tabela 1 contém as especificações de segurança para o SIS cujo LD é apresentado na Figura 3. A segunda coluna da MCE especifica que a saída Q_02 do CLP deve ser ativada quando forem sensibilizadas as entradas IN_01 e IN_03 simultaneamente, ou IN_04 por 10 segundos, ou IN_05.

Cada saída do SIS pode sofrer dois tipos de falha, dependendo da sua ativação. Uma Falha Perigosa (*DF - dangerous failure*) acontece quando é atendida a lógica especificada entre as causas, mas a saída do CLP não é ativada. Uma Falha Segura (*SF - safe failure*) acontece quando a saída é ativada sem que seja atendida a lógica especificada entre as causas. Assim, para verificar por *model checking* a ausência de falhas no LD, são definidas duas propriedades em LTL para cada efeito da MCE, que especificam que a saída é livre de Falha Perigosa (*DFF - dangerous failure free*) e livre de Falha Segura (*SFF - safe failure free*). Estas propriedades podem ser extraídas sistematicamente da MCE através das seguintes fórmulas LTL:

$$DFF_{effect} : \Box\neg(probe \wedge cause \wedge \neg effect)$$

$$SFF_{effect} : \Box\neg(probe \wedge \neg cause \wedge effect)$$

onde:

- *effect* é a variável de saída a ser verificada;
- *cause* é uma expressão LTL sobre as entradas da MCE correspondente às relações booleanas expressas na coluna, conforme a semântica definida pela ET-3000.00-1200-800-PGT-006;
- *probe* corresponde à ativação do último estado do ciclo de varredura (estado *Writing* do *Process scan_cycle*), quando as saídas físicas do CLP são atualizadas em função dos valores das respectivas variáveis de saída.

Por exemplo, a terceira coluna da MCE da Tabela 1 é traduzida nas seguintes fórmulas LTL:

$$DFF_{\text{Q\_03}} : \Box\neg((state\ Writing) \wedge (\text{IN\_01} \vee \text{IN\_06A} \vee \text{IN\_06B} \vee \text{IN\_06C}) \wedge \neg\text{Q\_03})$$

$$SFF_{\text{Q\_03}} : \Box\neg((state\ Writing) \wedge \neg(\text{IN\_01} \vee \text{IN\_06A} \vee \text{IN\_06B} \vee \text{IN\_06C}) \wedge \text{Q\_03})$$

Quando a coluna de uma MCE especifica uma uma entrada com o símbolo T*x*, as propriedades de *DFF* e *SFF* devem considerar uma temporização de x segundos nas causas correspondentes. Entretanto, na lógica temporal LTL o tempo é representado de forma implícita, ao contraponto que na MCE o tempo é representado de forma explícita. Tendo em vista este problema, propõe-se construir um observador temporal conforme o modelo da Figura 6 para cada causa temporizada da MCE, de modo que as propriedades relacionadas possam ser expressas em lógica LTL em função do estado do observador anexo ao modelo FIACRE do CLP.

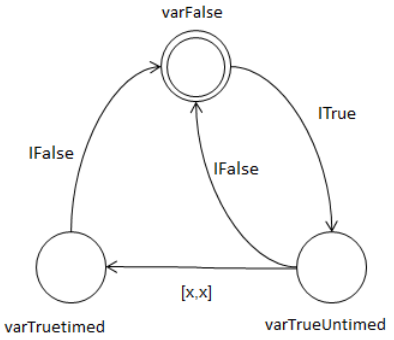

Figura 6: Contador temporal para entradas T*x* na MCE

O contador temporal conta o tempo de forma explícita para permitir representar em LTL as especificações de uma entrada relacionada pelo símbolo T*x* na MCE. Seu comportamento é determinado da seguinte maneira: o contador inicia e se mantém no estado inicial *"varFalse"* enquanto a variável da causa relacionada for falsa. Quando a entrada correspondente for sensibilizada, a transição *"ITrue"* é ativada e o contador muda para o estado *"VarTrueUntimed"*. Quando o contador entra neste estado, é iniciada a contagem dos *x* segundos especificado na MCE. Se a causa for dessensibilizada antes do tempo *x* a transição *"IFalse"* é ativada e o contador retorna ao estado inicial. Porém, se a causa for mantida ativa por *x* segundos, o contador se atualiza para o estado *"VarTrueTimed"* e nele permanece enquanto a causa estiver ativa.

Assim, uma especificação sobre uma entrada temporizada na MCE pode ser representada em LTL em função do estado *varTrueTimed* do respectivo contador temporal no modelo FIACRE. Por exemplo, a segunda coluna da Tabela 1 especifica que IN_04 deve se manter sensibilizado por 10 segundos para ativar o efeito Q_02. Neste caso, cria-se uma instância do contador para a entrada IN_04 (*process counterIN_04*) em composição com o *component PLC* do código LD.

Com isso, as propriedades DFF e SFF para o efeito Q_02 podem ser expressas em LTL como:

$$DFF_{\text{Q\_02}} : \Box\neg((state\ Writing) \wedge ((IN\_01 \wedge IN\_03) \vee counterIN\_04\_state\_VarTrueTimed \vee IN\_05) \wedge \neg \text{Q\_02})$$

$$SFF_{\text{Q\_02}} : \Box\neg((state\ Writing) \wedge \neg((IN\_01 \wedge IN\_03) \vee counterIN\_04\_state\_VarTrueTimed \vee IN\_05) \wedge \text{Q\_02})$$

3.3 Redução por Cone de Influência

Sistemas instrumentados de segurança são sistemas grandes contendo centenas de entradas e saídas. Desta forma aplicar *model checking* sobre um modelo formal que leve em conta as combinações de entradas de um SIS é inviávelxsem a utilização de técnicas de redução devido ao explosão combinacional de estado no TTS. Cone de influência (COI) (Clarke et al., 1999; Adiego et al., 2015) é uma técnica de redução, cuja principal ideia é identificar qual parte do modelo completo é relevante para a avaliação da propriedade verificada. As partes que não têm influência sobre a propriedade a ser verificada podem ser abstraídas do modelo sem afetar o resultado da verificação.

O primeiro passo para reduzir o número de estados no modelo é decompor o *model checking* para verificar as propriedades de cada saída separadamente, de modo que seja gerado um modelo reduzido por COI para cada saída do SIS, cada qual considerando a apenas as variáveis que influenciam nas propriedades *DFF* e *SFF* para a respectiva saída. Assim, o método de cone de influência é aplicado sobre o modelo FIACRE completo gerando um modelo reduzido para cada saída a pelo algorítimo seguir:

1. Identificar no modelo FIACRE recursivamente o conjunto de variáveis relevantes, que contém a variável de saída e todas aquelas que influenciam diretamente nos elementos do conjunto de variáveis relevantes ;

2. Remover do modelo FIACRE as variáveis que não forem relevantes;

3. Abstrair as transições que não alteram o valor de variáveis relevantes;

4. No reinício do ciclo de varredura, associar todos os valores possíveis ("any") apenas para as entradas que pertençam ao conjunto de variáveis relevantes ou às propriedades *DFF* e *SFF* da respectiva saída.

A Figura 7 ilustra o cone de influência nas saídas Q_01 e Q_02 do modelo FIACRE para o código LD da Figura 3. Nela pode-se notar, por exemplo, que somente IN_01, IN_02, TON0_IN e TON0_Q influenciam no valor final da saída Q_01 pelo código LD. Não há variáveis adicionais nas propriedades DFF_{Q_01} e SFF_{Q_01} da MCE na Tabela 1. Desta forma, quando for gerar o arquivo reduzido para esta saída, todo o restante das variáveis contidas no código pode ser descartado sem afetar a verificação. Os traços que partem de cada saída da Figura 7 indicam o cone de influência.

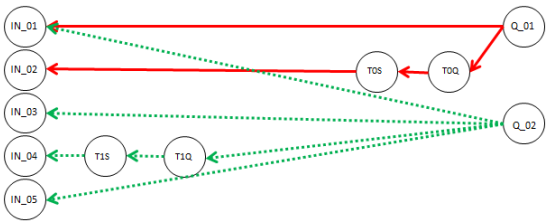

Figura 7: Cone de Influência

4 VERIFICAÇÃO FORMAL

Após a redução por COI na etapa de transformação, obtém-se um modelo FIACRE para cada saída do sistema, contendo tanto o modelo reduzido do código de CLP quanto as respectivas propriedades *DFF* e *SFF* com seus contadores temporais. Cada modelo FIACRE é então transformado de forma automática pelo compilador FRAC em um Sistema de Transição Temporizada (TTS) e na lógica LTL. A verificação formal do SIS compreende então o model checking de todas as propriedades LTL sobre seus respectivos modelos TTS reduzidos.

A Figura 8 ilustra o bloco de verificação formal. O TTS é transformado pelo programa TINA em uma estrutura de *Kripke*, um grafo utilizado na descrição de sistemas concorrentes, capaz de representar sistemas com comportamento infinito onde os vértices representam os estados e as arestas modelam mudanças de estado atômicas (Biere et al., 1999). Este grafo e as propriedades em LTL são as entradas do *model checker* SELT, que pertence ao ambiente TINA (Berthomieu and Vernadat, 2006). Caso as propriedades não sejam satisfeitas o sistema retorna um contraexemplo na forma de uma sequência de transições de estados da estrutura de Kripke, que levam à situação de falha.

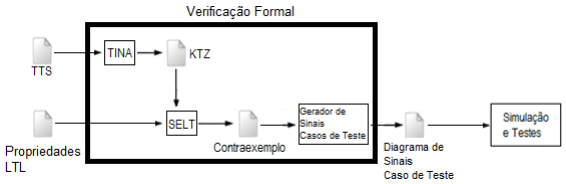

Figura 8: Etapa de Verificação Formal

Os contraexemplos gerados pelo SELT são de difícil compreensão pelo engenheiro que programa o CLP, pois não é simples a associação de transições de estado no modelo formal com a execução do LD no CLP. Com o intuito de realizar a verificação formal em linguagem de usuário de ponta a ponta do processo e tornar o método acessível a engenheiros não familiarizados com métodos formais, propõe-se que o con-

traexemplo fornecido pela ferramenta SELT passe por um processo de tradução, como indicado na Figura 9, para ser apresentado na forma de diagrama de sinais digitais ou na forma de caso de teste para a ferramenta de simulação.

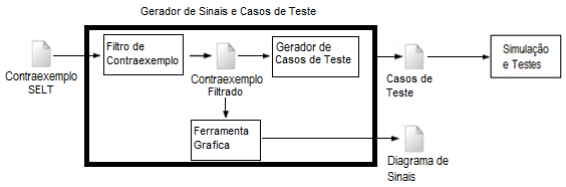

Figura 9: Processo de tradução do contraexemplo

A tradução do contraexemplo inicia por uma filtragem que remove do formato gerado pelo SELT todos os estados e eventos não relevantes, textos não relacionados ao contraexemplo, etc. O contraexemplo é organizado então para mostrar sequencialmente os estados das variáveis de saída e de entrada a cada ciclo de execução. O resultado é salvo num arquivo de texto que serve de entrada para diversas ferramentas gráficas, onde o engenheiro pode plotar o contraexemplo como um diagrama de sinal digital.

4.1 Exemplo

Para ilustrar o método de verificação formal, foram inseridos dois erros no LD da Figura 3. Como primeiro erro, o bloco temporizador no *rung_2* foi alterado de 10 para 5 segundos. O segundo erro foi inserido no *rung_4*, trocando-se a entrada C do bloco de função vote2oo3 de IN_06C para IN_06A.

O método proposto neste artigo foi aplicado então ao LD modificado considerando as propriedades da MCE da Tabela 1, utilizando um PC com 2.4 GHz e 4 GB de RAM e Linux Ubuntu 16.0d. As saídas foram verificadas com e sem redução por COI para fins de comparação. Os resultados são apresentados na Tabela 2, que apresenta o número de estados do TTS, o tempo de execução e o veredito do *model checking*: OK (se a propriedade foi atendida pela saída verificada), *Safe Failure*(SF) e *Dangerous Failure*(DF).

É possível observar na Tabela 2 que foram detectados os erros inseridos: SF em Q_02 e DF em Q_04. Não foram encontradas falhas nas saídas Q_01 e Q_03, logo ambas as saídas não possuem nenhuma das falhas analisadas na verificação formal. Os contraexemplos oriundos dos erros encontrados foram filtrados e os diagramas de sinais obtidos podem ser observados na Figura 10.

Na Tabela 2, também é possível observar que, sem a aplicação do COI, o número de estados no TTS foi consideravelmente alto, mesmo para um programa pequeno com 8 entradas e 4 saídas. Também é possível observar na Tabela 2 que aplicando a etapa de redução, as falhas também são detectadas, mas o número de estados do TTS é muito menor, no caso mais extremo, reduzindo de 28 000 estados para apenas 160.

Tabela 2: Resultados da aplicação da cadeia de verificação formal no exemplo ilustrativo

Resultados sem redução por COI			
Efeitos	Estados	Tempo de exec	SF/DF/OK
Q_01	7313	14 s	0/0/2
Q_02	21176	22 s	1/0/1
Q_03	28000	24 s	0/0/2
Q_04	28000	23 s	0/1/1
Resultados com redução por COI			
Efeitos	Estados	Tempo de exec	SF/DF/OK
Q_01	592	06 s	0/0/2
Q_02	1320	06 s	1/0/1
Q_03	160	06 s	0/0/2
Q_04	160	06 s	0/1/1

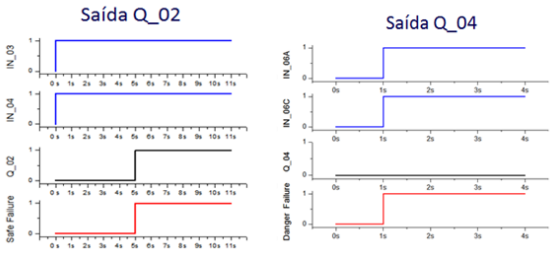

Figura 10: Contraexemplo na forma de diagrama de sinais para os efeitos Q_02 e Q_04

5 APLICAÇÃO A UMA PLATAFORMA OFFSHORE

O SIS de uma plataforma de produção de petróleo "*offshore*" da Petrobras serve como estudo de caso para aplicação do método proposto nesse trabalho. O sistema é composto por CLPs responsáveis por funções de segurança específicas as quais são divididas por subsistemas: *electrical subsystem* (ELE), *shutdown subsystem* (ESD), *fire and gas subsystem* (F&G), *control subsystem* (CON), *turret subsystem* (TUR), e *vessel subsystem* (VES). O código dos CLPs é estruturado em um grande número de blocos de função programados em LD. Os requisitos de segurança da plataforma são organizados em 130 Matrizes Causa e Efeito com no máximo 50 linhas e 50 colunas.

O teste na aplicação real envolve a verificação das propriedades de *DFF* e *SFF* sobre a lógica de intertravamento dos subsistemas de *fire and gas* e *shutdown*. As propriedades de segurança analisadas estão contidas em 6 matrizes causa e efeito, sendo que as matrizes 1 e 2 se referem ao sistema de *shutdown* e as restantes se referem ao sistema de *fire and gas*. Os resultados para estas 6 matrizes podem ser observados na Tabela 3, que contém para cada MCE: o número de causas e efeitos, o maior número de estados encontrados entre os efeitos, o tempo de execução da cadeia de verificação formal e o número de *Safe Failure*, *Dangerous Failure* e propriedades aprovadas (SF/DF/OK).

São verificadas sempre 2 propriedades por saída

Tabela 3: Resultados para 6 MCEs da plataforma

MCE	C/E	Max. Estados	Tempo exec.	SF/DF/OK
1	2/4	16	22 s	0/0/8
2	8/4	640	44 s	0/0/8
3	11/9	3459	57 s	0/0/18
4	11/9	512	56 s	0/0/18
5	14/9	32	56 s	0/1/17
6	30/2	$> 2^{30}$	$>>$ 4 h	-/-/-

da matriz causa e efeito. Nas 4 primeiras matrizes, todas as propriedades foram aprovadas, porém na matriz de número 5 foram aprovadas 17 propriedades e foi detectado 1 *Dangerous Failure*. O contraexemplo foi gerado e confrontado com o LD. Este auxiliou na detecção e correção do erro na aplicação real.

Já na matriz 6 não foi viável aplicar a verificação formal mesmo com a a redução por COI, pois nesta matriz temos 2 efeitos e 30 causas, onde todas as 30 causas estão relacionadas a ambos os efeitos. Sendo assim, o número de estados no TTS é de no mínimo 2^{30} estados, o que torna computacionalmente inviável realizar a verificação formal com apenas uma etapa de redução. Os resultados desta matriz apontam para a necessidade de aprimorar os métodos de redução em trabalhos futuros, explorando por exemplo simetria ou métodos simbólicos, como BDD.

6 CONCLUSÕES

Este trabalho apresentou uma metodologia para realizar verificação formal em programas de CLPs que controlam Sistemas Instrumentados de Segurança das indústrias de petróleo e gás natural. Esta metodologia foi proposta com o objetivo de integrar-se a atual metodologia de desenvolvimento de programas de CLP neste tipo de indústria, podendo realizar a verificação a nível de usuário e na etapa de elaboração do LD, diferente da atual metodologia de teste que é realizada apenas na implementação do CLP. A verificação formal é um método exaustivo e realizado de maneira automática, por conseguinte mais seguro que testes realizados de forma empírica. A realização da verificação na etapa de programação do LD proporciona a detecção e correção dos erros de forma prematura, economizando tempo e dinheiro no processo de elaboração de um SIS.

A cadeia de verificação formal foi projetada com o propósito de ser toda realizada de forma automática, tendo como *front-end* o código LD e a MCE, documentos existentes no projeto de um SIS. Contudo, a transformação de Ladder para FIACRE e as etapas de redução foram realizadas de forma manual neste trabalho. No entanto, esses processos podem ser automatizados conforme as regras de tradução e de redução apresentadas neste trabalho.

A atual cadeia foi testada em programas reais que controlam os SIS de plataformas de petróleo do tipo "*offshore*". Porém, mesmo com a aplicação da etapa de redução por cone de influência, em alguns casos ainda ocorre o problema de explosão combinacional. Sendo assim, em trabalhos futuros novas etapas de redução devem ser elaboradas bem como a metodologia deverá ser aplicada em mais casos reais.

Referências

Adiego, B. F., Darvas, D., Viñuela, E. B., Tournier, J. C., Bliudze, S., Blech, J. O. and Suárez, V. M. G. (2015). Applying model checking to industrial-sized PLC programs, *IEEE Transactions on Industrial Informatics* **11**(6): 1400–1410.

Berthomieu, B. and Vernadat, F. (2006). Time petri nets analysis with tina, *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*, IEEE, pp. 123–124.

Biere, A., Cimatti, A., Clarke, E. and Zhu, Y. (1999). Symbolic model checking without bdds, *Tools and Algorithms for the Construction and Analysis of Systems* pp. 193–207.

Clarke, E. M., Grumberg, O. and Peled, D. (1999). *Model checking*, MIT press.

de Souza, M. M. F., Farines, J.-M. and de Queiroz, M. H. (2010). Modelagem e verificaçao de programas em Diagrama Ladder para Controladores Lógicos Programáveis, *XVIII Congresso Brasileiro de Automática-CBA*, Bonito.

ET-3000.00-1200-800-PGT-006 (2000). Project Guidelines for the Confection of Cause and Effect Matrixes and Logic Diagrams, *Standard*, Petrobras, Brazil.

Farail, P., Gaufillet, P., Peres, F., Bodeveix, J.-P., Filali, M., Berthomieu, B., Rodrigo, S., Vernadat, F., Garavel, H. and Lang, F. (2008). FIACRE: an intermediate language for model verification in the TOPCASED environment, *European Congress on Embedded Real-Time Software (ERTS)*.

Farines, J.-M., de Queiroz, M. H., da Rocha, V. G., Carpes, A. M. M., Vernadat, F. and Crégut, X. (2011). A model-driven engineering approach to formal verification of PLC programs, *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conf. on*, Toulouse.

Gergely, E. I., Coroiu, L. and Popentiu-Vladicescu, F. (2011). Methods for validation of PLC systems, *Journal of Computer Science and Control Systems* **4**(1): 47.

IEC 61131-3 (2003). Programmable controllers–part 3: Programming languages, *Standard*, International Electrotechnical Commission, Geneva.

Ovatman, T., Aral, A., Polat, D. and Ünver, A. O. (2016). An overview of model checking practices on verification of PLC software, *Software & Systems Modeling* **15**(4): 937–960.