

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE
AUTOMAÇÃO E SISTEMAS**

Felipe da Silva Lázaro

**METODOLOGIA PARA DESENVOLVIMENTO DE SISTEMAS DE
CONTROLE E MONITORAÇÃO DE NAVIOS ASSISTIDO POR
MODEL CHECKING**

Florianópolis
2018

Felipe da Silva Lázaro

**METODOLOGIA PARA DESENVOLVIMENTO DE SISTEMAS DE
CONTROLE E MONITORAÇÃO DE NAVIOS ASSISTIDO POR
MODEL CHECKING**

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.

Orientador: Prof. Dr. Max Hering de Queiroz

Coorientador: Prof. Dr. Jean-Marie Farines

Florianópolis
2018

Ficha de identificação da obra elaborada pelo autor através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lázaro, Felipe da Silva
Metodologia para Desenvolvimento de Sistemas de Controle e Monitoração de navios assistido por Model Checking / Felipe da Silva Lázaro ; orientador, Max Hering de Queiroz, coorientador, Jean-Marie Farines, 2018.
131 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2018.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. model checking. 3. verificação formal. 4. diagrama de lógica binária. 5. Navios. I. de Queiroz, Max Hering. II. Farines, Jean-Marie. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

**METODOLOGIA PARA DESENVOLVIMENTO DE SISTEMAS DE
CONTROLE E MONITORAÇÃO DE NAVIOS ASSISTIDO POR
MODEL CHECKING**

Felipe da Silva Lázaro

Esta Dissertação foi julgada adequada para obtenção do Título de “Mestre em Engenharia de Automação e Sistemas” e aprovada em sua forma final pelo Programa Pós-Graduação de Engenharia de Automação e Sistemas

Florianópolis, 08 de fevereiro de 2018.

Prof. Max Hering de Queiroz, Dr.
Orientador

Prof. Jean-Marie Farines, Dr.
Coorientador

Prof. Daniel Ferreira Coutinho, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia de
Automação e Sistemas

Banca Examinadora:

Prof. Max Hering de Queiroz, Dr.
Presidente

Prof. Fabio Luis Baldissera, Dr.
DAS/UFSC

Prof. Leandro Buss Becker, Dr.
DAS/UFSC

Capitão de Corveta (T) Rui Rodrigues de Mello Junior, Me.
Marinha do Brasil

Este trabalho é dedicado à minha esposa Claudia, à minha filha Lara e aos meus pais.

AGRADECIMENTOS

Agradeço primeiramente a Deus por tudo de bom que ele provê para minha vida e minha família.

À minha esposa Claudía, por estar sempre ao meu lado, pelo apoio dado em todos os momentos e por ter aceito nossa vinda a Florianópolis para a realização do curso.

À minha filha Lara, pelo momentos de alegria e pela paciência de aceitar os momentos em que eu não estava disponível para dá-la a atenção merecida.

Aos meus pais por tudo que sempre fizeram e fazem por mim.

À Marinha do Brasil, pelo apoio técnico, financeiro e por autorizar meu afastamento para realização do curso de forma integral.

Aos Professores Max e Jean-Marie pelas orientações pertinentes que muito contribuíram para este trabalho.

À UFSC, aos colegas de curso e aos professores do Programa de Pós-Graduação em Engenharia de Automação e Sistemas (PGEAS).

RESUMO

A Marinha do Brasil (MB) desenvolve o Sistema de Controle e Monitoração (SCM) para diversos navios de sua esquadra. Este sistema é responsável pelo controle das máquinas principais e auxiliares do navio, sendo essencial para sua operação e segurança. Neste trabalho é proposta uma metodologia de desenvolvimento de projetos para o SCM da MB, utilizando verificação formal por *model checking*. A metodologia estabelece documentos de especificação e métodos de tradução para a linguagem interpretada pela ferramenta de verificação formal. Assim, possibilitando certificar, de forma exaustiva e automática, que todas as propriedades de segurança essenciais ao projeto estão constando no documento de especificação do código para o Controlador Lógico Programável (CLP), gerando redução de custos na correção de erros, e aumentando a confiabilidade e disponibilidade do navio. Por ser automático, o método dispensa a necessidade dos projetistas terem conhecimentos em verificação formal. Um estudo de caso típico da MB foi utilizado e confirmou a viabilidade e eficácia da metodologia proposta.

Palavras-chave: Model Checking. FIACRE. Verificação Formal, Navios, Diagramas de Lógica Binária.

ABSTRACT

The Brazilian Navy develops the Control and Monitoring System for several ships in its fleet. This system is responsible for controlling the ship's main and auxiliary engines and is essential for its operation and safety. This work proposes a project development methodology for the Brazilian Navy Control and Monitoring System, using formal verification by model checking. The methodology establishes specification documents and translation methods for the language interpreted by the formal verification tool. Thus, making it possible to certify in an exhaustive and automatic way that all the essential safe properties to the project are included in the code specification document for the Programmable Logic Controller (PLC), generating cost reduction in error correction, and increasing reliability and availability of the ship. Because it is automatic, the method eliminates the need for designers to have formal verification skills. A typical Brazilian Navy's case study was used and confirmed the feasibility and efficacy of the proposed methodology.

Keywords: Model Checking. FIACRE. Formal Verification. Ships. Binary Logic Diagrams.

LISTA DE FIGURAS

Figura 1 - Estágios de Sistemas com CLPs e custos de correção e detecção de erros (GERGELY; COROIU; POPENTIU-VLADICESCU, 2011).	23
Figura 2 - Metodologia atual dos projetos de SCM.	32
Figura 3 - Proposta de metodologia para os projetos de SCM.	33
Figura 4 - Proposta de metodologia para os projetos de SCM, incluindo <i>model checking</i>	34
Figura 5 - Símbolos básicos da ISA 5.2 (1992) (a) porta <i>and</i> (b) porta <i>or</i> (c) porta <i>not</i> (d) bloco <i>set-reset</i> (e) bloco temporizador com atraso na inicialização da saída (<i>Delay Initiation of output</i>) (f) bloco Temporizador com atraso no desligamento da saída (<i>Delay Termination of output</i>) - DT (g) Temporizador por pulso (<i>Pulse Output</i>) - PO.	38
Figura 6 - Características comportamentais dos temporizadores (a) DI, (b) PO e (c) DT.	38
Figura 7 - BLD do procedimento de parada de emergência.	39
Figura 8 - Cadeia de verificação proposta no método.	52
Figura 9 - Arquitetura do programa em FIACRE.	54
Figura 10 - Arquitetura do programa em FIACRE do BLD da Figura 7.	55
Figura 11 - Ciclo de leitura de BLDs.	56
Figura 12 - Ciclo de leitura de BLDs em FIACRE.	57
Figura 13 - Contador temporal para entradas temporizadas.	66
Figura 14 - Cone de influência para a MCE da Tabela 11.	69
Figura 15 - Cone de influência da MCE da Tabela 9.	69
Figura 16 - Contraexemplo fornecido pelo SELT.	76
Figura 17 - Apresentação do contraexemplo como diagrama de sinais.	76
Figura 18 - BLD corrigido.	77
Figura 19 - BLD do Intertravamento de partida do MCP.	83
Figura 20 - BLD do procedimento para partida normal do MCP.	83
Figura 21 - BLD do Intertravamento para parada normal do MCP.	84
Figura 22 - BLD do procedimento para Parada em Emergência do MCP.	84
Figura 23 - BLD do Intertravamento para acoplar a ER.	85
Figura 24 - BLD do Intertravamento para desacoplar a ER.	86
Figura 25 - Cone de influência da aplicação da MB.	98
Figura 26 - Ilustração da redução por decomposição para o efeito “INTERLOCK_PARTIDA_MCP”.	100
Figura 27 - Contraexemplo do erro forçado, para o efeito “INTERLOCK_DESACOPLAR”.	103

LISTA DE TABELAS

Tabela 1 - Tabela resumo dos trabalhos relacionados.....	28
Tabela 2 - Representação da lógica $O1= I1 \text{ and not}(I2)$ com a simbologia da Petrobras.....	41
Tabela 3 - Representação da lógica $O1= (I1 \text{ and } I2)(T200)$ com a simbologia utilizada pela Petrobras.....	41
Tabela 4 - Representação da lógica $O1= (I1 \text{ and } I3) \text{ or } (I2 \text{ and not}(I1))$ com a simbologia utilizada pela Petrobras.....	41
Tabela 5 - Representação da lógica para a sentença “Ativar $O1$ se $I1$ and $I2$ e desativar se $\text{not}(I1)$ and $\text{not}(I2)$ ” com a simbologia utilizada pela Petrobras.....	42
Tabela 6 - Representação da lógica $O1= I1 \text{ and not}(I2)$ com o padrão proposto para a MB.....	43
Tabela 7 - Representação da lógica $O1= (I1 \text{ and } I2)(T200)$ com o padrão proposto para a MB.....	43
Tabela 8 - Representação da lógica $O1= (I1 \text{ and } I3) \text{ or } (I2 \text{ and not}(I1))$ com o padrão proposto para a MB.....	44
Tabela 9 - Representação da sentença lógica “Ativar $O1$ se $I1$ and $I2$ e desativar se $\text{not}(I1)$ and $\text{not}(I2)$ ” com o padrão proposto para a MB. ..	45
Tabela 10 - MCE das especificações de segurança para o procedimento de parada em emergência.....	46
Tabela 11 - MCE preenchida utilizando o padrão de escrita apresentado.....	48
Tabela 12 - Exemplo de MCE.....	68
Tabela 13 - Resultado do <i>model checking</i> realizado no BLD da Figura 7 e da MCE da Tabela 10.....	75
Tabela 14 - Resultados do estudo de caso.....	78
Tabela 15 - MCE da aplicação da MB.....	82
Tabela 16 - Resultado do <i>model checking</i> com a redução por cone de influência.....	99
Tabela 17 - Resultado do <i>model checking</i> com a redução por decomposição e por cone de influência.....	102

LISTA DE ABREVIATURAS E SIGLAS

BLD - *Binary Logic Diagram*
CLP - Controlador Lógico Programável
CTL - *Computation Tree Logics*
DEN - Diretoria de Engenharia Naval
DI - *Delay of Initiation*
DT - *Delay of Termination*
ER - Engrenagem Reversora
FBD - *Function Block Diagram*
FIACRE - *Format Intermédiaire pour les Architectures de Composants Répartis Embarqués*
FP - Falha Perigosa
FS - Falha Segura
IL - *Instruction List*
IPqM - Instituto de Pesquisas da Marinha
ISA - *International Society of Automation*
LFP - Livre de Falha Perigosa
LFS - Livre de Falha Segura
LTL - *Linear Temporal Logics*
MB - Marinha do Brasil
MCE - Matriz de Causa e Efeito
MCP - Motor a Combustão Principal
PO - *Pulse of Output*
SCAV - Sistema de Controle de Avarias
SCM - Sistema de Controle e Monitoração
SCMPA - Sistema de Controle e Monitoração de Propulsão e Auxiliares
SIS - Sistema Instrumentado de Segurança
SMR - Sistema Manual Remoto
SMV - *Symbolic Model Verifier*
TAF - Teste de Aceitação de Fábrica
TAM - Teste de Aceitação de Mar
TAR - Teste de Aceitação de Rio
TAP - Teste de Aceitação de Porto
TINA - *Time Petri Net Analyser*
TTS - *Timed Transition System*
XML - *eXtensible Markup Language*

LISTA DE SÍMBOLOS

- \neg Representa a lógica *not*
- \wedge Representa a lógica *and*
- \vee Representa a lógica *or*
- \square Representa o operador temporal “sempre”
- \diamond Representa o operador temporal “futuro”
- \circ Representa o operador temporal “próximo”

SUMÁRIO

1	INTRODUÇÃO	21
1.1	OBJETIVOS	24
1.1.1	Objetivo Geral	24
1.1.2	Objetivos Específicos	25
1.2	TRABALHOS RELACIONADOS	25
1.3	ESTRUTURA DA DISSERTAÇÃO	29
2	PROPOSTA DE METODOLOGIA PARA DESENVOLVIMENTO DE SCM DA MB ASSISTIDO POR MODEL CHECKING	31
2.1	<i>MODEL CHECKING</i> E LÓGICA TEMPORAL	35
2.2	ESPECIFICAÇÃO DO CÓDIGO PARA O CLP	37
2.3	REPRESENTAÇÃO DE PROPRIEDADES DE SEGURANÇA	40
2.3.1	Padrão de MCE da Petrobras	40
2.3.2	Proposta de padrão de MCE para a MB	42
2.3.3	Exemplos de MCE com o novo padrão	45
2.4	CONCLUSÃO DO CAPÍTULO	48
3	VERIFICAÇÃO FORMAL DE PROPRIEDADES DE SEGURANÇA EM BLDs	51
3.1	A LINGUAGEM INTERMEDIÁRIA FIACRE	53
3.2	TRADUÇÃO DE BLDs PARA FIACRE	54
3.2.1	Modelo do ciclo de leitura em FIACRE	56
3.2.2	Modelo de ciclo de leitura com blocos funcionais em FIACRE	58
3.2.3	Modelo do bloco temporizado DI	59
3.2.4	Modelo do bloco temporizado DT	60
3.2.5	Modelo do bloco temporizado PO	62
3.2.6	Modelagem de entradas digitais e analógicas em FIACRE	63
3.3	TRADUÇÃO DA MCE PARA LTL	64

3.4	ESTRATÉGIAS DE REDUÇÃO DO PROBLEMA DE EXPLOSÃO COMBINACIONAL.....	67
3.4.1	Redução por cone de influência	68
3.4.2	Redução por decomposição.....	69
3.5	APRESENTAÇÃO DE CONTRAEXEMPLO	75
3.5.1	Interpretação dos resultados e correção de erros	77
3.6	CONCLUSÃO DO CAPÍTULO.....	78
4	UTILIZAÇÃO DA NOVA METODOLOGIA EM UMA APLICAÇÃO REAL DA MB	79
4.1	A ESPECIFICAÇÃO DA MB.....	79
4.1.1	A MCE	81
4.2	DIAGRAMAS DE LÓGICA BINÁRIA DA MB	82
4.3	TRADUÇÃO DA MCE DA MB PARA LTL	86
4.4	TRADUÇÃO DOS DIAGRAMAS LÓGICOS DA MB PARA FIACRE.....	90
4.5	REALIZANDO <i>MODEL CHECKING</i>	97
4.6	CONCLUSÃO DO CAPÍTULO.....	104
5	CONCLUSÃO.....	105
5.1	TRABALHOS FUTUROS	106
	REFERÊNCIAS.....	109
	APÊNDICE A - Código FIACRE para <i>model checking</i> do BLD da Figura 7 para o primeiro efeito da MCE da Tabela 9.....	113
	APÊNDICE B - Código FIACRE completo para <i>model checking</i> da aplicação da MB.....	117

1 INTRODUÇÃO

A Marinha do Brasil (MB) para cumprir sua missão, que é preparar e empregar o Poder Naval, a fim de contribuir para a defesa da pátria, emprega seus diversos navios para patrulhamento do vasto território marítimo brasileiro e em missões no exterior quando solicitado. Para se ter um poder naval forte, a MB necessita de meios navais com tecnologias no estado da arte, que em grande parte, corresponde a navios com grandes níveis de automação, contribuindo com a eficiência do navio e disponibilizando maior quantidade de informações para tomada de decisões. Sendo assim, a MB tem adquirido e contruído navios cada vez mais automatizados, além de modernizar os que estão com tecnologia obsoletas. Navios mais automatizados exigem tripulação mais qualificada e requisitos de projeto mais rigorosos, visando reduzir a possibilidade de falhas humanas e dos equipamentos.

Inserido no contexto de uso de tecnologias no estado da arte, o Instituto de Pesquisas da Marinha (IPqM), organização militar da MB, desenvolve o Sistema de Controle e Monitoração (SCM). O projeto do SCM engloba o desenvolvimento de *software* e uso de *hardwares* comerciais. O SCM é o principal sistema responsável por comandar e monitorar as máquinas principais dos navios (Motores a combustão e/ou Turbinas a Gás) e seus sistemas auxiliares, além de adicionar estratégias de segurança de acordo com as peculiaridades de cada navio da MB. O SCM é composto de 03 subsistemas: o Subsistema de Controle e Monitoração de Propulsão e Auxiliares (SCMPA), o Subsistema de Controle de Avarias (SCAV) e o Subsistema Manual Remoto (SMR). O SCMPA tem como objetivo monitorar e controlar a propulsão do navio, fornecendo *setpoints* para os reguladores de motores e turbinas, além de implementar requisitos adicionais de segurança segundo os padrões da MB. Enquanto os reguladores cuidam do funcionamento desses equipamentos *standalone*, o SCM garante a correta integração entre as máquinas principais (turbina e motores), os acessórios (engrenagem redutora, acoplamento fluido e hélice de passo controlável) e o próprio navio (casco). Dessa forma, os *setpoints* fornecidos garantem o cumprimento dos requisitos especificados pela Diretoria de Engenharia Naval (DEN), além de otimizar o consumo de combustível, o nível de ruído e a utilização dos equipamentos. O SCMPA permite também, monitorar e atuar sobre equipamentos auxiliares do navio: bombas, válvulas, ventiladores, exaustores, “flaps”, ar-condicionado, estabilizador, etc. O SCAV tem o propósito de permitir a troca de informações entre uma Central de Controle de Avarias e seus reparos

(estações remotas de monitoração), monitorando e atuando em situações de avarias nos diversos compartimentos do navio, fornecendo apoio para uma rápida tomada de decisão do Comando. Já o SMR é um sistema de backup para operação em modo degradado, com as condições mínimas necessárias para operação do navio.

Para atender à criticidade e complexidade do projeto, além da necessidade de atender diversos requisitos de segurança, e por ser uma solução atrativa dada a sua robustez, confiabilidade e facilidade de utilização num ambiente similar ao industrial, o SCM utiliza Controladores Lógico Programáveis (CLP). CLPs são equipamentos eletrônicos digitais com memória programável para controlar através de implementações lógicas, entre entradas e saídas digitais ou analógicas, vários tipos de equipamentos ou processos (IEC 61131-1, 2003). A IEC 61131-3 (2013) estabelece as linguagens de programação de CLPs, que são: *Instruction List (IL)*, *Structured Text (ST)*, *Ladder Diagram (LD)*, *Function Block Diagram (FBD)* e *Sequential Function Chart (SFC)*. Erros na especificação e programação do CLP podem ocorrer e gerar danos catastróficos ao navio e sua tripulação, além de poder impedir que o navio efetivamente possa cumprir sua missão, que pode ser desde a salvaguarda da vida humana no mar, através de um resgate, até a atuação em conflitos.

Sistemas críticos como o SCM estão enquadrados na norma IEC 61508 (2010), que se aplica a qualquer software integrante de um sistema relacionado à segurança funcional ou relacionado à segurança de sistemas elétricos, eletrônicos ou com eletrônicos programáveis. A norma reconhece a importância e recomenda o uso de métodos formais em sistemas críticos, por impedir descrições incompletas, inconsistentes e ambíguas do sistema. Apesar disso, a atual metodologia utilizada pela MB no desenvolvimento de SCM não contempla o uso de métodos formais.

O trabalho sobre métodos de validação de sistemas com CLPs elaborado por Gergely, Coroiu, e Popentiu-Vladicescu (2011) apresenta a porcentagem de introdução e detecção de erros nos diferentes estágios do projeto, bem como o custo necessário para correção de erros (Figura 1). As etapas de projeto conceitual e programação são as que concentram o maior percentual de introdução de erros no sistema, e que em sua maior parte só são detectados na etapa de teste do sistema. Somente aproximadamente 15% de todos os erros são detectados antes das etapas de testes. Além disso, o custo de correção de erros é consideravelmente maior quanto mais as etapas se aproximam da etapa de operação do sistema. O custo de correção de erros antes das etapas de

testes é de aproximadamente 250 euros, durante as etapas de testes aproximadamente 1000 euros e durante a operação do sistema cerca de 12500 euros. Dada a relevância das fases de concepção e programação para o projeto como um todo, inclusive do ponto de vista econômico, há um interesse em métodos de verificação que possam reduzir ou minimizar o impacto desses erros em etapas futuras.

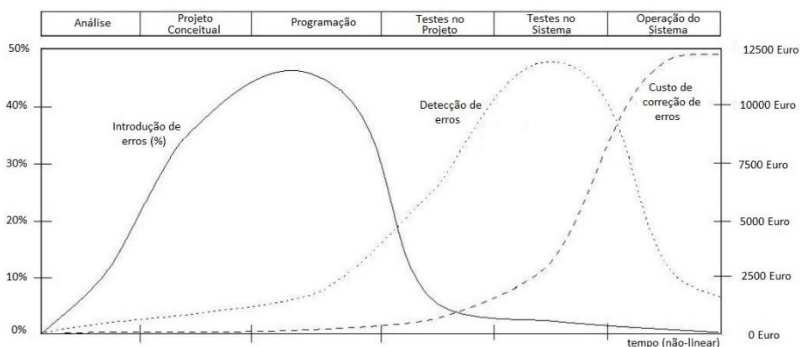


Figura 1 - Estágios de Sistemas com CLPs e custos de tempo de correção e detecção de erros (GERGELY; COROIU; POPENTIU-VLADICESCU, 2011).

Na prática, são utilizados vários métodos de verificação para assegurar que o resultado final corresponda aos requisitos inicialmente formulados, como (GERGELY; COROIU; POPENTIU-VLADICESCU, 2011)(FREY; LITZ, 2000):

- Revisão: é um processo manual realizado por equipe experiente e que normalmente não participa da elaboração do projeto. Gergely, Coroiu, e Popentiu-Vladicescu (2011) apresentam que 80% dos programas ainda são revisados manualmente;
- Modelagem e a simulação: baseiam-se em um modelo que descreve os comportamentos possíveis do sistema. O modelo geralmente é executável como um simulador de cenários, porém é limitado pela impossibilidade de simular todos os cenários possíveis;
- Teste: é uma técnica de verificação amplamente difundida que consiste na alimentação do sistema com valores de entrada e observação das saídas produzidas pelo sistema. O teste é similar à simulação, com a principal diferença de que o teste é feito em um sistema real. Geralmente é selecionado apenas um

subconjunto de domínio de entradas, não sendo assim, um método exaustivo; e

- Verificação formal: é uma técnica complementar à simulação e teste, consiste no uso de demonstrações matemáticas para provar que o sistema funciona corretamente. Constrói-se um modelo formal (matemático) do sistema e se expressa os requisitos por especificações formais. Para sistemas reais, essas demonstrações são muito complexas e requerem grande experiência. Sendo assim, é muito vulnerável a falhas.
 - *Model checking*: é uma técnica automática de verificação formal que faz uma exploração de todo espaço de estados do sistema, para provar matematicamente que uma determinada propriedade é atendida em todas as situações. Caso a propriedade não seja atendida um contraexemplo é apresentado. *Model checking* é a técnica utilizada neste trabalho e será abordada na seção 2.1.

Nesse contexto, vários trabalhos encontrados na literatura, e apresentados na seção 1.2, propõem o uso de verificação formal por *model checking* para realizar a detecção de erros em projetos que utilizam CLPs. Usualmente, realiza-se a verificação ou nas etapas de programação e de projeto conceitual ou somente na etapa de programação, porém, este estudo tem interesse em buscar uma solução que aborde esse problema numa fase inicial do projeto, no momento em que a lógica geral de funcionamento do CLP é especificada através de linguagens de alto nível, como o BLD (ISA 5.2, 1992).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O presente trabalho tem como objetivo geral propor uma metodologia de desenvolvimento de SCM para navios assistido por *model checking*. Diferentemente do encontrado na literatura, este trabalho realiza a verificação exclusivamente na etapa de projeto conceitual. A metodologia possibilita verificar se a especificação do código para o CLP contém todas as especificações de segurança desejadas pelo SCM, aumentando a confiabilidade do sistema, gerando

redução de custos em correção de erros e atendendo à IEC 61508 (2010).

1.1.2 Objetivos Específicos

Para atender ao objetivo geral deste trabalho, alguns objetivos específicos são necessários, como:

- Definir as formas de especificação das propriedades de segurança e do código para o CLP;
- Sistematizar as estratégias de tradução automática destas especificações para serem interpretadas por uma ferramenta de verificação formal;
- Utilizar estratégias de redução do problema de explosão combinacional de estados;
- Verificar formalmente por *model checking* (CLARKE; EMERSON; SISTLA, 1986), garantindo de forma exaustiva e automática que as propriedades de segurança estão contempladas na especificação do código para o CLP;
- Apresentar o resultado da ferramenta verificadora de forma acessível aos engenheiros envolvidos no projeto, facilitando e agilizando a correção de erros; e
- Utilizar a metodologia proposta em uma aplicação real da MB para verificar viabilidade e a eficácia da metodologia.

1.2 TRABALHOS RELACIONADOS

Na literatura são encontrados diversos trabalhos voltados para a modelagem e verificação formal de programas de CLP. O trabalho de Frey e Litz (2000) faz uma abordagem geral de métodos formais na programação de CLPs e foca nos métodos formais para verificação e validação. É feito um modelo genérico detalhado de um processo de projeto de controle. Os assuntos apresentados são categorizados usando três critérios: a abordagem geral para a tarefa (baseado em modelo, baseado em restrições ou sem um modelo), o formalismo (rede de Petri, autômato, sistema condição/evento) usado na descrição formal do estado, e o método (*model checking*, análise de alcançabilidade, prova por teorema) usado para analisar as propriedades.

Um dos primeiros e mais importantes trabalhos na área foi realizado por Moon (1994), o qual apresenta um método de verificação

para programas de CLP em LD. Regras de acionamentos típicos de um CLP são representadas em lógica temporal CTL (*Computation Tree Logic*) (HAFFER; THOMAS,1987) e modela um programa escrito em LD no formato de um código da ferramenta de verificação de modelos SMV (*Symbolic Model Verifier*) (MCMILLAN, 1993).

Outro trabalho importante foi feito por Canet et al. (2000), que utilizam a linguagem IL como entrada e a ferramenta verificadora SMV. Cada passo do modelo é um passo de execução do IL. Esta técnica leva a modelos que consomem mais espaço de estados na verificação do que a técnica de Moon (1994). Da mesma forma que Moon (1994), possui restrições de trabalhar apenas com variáveis booleanas, inteiros limitados e não modelar temporizadores.

Em Younis e Frey (2003) é proposta uma classificação para formalizar programas para CLPs em que utiliza critérios que vão desde a definição da linguagem na qual são escritos os programas para CLPs, até a utilização de um modelo formal que descreve tais programas. Estes critérios também levam em consideração os níveis de formalização que se baseiam na complexidade das estruturas que o processo de formalização pode tratar, bem como no objetivo da formalização que implica quais métodos podem ser aplicados para a geração de modelos formais.

Oliveira, C. (2006) descreveu uma metodologia sistematizada para a verificação de modelos em projetos de sistemas automatizados, criando um procedimento de tradução de programas escritos em FBD para linguagem reconhecida pela ferramenta de verificação SMV. Representou as especificações na forma de Matriz de Causa e Efeito (MCE) e realizou sua tradução para CTL. Dando continuação ao trabalho de Oliveira, C. (2006), Silva (2008) desenvolveu um projeto de melhoria do processo de engenharia de programas de CLPs de Sistemas Instrumentados de Segurança (SIS). Os programas considerados foram escritos em FBD e o documento básico para extração das especificações formais também foi a MCE. O trabalho descreve sua sequência principal, justificando a escolha do arcabouço teórico com: autômatos temporizados, matrizes de limites de diferenças e o verificador Uppaal (BEHRMANN; DAVID; LARSEN, 2004). Silva (2008) apresentou que a verificação de programas de CLP com mais de onze entradas, envolvendo um temporizador, é inviável. Isto considerando o uso do verificador Uppaal e sem se recorrer à abstração, a qual normalmente é feita por um especialista, dificultando o estabelecimento de um tratamento automático.

Oliveira, K. (2009) desenvolveu um método e uma ferramenta com a finalidade de gerar casos de teste para validar implementações a partir de especificações de SIS, mais especificamente nos programas de CLPs. O método consiste em transformar, de forma automática, *Binary Logic Diagrams* (BLD) (ISA 5.2, 1992), que representam a especificação, e programas em LD, que representam a implementação, em arquivos eXtensible Markup Language (XML). Estes arquivos descrevem autômatos temporizados no formato de entrada para a ferramenta Uppaal. Após a geração destes autômatos e de sua posterior validação na ferramenta Uppaal, testes de conformidade são aplicados, de forma automática. Para tal tarefa utiliza-se a ferramenta de teste Uppaal-TRON (LARSEN et al, 2005).

Em Da Silva et al. (2008), algoritmos para extração automática de autômatos temporizados a partir de FBDs são ilustrados. O modelo é composto por quatro tipos de autômatos. O primeiro é um modelo para flip-flops e elementos temporizados, o segundo é o modelo do ambiente para entradas físicas, o terceiro modela o ciclo de varredura, e o quarto modela o controle de execução e monitora a convergência de blocos de funções.

Em Barbosa et al. (2007), BLDs atendendo à ISA 5.2 (1992) são convertidos em autômatos temporizados utilizando uma modelagem similar à metodologia de Da Silva et al. (2008), a partir de BLDs para serem verificados pela ferramenta Uppaal.

Souza (2010) e Farines et al. (2011) apresentam uma abordagem de engenharia dirigida a modelos para modelar e verificar programas de CLP escritos em LD. CLP e planta são modelados na linguagem FIACRE (BERTHOMIEU et al., 2008)(BERTHOMIEU et al., 2007) de acordo com os modelos de transformação. A ferramenta verificadora SELT dentro do ambiente TINA (BERTHOMIEU; VERNADAT, 2006) foi utilizada a fim de garantir a satisfação de propriedades genéricas e orientadas à aplicações. O potencial desta abordagem e da cadeia de ferramentas associada é testado em um sistema pneumático controlado por CLP. A transformação de LD para os modelos FIACRE é descrita em detalhes e a verificação do CLP sozinho ou vinculado a uma planta é discutida no contexto da aplicação.

Adiego et al. (2015) elaboraram um metodologia baseada em um modelo intermediário para transformar programas de CLP escritos em várias linguagens: ST, SFC, e etc, para linguagens interpretadas por ferramentas de verificação. Apresentaram a sintaxe e a semântica do modelo intermediário, e as regras de transformação das linguagens ST e SFC para o verificador do modelo nuXmv (CAVADA et al., 2015).

A Tabela 1 apresenta, de forma resumida, os trabalhos relacionados a este e que realizam verificação formal por *model checking* em projetos envolvendo o uso de CLPs. Explicita as linguagens de programação utilizada nos trabalhos, como as propriedades a serem verificadas são formuladas e a ferramenta de verificação utilizada para realizar o *model checking*.

Tabela 1 - Tabela resumo dos trabalhos relacionados.

Autor(es)	Linguagem	Formalismo para as propriedades	Ferramenta de verificação
Moon (1994)	LD	CTL	SMV
Canet et al. (2000)	IL	LTL	SMV
Oliveira, C. (2006)	FBD	CTL	UPPAAL
Silva (2008)	FBD	MCE \rightarrow CTL	UPPAAL
Souza (2010)	LD	LTL	SELT
Farines et al. (2011)	LD	LTL	SELT
Adiego et al. (2015)	IL, ST e SFC	LTL	nuXmv

Esta dissertação apresenta consideráveis contribuições em relação aos trabalhos citados, como:

- Atuação em novo domínio de aplicação;
- Um método para desenvolvimento assistido por *model checking* aplicada às praticas da MB;
- Sistematização de especificações de segurança para geração automática de propriedades em lógica temporal,
- Sistematização de especificações do código para o CLP para geração automática de modelos formais;

- Aplicação da cadeia de verificação TINA-FIACRE e validação de BLDs; e
- Estratégia para abstração sistemática para lidar com a complexidade computacional do modelo formal.

1.3 ESTRUTURA DA DISSERTAÇÃO

Além do capítulo introdutório, esta dissertação está organizada da seguinte forma:

- Capítulo 2: Apresenta a atual metodologia de projetos de desenvolvimento de SCM da MB e suas limitações. Descreve a nova proposta de metodologia para o desenvolvimento de SCM da MB. Define e justifica a utilização dos documentos para especificação do código para o CLP e para as propriedades de segurança, bem como a simbologia utilizada;
- Capítulo 3: Aborda a linguagem intermediária de verificação FIACRE e as traduções necessárias nos BLDs e na MCE para serem interpretadas pela ferramenta verificadora. Apresenta também, a cadeia de verificação formal que viabiliza a realização do *model checking*, além de estratégias de redução do problema de explosão combinacional;
- Capítulo 4: Uma aplicação real e complexa da MB é utilizada para testar a nova metodologia proposta e apresenta-se os resultados encontrados; e
- Capítulo 5: Conclui o trabalho analisando os resultados obtidos com a metodologia proposta e apresenta as perspectivas de estudos futuros.

2 PROPOSTA DE METODOLOGIA PARA DESENVOLVIMENTO DE SCM DA MB ASSISTIDO POR MODEL CHECKING

Este capítulo apresenta a atual metodologia utilizada pela MB no desenvolvimento de SCM, e suas limitações. Também propõe uma metodologia para suprir as limitações da atual metodologia com documentos de especificação adequados às práticas da MB.

A metodologia atualmente utilizada pela MB para projeto do SCM (Figura 2), baseia-se na elaboração do código para o CLP a partir das informações contidas na especificação operativa do SCM e da experiência da equipe de projetistas. Na especificação operativa constam informações em linguagem natural e fluxogramas das funcionalidades desejadas, além dos requisitos operacionais e de segurança do sistema. A especificação do código para o CLP é apresentada como um pseudocódigo, que é entregue à equipe de programação para interpretá-lo e gerar o código para o CLP. Com o código elaborado, o seguinte plano de testes é realizado:

1. Teste de aceitação de fábrica (TAF): Ainda na fábrica, são realizados testes em busca de erros nas lógicas implementadas, de forma a se verificar o correto funcionamento das lógicas empregadas. Também são realizados testes com o hardware utilizado, simulando o sistema de forma quase completa;

2. Teste de Aceitação de Porto (TAP): Com o sistema instalado no navio, diversos testes são realizados, com cada equipamento individualmente e posteriormente com o conjunto de vários equipamentos até a sua totalidade; e

3. Teste de Aceitação de Mar ou Rio (TAM/TAR): É o teste em que o navio realmente navega sendo controlado pelo SCM, nesta etapa é feito um ajuste mais fino do sistema tendo como base todas as informações reais lidas pelo CLP.

Caso seja encontrado qualquer tipo de inconformidade em quaisquer testes realizados, são necessárias correções no código para o CLP. O navio estará apto para navegar com o SCM somente após receber aprovação em todos os testes.

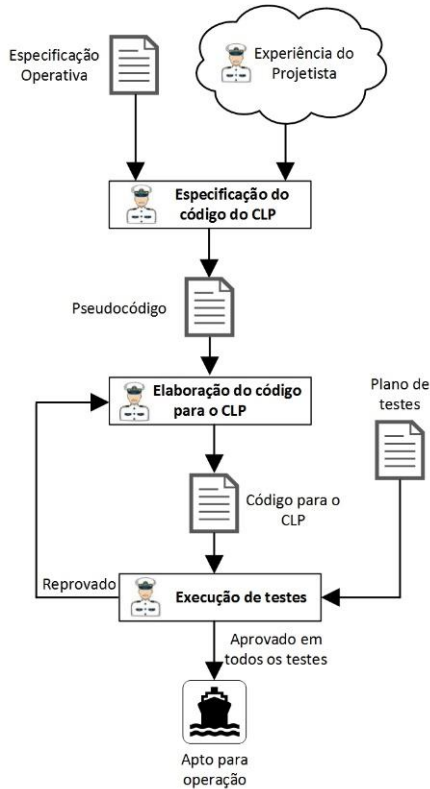


Figura 2 - Metodologia atual dos projetos de SCM.

As limitações do método atual estão na falta de um documento adequado para especificar o código para o CLP, a ausência de uma representação exclusiva para as especificações de segurança e a ausência de um método exaustivo, previsto pela IEC 61508 (2010), que garanta que o que foi especificado realmente contém todas os requisitos de segurança desejadas.

A proposta de metodologia para desenvolvimento de SCM da MB assistido por *model checking* visa suprir as limitações da metodologia atualmente utilizada. Consiste na criação de documentação padronizada para especificação do código para o CLP e para as propriedades de segurança do projeto. A Figura 3 apresenta a nova proposta de metodologia, onde o pseudocódigo atualmente utilizado para especificar o código para o CLP é substituído por um documento padronizado pela ISA 5.2 (1992), que são os BLDs. As características

dos BLDs são apresentadas na seção 2.2. Para especificar a propriedades de segurança do projeto é criado um documento específico para esse fim, ressaltando as propriedades de segurança desejadas, tal documento é a MCE, que será apresentada na seção 2.3. A nova metodologia permite determinar a fonte do erro em caso de reprovação do código para o CLP, que poderá ser proveniente de erro na elaboração do código ou um erro na especificação do código. Com a fonte do erro detectada, é realizada a correção e se mantém o documento de especificação do código e o código atualizados com todas as correções realizadas.

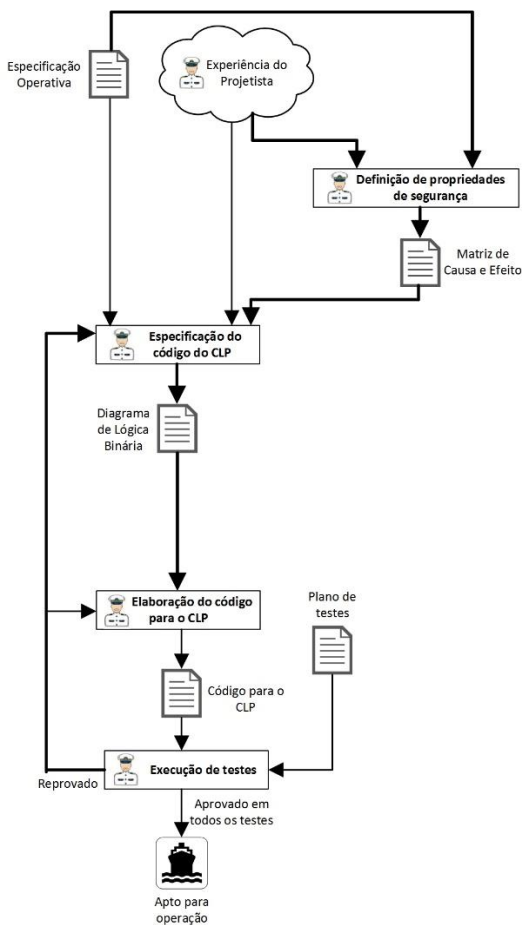


Figura 3 - Proposta de metodologia para os projetos de SCM.

Com as formas de especificação propostas por este trabalho, é possível realizar *model checking* nos BLDs para verificar, de forma exaustiva, que eles possuem todos os requisitos de segurança constantes na MCE, melhorando consideravelmente a metodologia atual. Pois, além de uma documentação formal e padronizada para as especificações, há a verificação exaustiva de propriedades desejadas sem que os técnicos e engenheiros envolvidos no projeto necessitem ter conhecimentos de *model checking* e modelagem formal (Figura 4). De acordo com Gergely, Coroiu, e Popentiu-Vladicescu (2011), a realização de *model checking* na etapa de projeto conceitual, que é uma das etapas em que mais são introduzidos erros, reduz consideravelmente os custos de correção de erros, pois são identificados em uma etapa que os custos de correção são baixos.

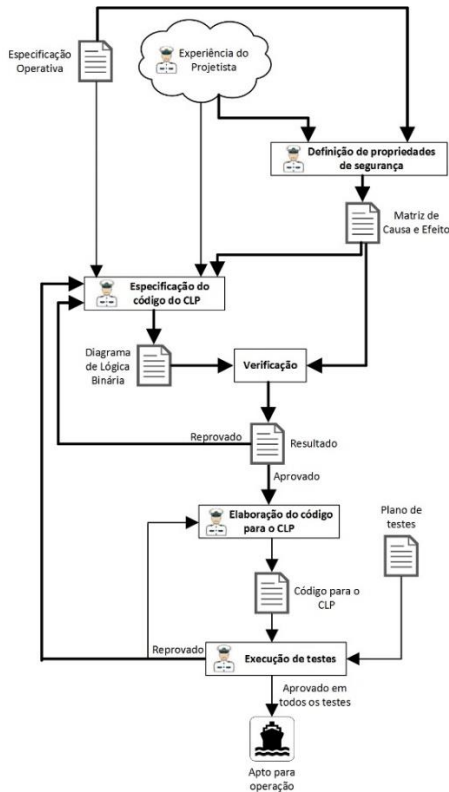


Figura 4 - Proposta de metodologia para os projetos de SCM, incluindo *model checking*.

2.1 MODEL CHECKING E LÓGICA TEMPORAL

Model checking é uma técnica automática para verificação e validação formal de sistemas concorrentes de estados finitos (CLARKE; EMERSON; SISTLA, 1986). Possibilita explorar todos os cenários de execução de um sistema, de forma a se certificar exaustivamente que o sistema verdadeiramente satisfaz certas propriedades que dificilmente seriam identificadas em emulações, testes e simulações, pois não são técnicas exaustivas. Apesar do *model checking* ser uma técnica de verificação automática, a ferramenta verificadora precisa interpretar o modelo formal do sistema que se deseja verificar, bem como as propriedades desejadas, em um modelo preciso e sem ambiguidades. Com base nessas premissas, o *model checker* (ferramenta verificadora) examina todos os estados relevantes do sistema para checar se uma propriedade desejada é satisfeita. Se é encontrado um estado que viola a propriedade em consideração, a ferramenta fornece um contraexemplo no qual descreve um caminho de execução que leva do estado inicial ao estado indesejado, desse modo fornecendo uma informação importante para encontrar a origem do erro e consertá-lo (MOON, 1994).

Dois importantes vantagens do *model checking* em relação a métodos tradicionais são que o processo pode ser totalmente automatizado, não exigindo conhecimentos especiais em lógicas ou provas por teorema, e o fornecimento de um contraexemplo que é gerado quando uma determinada propriedade é falsa.

Ao aplicar o *model checking* em um projeto, as seguintes fases podem ser identificadas:

- Modelagem: modela-se o sistema em consideração e as propriedades a serem verificadas em linguagem apropriada para interpretação da ferramenta verificadora;
- Verificação: executa-se a verificação para checar a validade das propriedades sobre o modelo; e
- Análise: se a propriedade falhar, analisa-se o contraexemplo para encontrar a origem do erro para, em seguida, realizar as correções necessárias no projeto e executar novamente a verificação para confirmar a correção do erro. Caso a propriedade seja satisfeita, o sistema estará exaustivamente verificado.

O modelo formal mais utilizado em *model checking* para descrição do sistema a ser verificado é o sistema de transição. Um sistema de transição é basicamente, um grafo direcionado, onde nós representam estados e arcos representam transições (mudança de

estados). Um estado descreve informações sobre o sistema em algum momento de seu comportamento e as transições especificam como o sistema pode evoluir de um estado para outro.

Model checking é bastante aplicado em verificação de sistemas reativos, os quais se caracterizam por uma interação contínua com o ambiente no qual estão inseridos. Os sistemas dessa natureza tipicamente recebem estímulos do ambiente e quase que instantaneamente reagem às entradas recebidas. Tradicionalmente são distribuídos, concorrentes e não possuem um término de execução, ou seja, estão constantemente prontos para interagir com o usuário ou outros sistemas, como é o caso do SCM da MB. Para sistemas reativos, a validação de um correto funcionamento depende também da análise das execuções, que é a ordem em que os eventos ocorrem, e não somente da avaliação dos valores de saída resultantes para um determinado conjunto de valores de entrada.

Lógica temporal é um formalismo adequado para representar propriedades de sistemas reativos e, portanto, especificar seu correto funcionamento para fins de verificação. Ela estende a lógica proposicional ou de predicado, por modalidades que permitem se referir ao comportamento infinito desses sistemas. As lógicas temporais fornecem uma notação bem intuitiva, porém matematicamente precisa para expressar propriedades sobre relações entre estados em uma execução. A natureza subjacente de tempo em lógica temporal pode ser tanto linear quanto ramificada. Na perspectiva linear, a cada instante existe somente um instante sucessor, enquanto que na visão ramificada cada instante possui uma ramificação que pode se dividir em percursos alternativos. A lógica temporal CTL (CLARKE; EMERSON, 1981) adota a abordagem ramificada, enquanto que a lógica LTL (*Linear Temporal Logics*) (PNUELI, 1977) adota a perspectiva linear. Neste trabalho, a lógica utilizada é a LTL, pois a versão da ferramenta verificadora utilizada possui como entrada um subconjunto da lógica LTL para realização de verificação formal.

Apesar do termo "temporal" em LTL levar ao entendimento de ser uma relação com o comportamento em tempo real de sistemas reativos, isto é verdade apenas em sentido abstrato, pois permite a especificação da ordem relativa entre eventos, não suportando nenhum modo de se referir ao tempo preciso de acontecimento dos eventos. Em termos de sistemas de transição, nem o tempo para a realização de uma transição ou o tempo de permanência em um estado podem ser especificados usando as modalidades elementares da lógica temporal LTL. Ao invés disso, essas modalidades permitem especificar a ordem

na qual determinadas proposições atômicas são verdadeiras durante uma execução ou mais.

Uma propriedade descrita com fórmula LTL é formada pela combinação de proposições atômicas, conectores booleanos e operadores temporais. As proposições atômicas fazem afirmações sobre os estados, em que estas proposições são relações elementares, as quais, em um dado estado, possuem um valor verdadeiro bem definido. Os conectores booleanos são: conjunção, disjunção, negação, implicação e dupla implicação. Os operadores temporais permitem construir expressões relacionadas ao sequenciamento dos estados ao longo de uma execução e não apenas aos estados individualmente. Os operadores temporais presentes em LTL (considerando p e q como proposições atômicas) são:

- *Next* : $(\) p$ (define que p é válido no próximo estado)
- *Until*: $p U q$ (define que p pode ser válido até q ser)
- *Globally*: $[] p$ (define que p é válido em todos os estados)
- *Future*: $\langle \rangle p$ (define que q é eventualmente válido no futuro ou no estado atual)

2.2 ESPECIFICAÇÃO DO CÓDIGO PARA O CLP

Na metodologia proposta, a especificação do código para o CLP é representada na forma de BLDs. Os BLDs têm como função fornecer informações para controle, intertravamento para partida, operação, alarme e bloqueio de equipamentos e processos nos vários segmentos industriais. Uniformizam a simbologia e regras para a elaboração de documentos, facilitando o entendimento e a comunicação entre os técnicos e engenheiros envolvidos nos projetos.

A Figura 5 apresenta os elementos básicos estabelecidos pela ISA 5.2 (1992) que são: lógica E, OU, inversor, memória, atraso de inicialização, atraso de desligamento e pulso de saída. Com esses elementos é possível representar diversas relações lógicas.

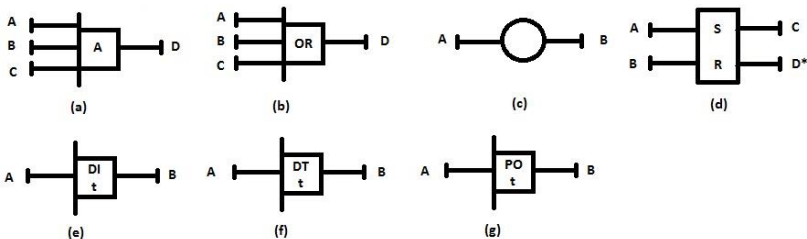


Figura 5 - Símbolos básicos da ISA 5.2 (1992) (a) porta *and* (b) porta *or* (c) porta *not* (d) bloco *set-reset* (e) bloco temporizador com atraso na inicialização da saída (*Delay Initiation of output*) (f) bloco Temporizador com atraso no desligamento da saída (*Delay Termination of output*) - DT (g) Temporizador por pulso (*Pulse Output*) - PO.

As características comportamentais dos temporizadores DI, PO e DT da ISA 5.2 (1992) estão apresentadas na Figura 6.

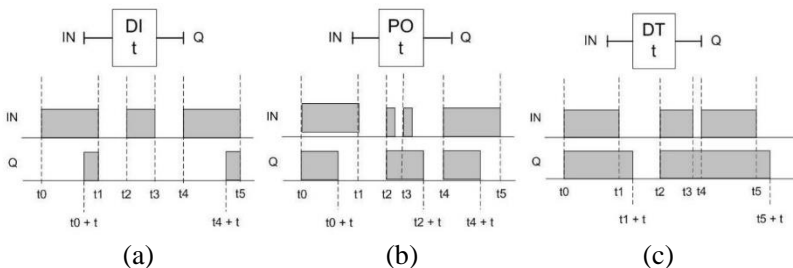


Figura 6 - Características comportamentais dos temporizadores (a) DI, (b) PO e (c) DT.

O temporizador DI, Figura 6a, tem como característica ativar sua saída sempre que sua entrada estiver ativa por um intervalo de tempo determinado. Caso a qualquer momento a entrada deixe de estar ativa, a saída será desativada.

No temporizador PO (Figura 6b), sempre que ocorrer no mínimo um pulso de ativação na entrada, a saída permanecerá ativa por um intervalo de tempo determinado, independente do estado da entrada. A saída somente poderá ser ativada novamente após nova ocorrência de no mínimo um pulso de ativação na entrada.

Para o temporizador DT (Figura 6c), sua saída permanecerá ativa sempre que a entrada estiver ativa, porém quando a entrada for desativada, a saída ainda permanecerá ativa por um intervalo de tempo determinado. Caso a entrada volte a ser ativada antes da desativação da

saída, o contador de tempo é zerado e só volta a contar quando a entrada for desativada novamente.

A Figura 7 apresenta um exemplo de BLD atendendo à norma ISA 5.2 (1992), que consiste de um exemplo típico encontrado nos projetos de SCM da MB, que é o procedimento para parada em emergência do motor principal de um navio.

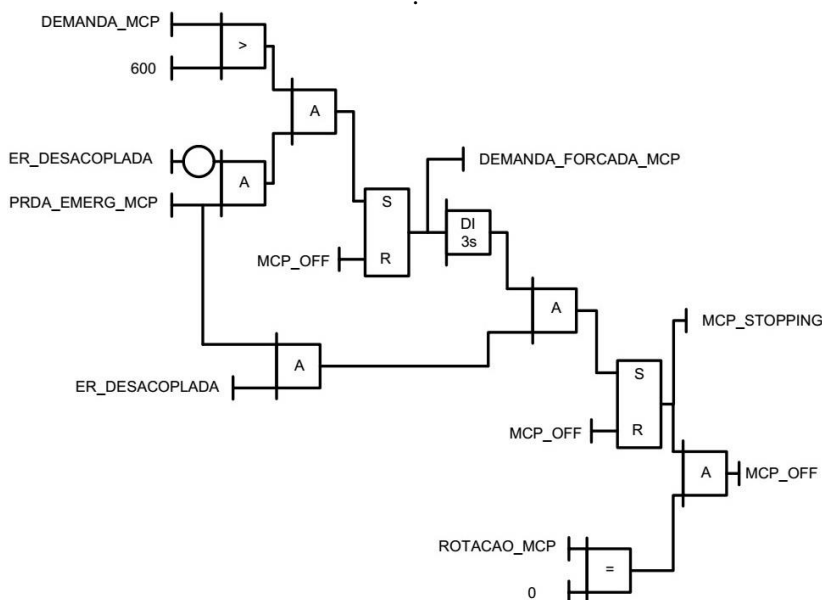


Figura 7 - BLD do procedimento de parada de emergência.

O BLD (Figura 7) especifica que a demanda forçada (**DEMANDA_FORCADA_MCP**) no Motor de Combustão Principal (MCP) permanecerá ativa se ao menos uma vez, simultaneamente, o botão de parada de emergência for pressionado, a Engrenagem Reversora (ER) não estiver desacoplada (**ER_DESACOPLADA**) e se a demanda estiver acima de 600rpm (**DEMANDA > 600**) e será desativada se o MCP estiver na condição parado (**MCP_OFF**). Além disso, permanecerá em modo parando (**MCP_STOPPING**) se ao menos uma vez, simultaneamente, o botão de parada de emergência (**PRDA_EMERG**) for pressionado, a ER estiver desacoplada e se for forçada uma demanda acima de 600rpm por no mínimo 3 segundos, e será desativado se o MCP encontrar-se na condição parado. Apresenta

também, que o MCP estará parado se a condição parando estiver ativa e a rotação chegar a zero (ROTACAO_MCP_=0).

2.3 REPRESENTAÇÃO DE PROPRIEDADES DE SEGURANÇA

Propriedades de segurança representadas na forma de MCE são uma forma de apresentar os requisitos de segurança dos sistemas com detalhes e de forma concisa, através de lógicas simbolizadas em suas linhas e colunas. A MCE é uma forma de representação matricial com as causas nas linhas e os efeitos nas colunas. Apresenta o inter-relacionamento entre eventos (causa) e as ações (efeito), que devem ocorrer de forma automática e controlada pelo sistema. Como a MCE é construída de forma segmentada (sistemas e subsistemas), é possível visualizar as entradas de cada segmento, identificando as relações entre eles. Para confecção da MCE, são necessários apenas os conhecimentos já adquiridos pelos técnicos e engenheiros envolvidos nos projetos, necessitando apenas de adaptações no sentido de representar as lógicas desejadas com simbologia padronizada na MCE.

Uma das propostas desta dissertação consiste em propor uma simbologia para ser utilizada na MCE da MB. Tal simbologia se baseou na utilizada pela Petrobras (PETROBRAS, 2005), que também representa suas especificações de segurança na forma de MCE.

2.3.1 Padrão de MCE da Petrobras

A Petrobras estabelece que as simbologias utilizadas na MCE são apenas as lógicas “e”, “ou”, “não ou”, e “ou temporizado” (Petrobras, 2005). Desta forma, a seguinte simbologia pode ser utilizada:

X : Representação para lógica *or*;

N : Representação para lógica *not or*;

A_i : Representação para lógica *and*;

T_n : Representação para lógica temporizada;

Onde *i* é um número natural usado para distinguir conjuntos de lógicas entre si, e *n* é o tempo especificado para o temporizador.

As Tabelas 2, 3, 4 e 5 apresentam exemplos de como algumas lógicas podem ser representadas com a simbologia da Petrobras.

Tabela 2 - Representação da lógica $O1 = I1 \text{ and not}(I2)$ com a simbologia da Petrobras.

	aux	O1
I1		A
I2	N	
aux		A

Tabela 3 - Representação da lógica $O1 = (I1 \text{ and } I2)(T200)$ com a simbologia utilizada pela Petrobras.

	aux	O1
I1	A	
I2	A	
aux		T200

Tabela 4 - Representação da lógica $O1 = (I1 \text{ and } I3) \text{ or } (I2 \text{ and not}(I1))$ com a simbologia utilizada pela Petrobras.

	aux1	aux2	aux3	O1
I1	A	N		
I2			A	
I3	A			
aux1				X
aux2			A	
aux3				X

Tabela 5 - Representação da lógica para a sentença “Ativar O1 se I1 and I2 e desativar se not(I1) and not(I2)” com a simbologia utilizada pela Petrobras.

	aux1	aux2	aux3	aux4	O1
I1	A	N			
I2	A		N		
aux1				X	
aux2					A
aux3					A
aux4					A
O1				X	

2.3.2 Proposta de padrão de MCE para a MB

Levando-se em conta a simbologia utilizada pela Petrobras e às especificidades da MB, algumas alterações na simbologia da Petrobras são necessárias para atender à MB de forma mais adequada. Sendo assim, as características desejadas para a MCE proposta para a MB são:

- Todas as simbologias do padrão da Petrobras;
- Utilização das simbologias “e negado” e “e temporizado”;
- Previsão de possibilidade de representar mais de uma relação para mesma entrada;
- Previsão de simbologia para explicitar variáveis discretas provenientes de entradas analógicas; e
- Estrutura para contemplar saídas retentivas ou memórias.

De acordo com as características desejadas para a MCE da MB, se propôs a seguinte simbologia para expressar lógicas na MCE da MB:

- X : Representação para lógica *or*;
- N : Representação para lógica *not or*;
- Ai : Representação para lógica *and*;
- NAi : Representação para lógica *not and*;
- Tn : Representação de atraso temporal;

+ : Representação para lógicas que levam a ativação do efeito;
 - : Representação para lógicas que desativam o efeito; e
 Múltiplas relações (,) : Representação para múltiplas relações entre entradas da mesma causa para um mesmo efeito.

Onde i é um número natural usado para distinguir conjuntos de lógicas entre si, e n é o tempo especificado para o temporizador.

As Tabelas 6, 7, 8 e 9 apresentam exemplos de lógicas expressadas com a simbologia proposta para a MB e que atendem às características desejadas.

A simbologia para a lógica “e negado” está apresentada na Tabela 6, que expressa a lógica $O1 = I1 \text{ and not}(I2)$ com a nova simbologia proposta. Comparando as MCEs (Tabela 2 e Tabela 6) com a simbologia da Petrobras e a proposta para a MB, pode-se verificar que o novo padrão proposto para a MB viabilizou redução no tamanho da MCE e facilitou a interpretação da lógica expressada, pois a utilização de uma variável auxiliar deixou de ser necessária para expressar a lógica desejada.

Tabela 6 - Representação da lógica $O1 = I1 \text{ and not}(I2)$ com o padrão proposto para a MB.

	O1
I1	A
I2	NA

Para a simbologia “e temporizado”, ao se expressar a lógica $O1 = (I1 \text{ and } I2)(T200)$ com o padrão proposto de MCE para a MB (Tabela 7), também verifica-se que o novo padrão proposto suprimiu uma coluna da MCE, comparando as MCEs (Tabela 3 e Tabela 7). A nova simbologia também permitiu redução na dimensão da MCE sem dificultar a interpretação da lógica expressada na MCE.

Tabela 7 - Representação da lógica $O1 = (I1 \text{ and } I2)(T200)$ com o padrão proposto para a MB.

	O1
I1	AT200
I2	AT200

Na característica desejada para “Previsão de possibilidade de representar mais de uma relação para mesma entrada” observa-se que, por exemplo, a representação da lógica $O1 = (I1 \text{ and } I3) \text{ or } (I2 \text{ and not}(I1))$ com o padrão da Petrobras gera uma MCE (Tabela 4) de grandes dimensões. A mesma lógica representada com o padrão proposto para a MB, e apresentada na Tabela 8, gerou grande redução nas dimensões da MCE sem prejudicar a interpretação da lógica expressada na MCE.

Tabela 8 - Representação da lógica $O1 = (I1 \text{ and } I3) \text{ or } (I2 \text{ and not}(I1))$ com o padrão proposto para a MB.

	O1
I1	A1, NA2
I2	A2
I3	A1

A característica desejada para “Previsão de simbologia para explicitar variáveis discretas provenientes de entradas analógicas” é satisfeita estabelecendo um padrão construtivo para a MCE, pela distinção em colunas para entradas analógicas e digitais, favorecendo a interpretação da MCE e a identificação de relações entre sistemas e subsistemas que compartilham mesma entrada analógica. Uma variável analógica sendo utilizada em diferentes limites não deixa claro que estão relacionadas entre si, caso fossem representadas apenas em suas formas discretas. Também, para facilitar a compreensão da MCE e das peculiaridades entre as causas presentes, causas com relações de dependência são agrupadas na MCE.

Na simbologia proposta para representar “Estrutura para contemplar saídas retentivas ou memórias”, ao se expressar na MCE (Tabela 9) a lógica para a sentença “Ativar O1 se I1 and I2 e desativar se not(I1) and not(I2)”, pode-se verificar que o novo padrão proposto para a MB também viabiliza grande redução no tamanho da MCE em relação à MCE da Petrobras (Tabela 5), com a supressão de três linhas e cinco colunas da MCE, também sem dificultar a interpretação da lógica expressada.

Tabela 9 - Representação da sentença lógica “Ativar O1 se I1 and I2 e desativar se not(I1) and not(I2)” com o padrão proposto para a MB.

	O1
I1	A1+, NA2-
I2	A1+, NA2-

Para facilitar a interpretação da MCE, foram adicionadas algumas informações importantes não existentes no padrão utilizado pela Petrobras, como a coluna AI, que apresenta os sinais analógicos existentes, a coluna DI para segmentar os sinais discretos. A coluna *voting* que é o campo para explicitar caso de votações, a coluna *TAG* para identificar a *tag* de cada sinal, e o campo *NOTES* que apresenta eventuais relações não representáveis na MCE ou de complexidade alta, presentes no padrão da Petrobras foram mantidas.

2.3.3 Exemplos de MCE com o novo padrão

Um exemplo de MCE está apresentado na Tabela 10, onde constam as propriedades de segurança para o procedimento de parada de emergência do motor principal de um navio, e que será utilizado neste trabalho para apresentar e exemplificar cada etapa da nova metodologia proposta.

Tabela 10 - MCE das especificações de segurança para o procedimento de parada em emergência.

AI	DI	VOTING	TAG		DEMANDA_FORCADA_MCP	MCP_STOPPING	MCP_OFF	NOTES
			T	A				
ER	DESACOPLADA		1	NA1+	A1+			
	PRDA_EMERG_MCP		2	A1+	A1+			
DEMANDA_MCP	>600		3	A1+				
ROTACAO_MCP	=0		4			A1		
	MCP_OFF		5	X-	X-			
	DEMANDA_FORCADA_MCP		6		T3+			
	MCP_STOPPING		7			A1		

Interpretando-se a lógica representada na MCE da Tabela 10, para o primeiro efeito (DEMANDA_FORCADA_MCP), o efeito será ativado sempre que, simultaneamente, a ER estiver acoplada, o botão de parada estiver pressionado e a demanda do MCP estiver acima de 600rpm, e será desativado quando o MCP estiver parado. Por definição, sempre a desativação possuirá prioridade sobre a ativação.

Como a proposta deste trabalho inclui nova simbologia para preenchimento da MCE, esta subseção apresenta alguns padrões de escrita com intuito de facilitar o preenchimento da MCE a partir da linguagem natural que normalmente está presente nos documentos de especificação da MB. A seguir são utilizados alguns exemplos em linguagem natural retirados de documentos da MB, com a apresentação de como eles podem ser reescritos de forma a facilitar o preenchimento da MCE, e posteriormente, como a MCE é preenchida com essas informações. Exemplificando:

- Linguagem natural: “Em caso de partida normal, a chave de bateria deve estar ligada e o “starter” do motor deve encontrar-se na posição “off””.
Reescrevendo em linguagem padronizada: O comando de partida normal será habilitado se a chave de bateria estiver ligada e o “starter” do motor encontrar-se na posição off.
- Linguagem natural: “É necessário aguardar um intervalo de 2 segundos entre mudança de estados na Engrenagem Reversora”
Reescrevendo em linguagem padronizada: O comando acoplar da Engrenagem Reversora será habilitado se uma mudança de estados tiver ocorrido a mais de 2 segundos.
- Linguagem natural: “Para que seja habilitado o comando de parar o motor normalmente, é estipulado pelo fabricante que o mesmo deve permanecer em ralenti por 5 minutos.”
Reescrevendo em linguagem padronizada: O comando parar normal será habilitado se o motor permanecer em ralenti por mais de 5 minutos.
- Linguagem natural: “Caso o motor se encontre virando em baixa rotação (abaixo de 1000 rpm) por mais de 4 horas é recomendado o procedimento de descarga de resíduos dos cilindros”
Reescrevendo em linguagem padronizada: O comando descarga recomendada será habilitado quando o MCP estiver em baixa rotação por mais de 4 horas.
- Linguagem natural: “Caso o sinal REC esteja ativo simultaneamente com o sinal EPB, o sinal EBS deverá ser ativado. Caso o sinal REC esteja ativo simultaneamente com o sinal EPB desativado, o sinal de EBS deverá ser desativado”.
Reescrevendo em linguagem padronizada: O sinal EBS deverá ser habilitado se não existir o sinal REC e existir o sinal EPB. Além disso, deverá ser desabilitado caso não exista o sinal EPB e exista o sinal REC.

Com a linguagem padronizada interpretada das especificações dos exemplos, o preenchimento da MCE se torna mais simples e direta. A Tabela 11 apresenta a MCE preenchida com as propriedades definidas nos exemplos.

Tabela 11 - MCE preenchida utilizando o padrão de escrita apresentado.

AI	DI	VOTING	TAG	TAG							
					COMANDO_PARTIDA_NORMAL	COMANDO_ACOPLAR	COMANDO_PARADA_NORMAL	DESCARGA_RECOMENDADA	EBS	NOTES	
	CHAVE_LIGADA			1	A						
	STARTER_OFF			2	A						
ER	DESACOPLADA			3		T2					
ER	ACOPLADA_AV			4		T2					
ER	ACOPLADA_AR			5		T2					
ROTACAO_MCP	>=600			6		A1T300					
ROTACAO_MCP	<700			7		A1T300					
ROTACAO_MCP	<1000C			8			T14400				
	EPB			9					A1+,NA2-		
	REC			10					NA1+,A2-		

2.4 CONCLUSÃO DO CAPÍTULO

O presente capítulo apresentou a atual metodologia de desenvolvimento de SCM da MB e suas limitações. Buscando supri-las, uma nova metodologia foi proposta, apresentando como devem ser especificados o código para o CLP e as propriedades de segurança. Uma simbologia para MCE, com alterações à utilizada pela Petrobras, foi proposta e as justificativas para tais alterações realizadas foram apresentadas. Por se tratar de uma nova forma de especificação na MB e para facilitar o uso por parte dos engenheiros e técnicos envolvidos no projeto, foram apresentados exemplos de como a MCE para a MB deve ser preenchida.

Especificar o código para o CLP na forma de BLDs facilita a interpretação da equipe programadora, agilizando o processo de elaboração do código para o CLP. Permite também, documentar adequadamente as alterações provenientes de erros encontrados nas lógicas implementadas, o que antes não era possível devido ao uso de pseudocódigo para especificar o código para o CLP.

O uso de MCE para representar propriedades de segurança ressalta a importância de tais propriedades no projeto, tanto para os projetistas como para os operadores do sistema.

A proposta de simbologia para ser utilizada na MCE da MB baseada na utilizada pela Petrobras possibilita representar as propriedades de segurança de forma ainda mais concisa e de mais fácil compreensão do que com a simbologia da Petrobras.

O capítulo seguinte dá sequência a este abordando as estratégias para viabilizar o uso de técnica de verificação formal para se certificar que os BLDs possuem os requisitos apresentados na MCE.

3 VERIFICAÇÃO FORMAL DE PROPRIEDADES DE SEGURANÇA EM BLDs

A partir da nova metodologia proposta, em que os BLDs especificam o código para o CLP e a MCE as propriedades de segurança, este capítulo apresenta os modelos de tradução do BLD e da MCE para FIACRE e LTL, respectivamente. São apresentadas também técnicas de redução do problema de explosão combinacional de estados para realização da verificação formal por *model checking*.

Na metodologia de desenvolvimento de SCM assistido por *model checking* proposta nesta dissertação (Figura 4), a verificação consiste na comparação da especificação do código para o CLP escrito em BLD com as propriedades de segurança apresentadas em MCE. Porém, as ferramentas verificadoras existentes não são capazes de interpretar diretamente BLDs e MCE, pois trabalham com linguagens baseadas em formalismos matemáticos (autômatos, redes de Petri, sistemas de transição temporizados) (BERTHOMIEU et al., 2008). Então, alguns procedimentos são necessários para viabilizar a automação do *model checking*, como apresentar as especificações de segurança e do código para o CLP em uma linguagem interpretável por uma ferramenta verificadora.

A linguagem FIACRE (BERTHOMIEU et al., 2008)(BERTHOMIEU et al., 2007) foi projetada para ser uma linguagem intermediária formal de alto nível entre as linguagens de modelagem e as ferramentas de verificação. Linguagens de alto nível são aquelas cuja sintaxe se aproxima mais da linguagem humana e se distanciam mais da linguagem de máquina. Nesta dissertação, a elaboração de modelos de tradução do BLD e da MCE para a linguagem FIACRE é facilitado por ser uma linguagem de alto nível e por possuir ferramentas automáticas de tradução para *Timed Transition System* (TTS), que é uma representação mais complexa que FIACRE e é diretamente interpretada por ferramentas de verificação formal. A ferramenta verificadora SELT dentro do ambiente TINA (BERTHOMIEU; VERNADAT, 2006) foi utilizada na metodologia por ser capaz de interpretar a linguagem formal gerada pelas ferramentas de tradução automática de FIACRE.

A Figura 8 apresenta as etapas de tradução tanto dos BLDs quanto da MCE para FIACRE. As propriedades extraídas da MCE são representadas em como fórmulas em LTL (PNUELI, 1977). Após a tradução dos BLDs para FIACRE, o código é compilado pela ferramenta automática FRAC (BERTHOMIEU; VERNADAT, 2006), onde o

modelo em FIACRE é transformado em TTS e LTL. Posteriormente, a ferramenta verificadora TINA/SELT confronta as propriedades de segurança formuladas em LTL com a especificação do código para o CLP em TTS. Em caso de resultado falso, o SELT fornece um contraexemplo, que na metodologia proposta é apresentado na forma de diagrama de sinais, facilitando a identificação e posterior correção do erro. Em caso de resultado verdadeiro, os BLDs são considerados como aprovados e podem seguir para a etapa seguinte de elaboração do programa do CLP.

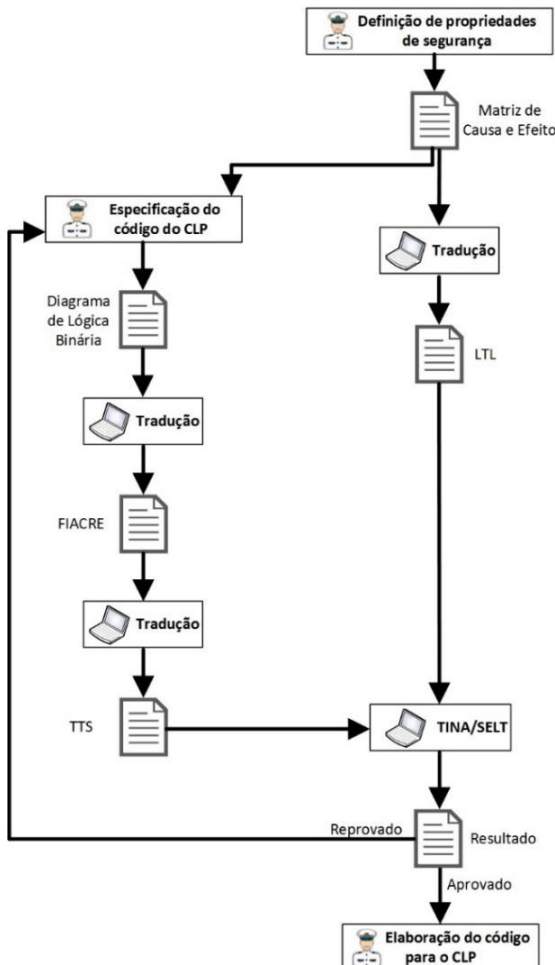


Figura 8 - Cadeia de verificação proposta no método.

3.1 A LINGUAGEM INTERMEDIÁRIA FIACRE

FIACRE significa *Format Intermédiaire pour les Architectures de Composants Répartis Embarqués* (Formato Intermediário para Arquiteturas de Componentes Distribuídos Embarcados). É um modelo intermediário formal para representar os aspectos comportamentais e temporais dos sistemas.

A sintaxe da linguagem FIACRE é descrita da seguinte forma (BERTHOMIEU et al., 2008):

- Processos: um processo FIACRE é uma máquina de estados composta por um conjunto finito de portas (sincronização com outros componentes ou processos), de parâmetros, de estados para controle interno, de variáveis locais e de transições atômicas entre os estados. As transições definem o comportamento do processo, sendo que cada transição possui um estado de partida e outro de chegada, e seu conteúdo é formado por uma estrutura de controle;
- Componente: Um componente descreve as interações entre processos ou componentes, de forma hierárquica, e possivelmente restringindo essas interações com requisitos de tempo e/ou prioridade. É composto por um conjunto finito de portas, de parâmetros, variáveis locais, de portas locais associadas a restrições temporais (canais de comunicação interligam as instâncias que compõem o componente), um conjunto finito de prioridades e uma composição paralela de instâncias de processos (descreve a interação entre as instâncias que compõem o componente);
- Tipos: os tipos de dados aceitos são : inteiro (int), natural (nat), booleano (bool) e tipos nativos: matrizes, uniões, pilhas, enumerações etc;
- Portas e canais de comunicação: as portas (*port*) fazem parte da interface de um processo FIACRE. São responsáveis pela comunicação, que pode ser síncrona ou não, e podem ser utilizadas para a troca de dados. Os canais (*channel*) são usados para definir um conjunto de tipos de dados aceitos por uma porta. Um perfil do tipo *none* sinaliza que a comunicação em questão é uma sincronização sem troca de valor;
- Comunicação: o FIACRE permite a comunicação síncrona entre processos e/ou componentes através das portas de comunicação. Estas portas permitem a sincronização pura ou a

transferência de um, ou diversos valores. Os operadores “?” e “!” determinam o sentido da comunicação, ou seja, se a transição envia ou recebe um dado; e

- Composição: a comunicação síncrona presente em FIACRE é resultado da composição paralela de um conjunto de instâncias. A comunicação entre os processos pode ser totalmente em paralelo, totalmente sincronizada ou com algumas ações sincronizadas.

Um programa FIACRE, para ser completo, necessita de um conjunto de tipos, canais de comunicação, processos e componentes.

3.2 TRADUÇÃO DE BLDs PARA FIACRE

Para modelar a lógica de funcionamento de CLP especificada por um BLD, esta dissertação propõe uma arquitetura de código em FIACRE baseada na composição paralela entre a instância do processo do ciclo de leitura do BLD (*diagram*) e as instâncias dos N processos dos blocos funcionais (BF) que podem existir, conforme a Figura 9.

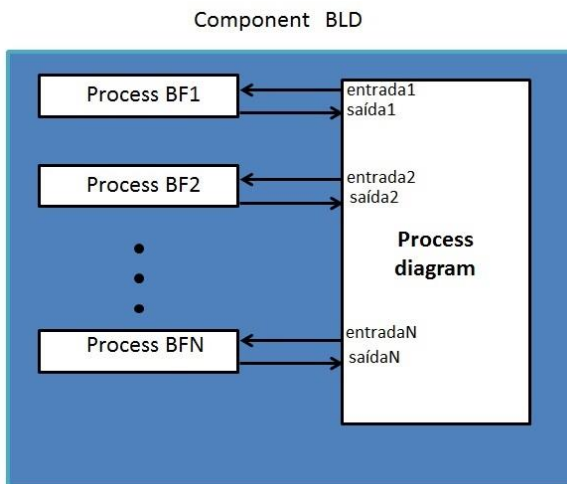


Figura 9 - Arquitetura do programa em FIACRE.

O programa do BLD em FIACRE é representado por um processo correspondendo ao ciclo de leitura (processo *diagram*). Os

blocos funcionais com temporização são representados por outros processos, que podem ser executados em paralelo para que não haja interrupção no processo *diagram*.

A Figura 10 apresenta a arquitetura do programa em FIACRE do BLD da Figura 7, em que o processo *diagram* é composto paralelamente com o processo do temporizador DI, único temporizador existente no BLD.

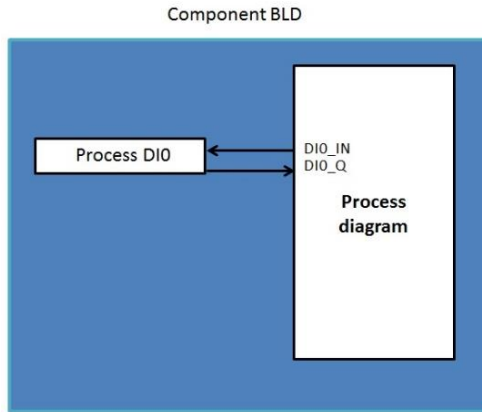


Figura 10 - Arquitetura do programa em FIACRE do BLD da Figura 7.

O Código 1 apresenta o código em FIACRE da composição das instâncias dos processos presentes no BLD da Figura 7. O operador “par” da linguagem FIACRE é o elemento que permite a composição paralela das instâncias dos processos correspondentes ao ciclo de leitura e dos blocos funcionais.

```

component BLD
is
  port portDIO_IN: in out bool in [0,0],
    portDIO_Q: in out bool in [0,0],
    portDIO_Timer: sync in [3,3]

  priority portDIO_Timer > portDIO_Q

  par * in
    diagram [portDIO_IN,portDIO_Q]
  || DI [portDIO_IN,portDIO_Q,portDIO_Timer]
  end

```

BLD

Código 1 - Component BLD para o diagrama lógico da Figura 7.

3.2.1 Modelo do ciclo de leitura em FIACRE

Diferente da linguagem FBD, que é uma linguagem de programação de CLPs, em que seu ciclo de execução é o próprio *scan cycle* do CLP, o BLD não é uma linguagem de programação. O fluxo de leitura (*flow of intelligence*) dos BLDs é normatizado pela ISA 5.2 (1992), que estabelece que deve ocorrer ordenadamente da esquerda para a direita, e de cima para baixo. As operações devem ser calculadas sequencialmente porque o BLD especifica a lógica a ser implementada em CLPs, que funcionam em *scan cycles* para simular o comportamento concorrente de circuitos digitais através de um único processador computacional.

Para representar o comportamento especificado pela ISA 5.2, o fluxo de leitura é dividido em duas etapas: atualização das entradas e atualização das saídas. Primeiro, as variáveis de entrada são atualizadas. Posteriormente ocorre a etapa de atualização dos valores de saída, calculados pela aplicação das operações lógicas especificadas no BLD sobre os valores de entrada da etapa anterior, priorizando a atualização no sentido de cima para baixo. Feito isso, o fluxo de leitura pode recomeçar, caracterizando assim, um ciclo de leitura (Figura 11).

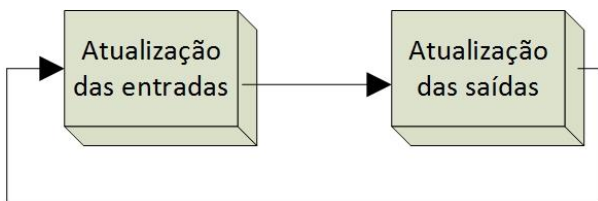


Figura 11 - Ciclo de leitura de BLDs.

Para representar o padrão do ciclo de leitura em FIACRE, são criados estados para atualização das entradas e das saídas. Para garantir que os valores nas saídas estão sempre relacionadas com os valores das entradas ao final de cada ciclo, a transição entre os estados de atualização das entradas e de atualização das saídas são definidas como instantâneas (comando `wait[0,0]` em FIACRE). A Figura 12 apresenta o ciclo de leitura em FIACRE descrito, acrescido do estado *PROBE* e *WAITING*. O estado *PROBE* é o estado em que será realizado o *model checking* e que é posicionado após o estado de atualização do efeito que se deseja verificar, garantindo que todas as variáveis relacionadas ao efeito estão atualizadas no estado. O estado *WAITING* representa o

reinício do ciclo de leitura dos BLDs, sendo sempre posicionado ao final do código. Como a norma não estabelece um padrão para o reinício do ciclo de leitura dos BLDs, ele foi suposto como sendo variável. O intervalo de tempo para reinício do ciclo de leitura é gerado ao final de cada ciclo, pela não inclusão do comando *wait* na transição entre o estado *WAITING* e de atualização das entradas. A não inclusão do comando *wait* na transição faz com que todas as possibilidades de intervalos de tempos ocorram. Assim, as propriedades validadas para o modelo independem do *scan cycle* do CLP escolhido para implementação da lógica especificada pelo BLD.

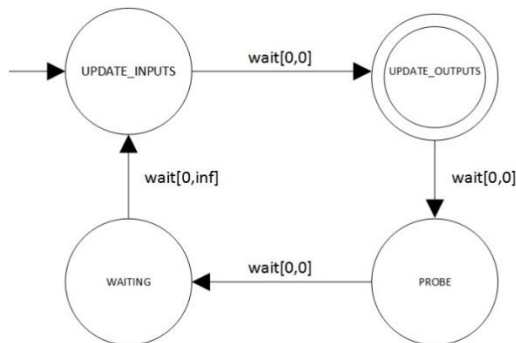


Figura 12 - Ciclo de leitura de BLDs em FIACRE.

Seguindo a metodologia proposta, um trecho do código FIACRE da tradução do BLD da Figura 7 está apresentado no Código 2.

```

init to UPDATE_OUTPUTS0
from UPDATE_INPUTS0
...
to UPDATE_OUTPUTS0
from UPDATE_OUTPUTS0
DEMANDA_FORCADA_MCP:=((DEMANDA_FORCADA_MCP or
(DEMANDA_MCP_MAIOR_600 and not(ER_DESACOPLADA) and
PRDA_EMERG_MCP)) and not(MCP_OFF));
portDIO_IN! ((DEMANDA_FORCADA_MCP or
(DEMANDA_MCP_MAIOR_600 and not(ER_DESACOPLADA) and
PRDA_EMERG_MCP)) and not(MCP_OFF));
to PROBE
from PROBE
wait[0,0];
to WAITING
from WAITING
to UPDATE_OUTPUTS1
from UPDATE_OUTPUTS1
portDIO Q? DIO Q;

```

```

MCP_STOPPING:= ((MCP_STOPPING or (DI0_Q and
(ER_DESACÓPLADA and PRDA_EMERG_MCP))) and not(MCP_OFF));
to UPDATE_OUTPUTS2
from UPDATE_OUTPUTS2
MCP_OFF:= (MCP_STOPPING and ROTACAO_MCP_IGUAL_0);
wait[0,0];
to UPDATE_INPUTS0

```

Código 2 - Trecho do código FIACRE da tradução do BLD da Figura 7.

3.2.2 Modelo de ciclo de leitura com blocos funcionais em FIACRE

Para o caso de BLDs com blocos funcionais do tipo temporizadores ou contadores, é necessário realizar a atualização das variáveis de entrada e de saída dos blocos funcionais em FIACRE. Isto é feito pelo sincronismo entre o processo do ciclo de execução e o do bloco funcional, com o envio de mensagem (!) do valor de entrada (IN) do bloco funcional do processo *diagram* para o processo do bloco funcional. Também é necessária uma etapa de sincronização para o recebimento (?), pelo processo *diagram*, do valor de saída (Q) oriundo do processo do bloco funcional.

A modelagem proposta é dividida em duas partes: a primeira delas durante o processo *diagram* permite a atualização das variáveis de entrada e saída do bloco funcional. A segunda parte é referente ao funcionamento de fato do bloco (por exemplo, DI, DT, PO), que na modelagem é representada por um processo em paralelo ao processo *diagram*. Quando um BLD contém um bloco funcional, as transições são modeladas de forma a permitir a execução paralela dos processos. Uma representa a parte de entrada do bloco, com a lógica de ativação atribuída a variável de entrada do bloco funcional, e a outra representa a saída do bloco funcional, com o valor da variável de saída do bloco funcional atribuída para uma variável do processo *diagram*. Entre estas duas transições ocorre a atualização dos dados do bloco funcional com o processo do bloco funcional. Esta etapa é necessária para a atualização dos dados no ciclo de leitura, e serve para qualquer tipo de bloco funcional com tempo.

Cabe aqui ressaltar uma característica do FIACRE, apresentada no Código 2, em que não é permitido o envio e recebimento de mensagem na mesma transição, sendo necessário reservar um estado com uma transição para o envio de mensagem e um estado seguinte com nova transição para recebimento de mensagem. Além disso, por definição, transições subsequentes a estados com envio ou recebimento são tratados em FIACRE como transições instantâneas (*wait[0,0]*).

3.2.3 Modelo do bloco temporizado DI

Levando-se em conta as características comportamentais dos temporizadores DI apresentada na Figura 6a, pode-se representar tal comportamento em FIACRE conforme o Código 3. Para isso são necessários três estados (*idle*, *running*, *elapsed*). O estado inicial *idle* está associado à transição que recebe o valor da variável de entrada IN do processo *diagram* e verifica se está ativa (valor booleano *true*), caso não esteja ativa, permanece no mesmo estado e mantém a saída Q desativada. O segundo estado é o *running* que é atingido por uma transição oriunda do estado *idle* caso a entrada IN estiver ativa. Estando no estado *running*, é possível zerar o temporizador, voltando para o estado *idle* caso a entrada IN seja desativada antes do tempo. A outra opção, a partir deste estado, consiste em contar o tempo até o valor determinado no bloco DI do BLD. Quando este é atingido, através da transição *portTimer*, a saída Q é ativada, ocorrendo a mudança para o estado chamado *elapsed*. No estado *elapsed*, é possível desativar a saída Q do temporizador caso a entrada IN for desativada, para voltar ao estado inicial. Todos os estados possuem um autolaço com a transição de atualização da variável Q, pelo envio de mensagem para o processo *diagram*. Esse processo DI é executado em paralelo ao processo *diagram*, atualizando os valores de entrada e saída do bloco por intermédio da etapa de leitura e atualização dos dados do bloco temporizador.

```

process DI
  [portIN: in bool, portQ: out bool, portTimer: sync]
  is
  states idle, running, elapsed
  var IN: bool := false,
      Q: bool := false
  init
    to idle
  from idle
  select
    portIN? IN;
    if IN then
      to running
    else
      loop
    end
  end
  []
  portQ! Q;
  loop
end

```

```

from running
  select
    portIN? IN;
    if not IN then
      to idle
    else
      loop
    end
  []
  portTimer;
  Q := true;
  to elapsed
  []
  portQ! Q;
  loop
end
from elapsed
  select
    portIN? IN;
    if not IN then
      Q := false;
      to idle
    else
      loop
    end
  []
  portQ! Q;
  loop
end

```

Código 3 - Processo do temporizador DI em FIACRE.

3.2.4 Modelo do bloco temporizado DT

De acordo com as características comportamentais dos temporizadores DT apresentada na Figura 6b, pode-se representar tal comportamento em FIACRE conforme o Código 4, onde são necessários três estados (*idle*, *running*, *active*). O primeiro é o estado *idle* que está associado à transição que recebe o valor da variável de entrada IN do processo diagram e verifica se está ativa (valor booleano true), caso não esteja ativa, permanece no mesmo estado e mantém a saída Q desativada. O segundo estado é o *running* que é atingido por uma transição oriunda do estado *idle* caso a entrada IN estiver ativa e mantém a saída Q ativada. Estando no estado *running*, caso a entrada IN permaneça ativa, permanece no mesmo estado, caso a entrada IN for desativada ocorrerá a transição para o estado *active*. Estando no estado *active*, poderá ocorrer uma transição para o estado *running* caso a entrada IN for ativada antes do tempo. A outra opção, a partir do estado *running*, consiste em contar o tempo até o valor determinado no bloco

DT do BLD. Quando este é atingido, através da transição *portTimer*, a saída *Q* é desativada, ocorrendo a mudança para o estado inicial *idle*. Todos os estados possuem um autolaço com a transição de atualização da variável *Q*, pelo envio de mensagem para o processo *diagram*. Esse processo DT é executado em paralelo ao processo *diagram*, atualizando os valores de entrada e saída do bloco por intermédio da etapa de leitura e atualização dos dados do bloco temporizador.

```

process DT
[portIN: in bool, portQ: out bool, portTimer:sync]
is states idle, active, running
var IN: bool :=false, Q: bool :=false
Init to idle
    from idle select
    portIN? IN;
    if IN then
        Q:= true; to active
    else
        loop
    end
[]
    portQ! Q;
    loop
end
    from active select
    portIN? IN;
    If not IN then
        to running
    else
        loop
    end
[]
    portQ! Q;
    loop
end
    from running select
    portIN? IN;
    if IN then
        to active
    else
        loop
    end
[]
    portTimer;
    Q:= false;
    to idle
[]
    portQ! Q;
    loop
end

```

Código 4 - Processo do temporizador DT em FIACRE.

3.2.5 Modelo do bloco temporizado PO

A partir das características comportamentais dos temporizadores PO apresentada na Figura 6c, pode-se representar tal comportamento em FIACRE conforme o Código 5, onde são necessário três estados (*idle*, *running*, *timeout*). O estado inicial *idle* é o estado que está associado à transição que recebe o valor da variável de entrada IN do processo *diagram* e verifica se está ativa (valor booleano *true*), caso não esteja ativa, permanece no mesmo estado e mantém a saída Q desativada. O segundo estado é o *running* que é atingido por uma transição oriunda do estado *idle* caso a entrada IN estiver ativa, e mantém a saída Q ativada. Estando no estado *running*, permanecerá no mesmo estado, até que o tempo especificado no bloco PO do BLD seja atingido e ocorrer a transição *portTimer*, a saída Q é desativada, ocorrendo a mudança para o estado *timeout*. Estando no estado *timeout* só ocorrerá a transição para o estado inicial *idle* quando a entrada IN for desativada. Todos os estados possuem um autolaço com a transição de atualização da variável Q, pelo envio de mensagem para o processo *diagram*.

```

process PO
[portIN: in bool, portQ: out bool, portTimer:sync]
is states idle, running, timeout

var IN: bool :=false, Q: bool :=false

init to idle
  from idle select
    portIN? IN;
    If IN then
      Q:=true; to running
    else
      loop
    end
  []
  portQ! Q;
  loop
end
from running select
  portTimer;      Q:=false;
to timeout
[]
  portQ! Q;
  loop
end
from timeout select
  portIN? IN;
  if IN then
    loop

```

```
    else
      to idle
    end
  []
  portQ! Q;
  loop
end
```

Código 5 - Processo do temporizador PO em FIACRE.

3.2.6 Modelagem de entradas digitais e analógicas em FIACRE

Entradas digitais são modeladas em FIACRE de acordo com sua natureza booleana. Para que a variável booleana assuma todos os valores possíveis, é utilizada a sintaxe de FIACRE “*any*”. Conforme apresentado no Código 6, a variável booleana “MCP_EM_LOCAL” assumirá todos os valores possíveis (0 ou 1) para que a verificação aborde todas as possibilidades do modelo.

Quanto maior a quantidade de variáveis no modelo e de possibilidades para essas variáveis, maior será o espaço de estados gerado. Com isso, de forma a reduzir o espaço de estados gerados quando há entradas analógicas na MCE e no BLD, ao invés de se utilizar todo o *range* da variável analógica, é utilizado apenas os valores de fronteira. Assim, o código FIACRE contém uma constante do tipo *array* com o valor inteiro imediatamente inferior, igual e imediatamente superior a cada faixa presente na MCE e BLD. Por exemplo, supondo duas variáveis discretas (rotação \geq 600 e rotação $>$ 700) provenientes de um sinal analógico de rotação que varia de 0 a 1000 rpm. O código em FIACRE considera a faixa de operação como sendo apenas três valores inteiros de fronteira para cada variável discreta: 599, 600, 601, 699, 700 e 701, cobrindo toda a faixa de interesse. Para que a variável analógica assuma todos os valores possíveis, também é utilizada a sintaxe de FIACRE “*any*”. Conforme apresentado no Código 6, a variável “ROTACAO[ir]” assumirá todos os valores possíveis (599 ou 600 ou 601 ou 699 ou 700 ou 701), pois a variável “ir” recebe “*any*” a cada ciclo, fazendo com que a verificação aborde todas as possibilidades do modelo.

```

const ROTACAO: array 6 of nat is [599,600,601,699,700,701]
...
process diagram
...
  var ir:0..5:=0,
      MCP_EM_LOCAL: bool := false,
...
init to UPDATE_OUTPUTS
  from UPDATE_INPUTS0
  ir:=any;
  MCP_EM_LOCAL:=any;
...
  to UPDATE_INPUTS1
  from UPDATE_INPUTS1
  if ROTACAO[ir]>=600 then
...

```

Código 6 - Procedimento para tratamento de entradas discretas e analógicos em FIACRE.

3.3 TRADUÇÃO DA MCE PARA LTL

A MCE representando especificações de segurança consiste de ações (efeito) que devem ser tomadas quando determinadas condições (causa) são satisfeitas. Por exemplo, a abertura de uma válvula de segurança, caso dois sensores de pressão alta estiverem ativos.

Dois tipos de falhas podem ser definidas para cada efeito para fins de organização das propriedades de segurança. Uma falha segura (FS) (1) ocorre quando existe o efeito, mas não existe a causa. Por exemplo, a válvula de segurança abriria sem que os dois sensores de pressão alta estivessem ativos. Uma falha perigosa (FP) (2) ocorre quando existe a causa, mas não existe efeito. Por exemplo, a válvula não seria aberta mesmo com os sensores de pressão alta ativos.

$$FS = Efeito \wedge \neg Causa \quad (1)$$

$$FP = \neg Efeito \wedge Causa \quad (2)$$

As propriedades verificadas com *model checking* são as propriedades livre de falha segura (LFS) e livre de falha perigosa (LFP). A verificação de propriedades LFS e LFP é efetuada em um estado específico (*probe*) do ciclo de leitura do processo *diagram*. Além disso, as propriedades são verificadas de forma individual para cada efeito da MCE. As fórmulas baseadas na sintaxe LTL das propriedades LFS e

LFP estão apresentadas em (3) e (4). Na propriedade LFS se verifica se sempre não haverá, em um estado específico (*probe*), o efeito ativo e a causa desativada. Na propriedade LFP se verifica se sempre não haverá, em um estado específico (*probe*), o efeito desativado e a causa ativada.

$$LFS = \square \neg (probe \wedge \neg causa \wedge efeito) \quad (3)$$

$$LFP = \square \neg (probe \wedge causa \wedge \neg efeito) \quad (4)$$

Por vezes, causas podem ser formadas por lógicas que utilizam a simbologia de ativação e desativação do efeito. Nesse caso, a definição do valor atualizado da causa necessita de um passo intermediário para sua definição, pois estão relacionadas com o valor do efeito no ciclo de leitura anterior. Sendo assim, uma causa formada por lógicas de ativação e desativação é definida da seguinte forma:

$$causa = \left\{ \begin{array}{c} (efeito \vee ("lógicas que ativam o efeito")) \\ \wedge \\ \neg ("lógicas que desativam o efeito") \end{array} \right\} \quad (5)$$

Onde *efeito*' é o valor do efeito no ciclo anterior.

Por exemplo, a tradução do primeiro efeito (DEMANDA_FORCADA_MCP) da MCE da Tabela 10, que possui simbologia de ativação e desativação de efeito, para LTL está explicitada no Código 7.

```

causa_1 = ((DEMANDA_FORCADA_MCP or (not(ER_DESACOPLADA)
and PRDA_EMERG_MCP and DEMANDA_MCP_MAIOR_600)) and
not(MCP_OFF))
efeito_1 = (DEMANDA_FORCADA_MCP)
Livre_falha_perigosa_1 = [] not(probe and causa_1 and
not(efeito_1))
Livre_falha_segura_1 = [] not(probe and not(causa_1) and
efeito_1)

```

Código 7 - Tradução do primeiro efeito da MCE para LTL.

Caso a MCE possua entradas temporizadas, é necessário a utilização de um contador temporal em substituição à entrada temporizada, pois como apresentado anteriormente, em LTL o tempo é representado de forma implícita através dos seus operadores lógicos. A Figura 13 apresenta o modelo de contador temporal para entradas

temporizadas utilizado neste trabalho. Sendo assim, uma variável temporizada é substituída pelo estado *VarTrueTimed*, estado esse que se terá certeza que a variável temporizada da MCE permaneceu ativa por no mínimo o tempo estipulado na MCE. O código em FIACRE do contador temporal está apresentado no Código 8.

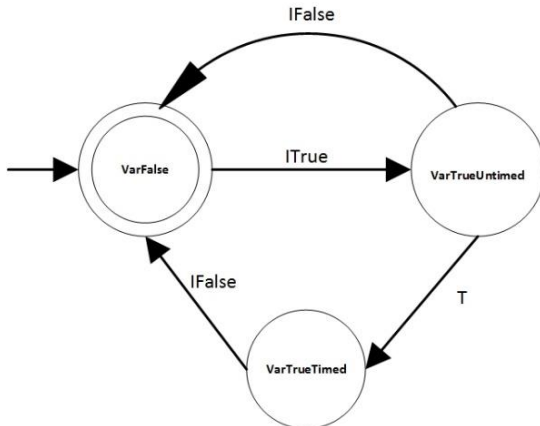


Figura 13 - Contador temporal para entradas temporizadas.

```

process timed_input [ITrue,IFalse,T: sync] is
  states VarFalse , VarTrueUntimed , VarTrueTimed
  init to VarFalse
  from VarFalse
    ITrue; to VarTrueUntimed
  from VarTrueUntimed select
    T; to VarTrueTimed
    [] IFalse; to VarFalse
  end
  from VarTrueTimed
    IFalse; to VarFalse

component obs_DI is
  port ITrue: sync in [0,0] is "condições que ativam
  ITrue",
  IFalse: sync in [0,0] is "condições que ativam
  IFalse",
  T: sync in ["intervalo","intervalo"]
  priority IFalse > T
  par
    timed_input [ITrue,IFalse,T]
  end
  obs_DI > BLD
  
```

Código 8 - Código FIACRE para o contador temporal.

Por exemplo, o Código 9 apresenta, em FIACRE, as propriedades em LTL para *model checking* do segundo efeito da MCE (MCP_STOPPING) expressa na Tabela 10, que é um caso em que é necessário o uso de contador temporal.

```

property OBS_DI0 is ltl (obs_DI0/1/state varTrueTimed)
property causa_2 is ltl ((MCP_STOPPING or ((ER_DESACOPLADA and
PRDA_EMERG_MCP) or (DEMANDA_FORCADA_MCP and OBS_DI0))) and
not(MCP_OFF))
property efeito_2 is ltl (MCP_STOPPING)

property Livre_Falha_Perigosa_2 is ltl [] not(probe and
causa_2 and not(efeito_2))
assert Livre_Falha_Perigosa_2

property Livre_Falha_Segura_2 is ltl [] not(probe and
not(causa_2) and efeito_2)
assert Livre_Falha_Segura_2

```

Código 9 - Propriedade em LTL utilizando o contador temporal.

3.4 ESTRATÉGIAS DE REDUÇÃO DO PROBLEMA DE EXPLOÇÃO COMBINACIONAL

Uma das principais limitações do *model checking* é a possibilidade de ocorrência de explosão combinacional de estados, que acontece quando o espaço de estados do sistema possui um tamanho maior do que é possível representar com a memória disponível nos sistemas computacionais. No modelo proposto, a explosão combinacional pode ocorrer devido à cada causa da MCE poder variar por "any", provocando a multiplicação do número de estados a serem verificados. Várias alternativas podem ser utilizadas para contornar o problema, que podem ser desde alternativas de representação de estados que explorem a regularidade entre eles, até a abstração dos modelos, no intuito de reduzi-los somente à parte que é relevante a propriedade verificada (CLARKE et al., 2001). Este trabalho aborda a estratégia de redução por cone de influência (CLARKE et al., 1999)(DARVAS et al., 2014) e realiza uma nova proposta de estratégia baseada na estrutura da MCE, intitulada de redução por decomposição.

3.4.1 Redução por cone de influência

A estratégia de redução por cone de influência (CLARKE; GRUMBERG; PELED, 1999) propõe realizar a verificação das propriedades de cada efeito da MCE utilizando somente as entradas que realmente influenciam no efeito desejado e descartando as variáveis que não influenciam. Tal estratégia possibilita redução no número de estados do sistema/subsistema cujo efeito se deseja verificar, minimizando o problema de explosão combinacional.

Exemplificando a estratégia de redução por cone de influência, a Figura 14 apresenta o cone de influência da MCE da Tabela 12.

Tabela 12 - Exemplo de MCE

AI	DI	VOTING	TAG		Q_01	Q_02	Q_03	Q_04	NOTES
	IN_01			1	X	A1	X		
	IN_02			2	T10				
	IN_03			3		A1			
	IN_04			4		T5			
	IN_05			5		X		X	
	IN_06A	1003		6			X		
	IN_06B	2003		7				X	
	IN_06C			8					

Conforme apresentado na Figura 14, não são todas as entradas que estão relacionadas a cada efeito. Por exemplo, para o efeito Q_01, somente o influenciam as entradas IN_01 e IN_02, e a entrada e saída do temporizador DI0. Portanto, para realização do *model checking* para o efeito Q_01, somente estas variáveis são levadas em conta, pois não há necessidade de manter variáveis que não influenciam no efeito Q_01.

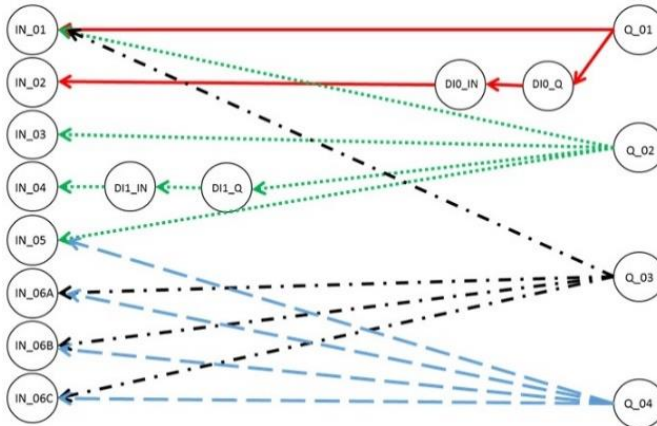


Figura 14 - Cone de influência para a MCE da Tabela 12.

Para a MCE da Tabela 10, o cone de influência (Figura 15) apresenta que todas as variáveis estão relacionadas entre si, logo a utilização da estratégia de redução por cone de influência para essa aplicação não permite redução no espaço de estados gerados.

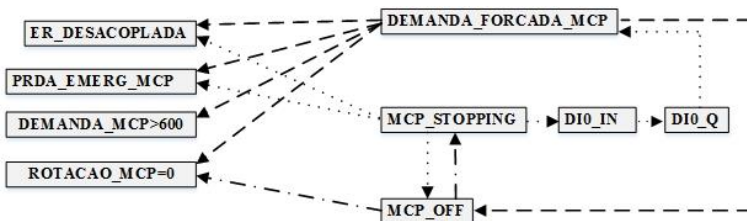


Figura 15 - Cone de influência da MCE da Tabela 10.

3.4.2 Redução por decomposição

Em alguns casos a estratégia de redução por cone de influência não é suficiente para reduzir o espaço de estados gerados e impedir explosão combinacional. Então, outra estratégia proposta é a redução por decomposição, que se baseia na decomposição das propriedades a serem verificadas. A estratégia de redução por decomposição parte do princípio de que as lógicas de uma causa podem ser expressadas, de forma generalizada, por combinações de lógicas *and* e *or*, conforme (6). Utilizando as leis de Morgan expressadas de (7) a (10) e a regra de equivalência para LTL expressada em (11) podem-se apresentar as

manipulações algébricas que possibilitam a estratégia de redução por decomposição, conforme apresentado a seguir.

$$causa = (A_1 \wedge \dots \wedge A_n) \vee (O_1 \vee \dots \vee O_m) \quad (6)$$

$n, m \in \mathbb{N}$

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r) \quad (7)$$

$$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r) \quad (8)$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q \quad (9)$$

$$\neg(p \wedge q) \equiv \neg p \vee \neg q \quad (10)$$

$$\Box(p \wedge q) \equiv \Box p \wedge \Box q \quad (11)$$

Para realizar a decomposição das propriedades LFS e LFP, o estado *probe* é descartado por ser somente o estado em que se deseja realizar a verificação, não tendo relação direta com as propriedades verificadas. Logo, utilizando (4), sem o estado *probe*, pode-se decompor a propriedade LFP da seguinte forma:

$$LFP \equiv \Box \neg (\neg Efeito \wedge Causa)$$

usando (5)

$$\equiv \Box \neg (\neg Efeito \wedge ((A_1 \wedge \dots \wedge A_n) \vee (O_1 \vee \dots \vee O_m)))$$

usando (9)

$$\equiv \Box (Efeito \vee \neg ((A_1 \wedge \dots \wedge A_n) \vee (O_1 \vee \dots \vee O_m)))$$

usando (9)

$$\equiv \Box (Efeito \vee (\neg(A_1 \wedge \dots \wedge A_n) \wedge \neg(O_1 \vee \dots \vee O_m)))$$

Usando (11), obtém-se (12):

$$\text{LFP} \equiv \left\{ \begin{array}{c} \boxed{\square}(\text{Efeito} \vee \neg A_1 \vee \dots \vee \neg A_n) \\ \wedge \\ \boxed{\square}(\text{Efeito} \vee \neg O_1) \\ \wedge \\ \boxed{\square}(\text{Efeito} \vee \neg O_2) \\ \wedge \\ \vdots \\ \wedge \\ \boxed{\square}(\text{Efeito} \vee \neg O_m) \end{array} \right\} \quad (12)$$

Para a propriedade LFS (3), sem o estado *probe*, pode-se reescrever a propriedade da seguinte forma:

$$\text{LFS} \equiv \boxed{\square} \neg (\text{Efeito} \wedge \neg \text{Causa})$$

usando (5)

$$\equiv \boxed{\square} \neg (\text{Efeito} \wedge \neg ((A_1 \wedge A_2 \wedge \dots \wedge A_n) \vee (O_1 \vee O_2 \vee \dots \vee O_m)))$$

usando (9)

$$\equiv \boxed{\square} (\neg \text{Efeito} \vee ((A_1 \wedge A_2 \wedge \dots \wedge A_n) \vee (O_1 \vee O_2 \vee \dots \vee O_m)))$$

$$\equiv \boxed{\square} (\neg \text{Efeito} \vee (O_1 \vee O_2 \vee \dots \vee O_m) \vee (A_1 \wedge A_2 \wedge \dots \wedge A_n))$$

Supondo: $\Delta = \neg \text{Efeito} \vee O_1 \vee O_2 \vee \dots \vee O_m$

logo:

$$\equiv \boxed{\square} (\Delta \vee (A_1 \wedge A_2 \wedge \dots \wedge A_n))$$

usando (6)

$$\equiv \boxed{\square} ((\Delta \vee A_1) \wedge (\Delta \vee A_2) \wedge \dots \wedge (\Delta \vee A_n))$$

usando (10), obtém-se:

$$\text{LFS} \equiv \left\{ \begin{array}{c} \boxed{\Delta \vee A_1} \\ \wedge \\ \boxed{\Delta \vee A_2} \\ \wedge \\ \vdots \\ \wedge \\ \boxed{\Delta \vee A_n} \end{array} \right\} \quad (13)$$

Sendo assim, a redução por decomposição possibilita a verificação das propriedades em subconjuntos, onde cada subconjunto consiste das lógicas precedidas pelo operador LTL “sempre” ($\boxed{}$) nas equações (12) e (13), reduzindo o problema de explosão combinacional pela diminuição do espaço de estados gerados no *model checking*, devido ao descarte das entradas não relacionadas a cada subconjunto.

Para o caso particular da causa formada somente por combinações de “and” pode-se realizar a decomposição, obtendo (14) e (15). Para a propriedade LFS:

$$\text{LFS} \equiv \boxed{\text{Efeito} \wedge \neg \text{Causa}}$$

$$\text{usando } \text{Causa} = (A_1 \wedge A_2 \wedge \dots \wedge A_n)$$

$$\text{LFS} \equiv \boxed{\neg(\text{Efeito} \wedge \neg(A_1 \wedge A_2 \wedge \dots \wedge A_n))}$$

usando (9)

$$\equiv \boxed{\neg \text{Efeito} \vee (A_1 \wedge A_2 \wedge \dots \wedge A_n)}$$

usando (6)

$$\equiv \boxed{(\neg \text{Efeito} \vee A_1) \wedge (\neg \text{Efeito} \vee A_2) \wedge \dots \wedge (\neg \text{Efeito} \vee A_n)}$$

usando (10), obtém-se:

$$\text{LFS} \equiv \left\{ \begin{array}{c} \boxed{\neg(\text{Efeito} \vee A_1)} \\ \wedge \\ \boxed{\neg(\text{Efeito} \vee A_2)} \\ \wedge \\ \vdots \\ \wedge \\ \boxed{\neg(\text{Efeito} \vee A_n)} \end{array} \right\} \quad (14)$$

Para a propriedade LFP:

$$\text{LFP} \equiv \boxed{\neg(\neg\text{Efeito} \wedge \text{Causa})}$$

$$\text{usando } \text{Causa} = (A_1 \wedge A_2 \wedge \dots \wedge A_n)$$

$$\text{LFP} \equiv \boxed{\neg(\neg\text{Efeito} \wedge (A_1 \wedge A_2 \wedge \dots \wedge A_n))}$$

usando (9)

$$\equiv \boxed{(\text{Efeito} \vee \neg(A_1 \wedge A_2 \wedge \dots \wedge A_n))}$$

usando (6), obtém-se:

$$\text{LFP} \equiv \boxed{(\text{Efeito} \vee \neg A_1 \vee \dots \vee \neg A_n)} \quad (15)$$

As expressões (14) e (15) apresentam que a decomposição de lógicas "and" só permite a verificação em subconjuntos para a propriedade LFS. Caso se faça a verificação exaustiva em subconjuntos para a propriedade de LFP, não se poderá certificar exaustivamente que o sistema está LFP, pois nesse caso, pode-se perder a influência de uma ou mais entradas no efeito que se está verificando. Esta limitação ainda não foi contornada e será motivo de trabalhos futuros.

Para o caso particular da causa formada somente por combinações de "or" pode-se realizar a decomposição, obtendo (16) e (17). Para a propriedade LFS:

$$\text{LFS} \equiv \boxed{\neg(\text{Efeito} \wedge \neg\text{Causa})}$$

$$\text{usando } \text{Causa} = (O_1 \vee O_2 \vee \dots \vee O_m)$$

$$\text{LFS} \equiv \boxed{\neg(\text{Efeito} \wedge \neg(O_1 \vee O_2 \vee \dots \vee O_m))}$$

usando (9), obtém-se:

$$\text{LFS} \equiv \Box(\neg\text{Efeito} \vee O_1 \vee O_2 \vee \dots \vee \neg O_m) \quad (16)$$

Para a propriedade LFP:

$$\text{LFP} \equiv \Box(\neg(\neg\text{Efeito} \wedge \text{Causa}))$$

$$\text{usando } \text{Causa} = (O_1 \vee O_2 \vee \dots \vee O_m)$$

$$\text{LFP} \equiv \Box(\neg(\neg\text{Efeito} \wedge (O_1 \vee O_2 \vee \dots \vee O_m)))$$

usando (9)

$$\equiv \Box(\text{Efeito} \vee \neg(O_1 \vee O_2 \vee \dots \vee O_m))$$

usando (8)

$$\equiv \Box(\text{Efeito} \vee (\neg O_1 \wedge \neg O_2 \wedge \dots \wedge \neg O_m))$$

usando (6)

$$\equiv \Box(((\text{Efeito} \vee \neg O_1) \wedge (\text{Efeito} \vee \neg O_2) \wedge \dots \wedge (\text{Efeito} \vee \neg O_m)))$$

usando (10), obtém-se:

$$\text{LFP} \equiv \left\{ \begin{array}{c} \Box(\text{Efeito} \vee \neg O_1) \\ \wedge \\ \Box(\text{Efeito} \vee \neg O_2) \\ \wedge \\ \vdots \\ \wedge \\ \Box(\text{Efeito} \vee \neg O_m) \end{array} \right\} \quad (17)$$

As expressões (16) e (17) apresentam que a decomposição de lógicas "or" só permite a verificação em subconjuntos para a propriedade LFP. Caso se faça a verificação exaustiva em subconjuntos para a propriedade de LFS, não se poderá certificar exaustivamente que o sistema está LFS, pois nesse caso, pode-se perder a influência de uma ou mais entradas no efeito que se está verificando. Esta limitação também ainda não foi solucionada e será motivo de estudos futuros.

3.5 APRESENTAÇÃO DE CONTRAEXEMPLO

A apresentação de contraexemplo em caso de resultado falso é uma das importantes vantagens da verificação por *model checking*. O projetista deve ser capaz de interpretar o contraexemplo e identificar o motivo do erro no BLD.

Realizando *model checking* entre o BLD da Figura 7 e a MCE da Tabela 10, obtém-se a presença de uma FP, conforme apresentado na Tabela 13, no efeito MCP_STOPPING. O Apêndice A apresenta o código completo para este exemplo de aplicação.

Tabela 13 - Resultado do *model checking* realizado no BLD da Figura 7 e da MCE da Tabela 10.

Efeitos	Estados	Tempo de exec.	FS/FP/OK
DEMANDA_FORCADA_MCP	534	0.004s	0/0/2
MCP_STOPPING	544	0.004s	0/1/1
MCP_OFF	566	0.004s	0/0/2
Legenda: FS = Falha Segura FP = Falha Perigosa OK = Propriedades aprovadas			

O contraexemplo gerado pelo SELT (Figura 16) é apresentado de forma pouco usual para os programadores de CLPs, não cumprindo adequadamente sua função que é apresentar a sequência de eventos que se chega a uma condição de erro, para viabilizar sua correção. Então, de forma a tornar o contraexemplo do SELT de fácil interpretação para agilizar a correção de erros, o contraexemplo é apresentado na forma de diagrama de sinais (Figura 17).

```

operator Dangerous_Failure_Free_2 : prop
0.000s
FALSE

...
state 236: DI_1_sidle diagram_1_sPROBE timed_input_1_svarFalse diagram_1_vROTACAO_MCP_IGUAL_0
diagram_1_vPRDA_EMERG_MCP diagram_1_vER_DESACOPLADA diagram_1_vDEMANDA_MCP_MAIOR_600
-diagram_1_t26->
state 237: L.div DI_1_sidle diagram_1_sWAITING timed_input_1_svarFalse
diagram_1_vROTACAO_MCP_IGUAL_0 diagram_1_vPRDA_EMERG_MCP diagram_1_vER_DESACOPLADA
diagram_1_vDEMANDA_MCP_MAIOR_600
-diagram_1_t27->
state 238: DI_1_sidle diagram_1_sUPDATE_OUTPUTS2 timed_input_1_svarFalse
diagram_1_vROTACAO_MCP_IGUAL_0 diagram_1_vPRDA_EMERG_MCP diagram_1_vER_DESACOPLADA
diagram_1_vDEMANDA_MCP_MAIOR_600
-diagram_1_t28->
state 239: DI_1_sidle diagram_1_sUPDATE_INPUTS0 timed_input_1_svarFalse
diagram_1_vROTACAO_MCP_IGUAL_0 diagram_1_vPRDA_EMERG_MCP diagram_1_vER_DESACOPLADA
diagram_1_vDEMANDA_MCP_MAIOR_600
-diagram_1_t9->
state 240: DI_1_sidle diagram_1_sUPDATE_OUTPUTS0 timed_input_1_svarFalse
diagram_1_vPRDA_EMERG_MCP diagram_1_vER_DESACOPLADA diagram_1_vDEMANDA_MCP_MAIOR_600
-DI_1_t1_diagram_1_t24->
state 241: DI_1_sidle diagram_1_sUPDATE_OUTPUTS1 timed_input_1_svarFalse
diagram_1_vPRDA_EMERG_MCP diagram_1_vER_DESACOPLADA diagram_1_vDEMANDA_MCP_MAIOR_600
-DI_1_t2_diagram_1_t25->
state 242: DI_1_sidle diagram_1_sPROBE timed_input_1_svarFalse diagram_1_vPRDA_EMERG_MCP
diagram_1_vER_DESACOPLADA diagram_1_vDEMANDA_MCP_MAIOR_600
-diagram_1_t26->
state 246: L.div DI_1_sidle diagram_1_sWAITING timed_input_1_svarFalse diagram_1_vPRDA_EMERG_MCP
diagram_1_vER_DESACOPLADA diagram_1_vDEMANDA_MCP_MAIOR_600
[accepting all]
0.000s
operator Safe_Failure_Free_2 : prop
0.000s
TRUE
0.004s

```

Figura 16 - Contraexemplo fornecido pelo SELT.

A Figura 17 apresenta o contraexemplo fornecido pelo SELT já expressado na forma de diagrama de sinais, para o *model checking* realizado no BLD da Figura 7 e da MCE da Tabela 10.

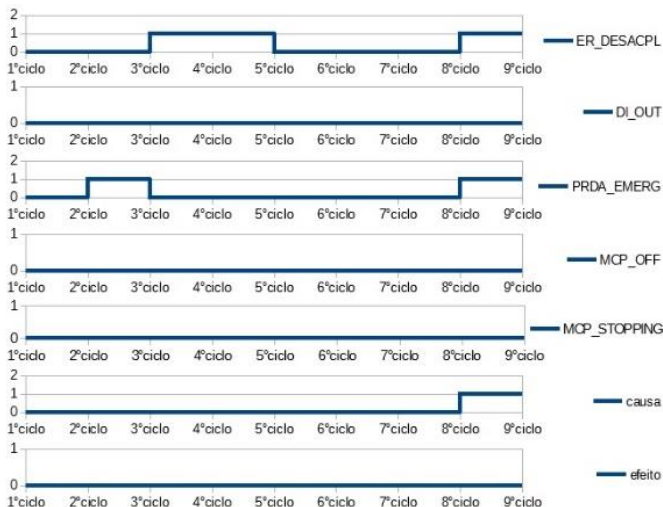


Figura 17 - Apresentação do contraexemplo como diagrama de sinais.

Como pode-se verificar na Figura 17, o erro é facilmente identificado no oitavo ciclo, pois existe a causa, mas não existe o efeito, configurando uma FP. Sendo assim, o BLD está reprovado e deve ser corrigido.

3.5.1 Interpretação dos resultados e correção de erros

A partir do contraexemplo na forma de diagramas sinais (Figura 17) e analisando a MCE da Tabela 10, verifica-se que a MCE determina que o efeito MCP_STOPPING deve estar ativo quando ER_DESACPL e PRDA_EMERG estiverem ativas ou a saída do temporizador estiver ativa, portanto, pelas condições no oitavo ciclo da Figura 17, o efeito deveria estar ativo. No BLD (Figura 7) identifica-se que o erro foi gerado por uma lógica da causa, que no BLD estava representada por uma lógica *and*, quando deveria ser *or*, pois no BLD o efeito é ativado quando ER_DESACPL, PRDA_EMERG e a saída do temporizador estiverem ativas simultaneamente. Interpretando o contraexemplo em diagrama de sinais, o erro pode ser encontrado e pode-se gerar o novo BLD corrigido (Figura 18) que também é submetido à metodologia proposta. A Tabela 14 apresenta o resultado do SELT de forma resumida, confirmando que o novo BLD proposto está livre de falhas, assim, atendendo às especificações de segurança e, portanto, está apto para servir de especificação para o código do CLP.

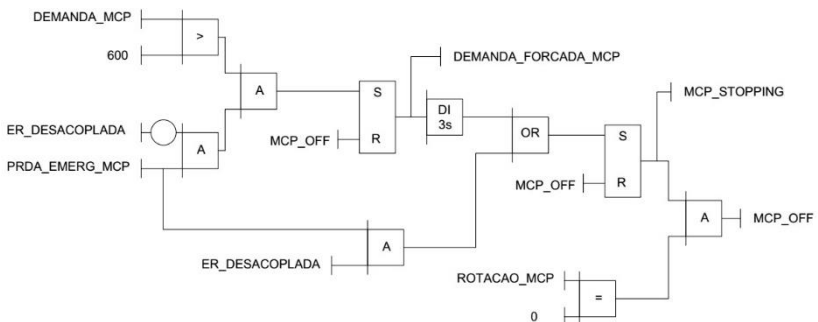


Figura 18 - BLD corrigido.

Tabela 14 - Resultados do estudo de caso.

Efeitos	Estados	Tempo de exec.	FS/FP/OK
DEMANDA_FORCADA_MCP	782	0s	0/0/2
MCP_STOPPING	954	0.004s	0/0/2
MCP_OFF	882	0.004s	0/0/2
Legenda: FS = Falha Segura FP = Falha Perigosa OK = Propriedades aprovadas			

3.6 CONCLUSÃO DO CAPÍTULO

Este capítulo teve foco na cadeia de verificação formal utilizada, apresentando a linguagem intermediária (FIACRE) entre os documentos de especificação e a linguagem interpretada pela ferramenta verificadora para realização de *model checking*.

O detalhamento dos modelos de tradução dos BLDs e da MCE para FIACRE possibilita automatizar o processo de verificação, agilizando consideravelmente o processo de certificação de que os BLDs possuem as propriedades de segurança desejadas.

Como o problema de explosão combinacional é uma das limitações do *model checking*, as estratégias para reduzir esse problema são essenciais para uma adequada utilização do método. O método proposto de redução por decomposição, que se baseia nas propriedades verificadas, foi um importante aliado em conjunto com a redução por cone de influência. Apesar da estratégia de redução por decomposição ter apresentado limitações em casos específicos, o uso das duas técnicas em conjunto viabiliza a verificação por *model checking* de grande parte das aplicações.

A apresentação do resultado da ferramenta verificadora em uma forma mais amigável aos projetistas, na forma de diagramas de sinais, também é uma importante contribuição deste trabalho. Facilitar a compreensão do contraexemplo torna ainda mais ágil o processo de correção de erros, justificando ainda mais a utilização da metodologia proposta.

4 UTILIZAÇÃO DA NOVA METODOLOGIA EM UMA APLICAÇÃO REAL DA MB

De forma a submeter a metodologia proposta neste trabalho a uma aplicação real de maior complexidade, a MB forneceu cinco BLDs e as especificações de segurança desejadas na forma textual.

Para utilização da nova metodologia proposta, os seguintes procedimentos devem ser executados:

- Interpretação da especificação textual e construção da MCE;
- Tradução da MCE para LTL;
- Tradução dos BLDs para FIACRE;
- Utilização das técnicas de redução por cone de influência e por decomposição;
- Execução do *Model Checking*;
- Apresentação do contraexemplo como diagramas de sinais em caso de resultado falso; e
- Correção dos erros em caso de resultado falso.

4.1 A ESPECIFICAÇÃO DA MB

A especificação textual fornecida pela MB consiste das condições necessária para o intertravamento para partida do MCP, para a partida normal do MCP, para a parada normal do MCP, para parada de emergência do MCP, e para acoplar e desacoplar a ER.

1. Intertravamento para Partida do MCP

O intertravamento para partida normal do MCP consiste das seguintes condições:

- Operação em remoto;
- A chave de bateria deve estar ligada
- O “starter” do motor deve encontrar-se na posição “off”.
- Engrenagem redutora desacoplada;
- Demanda do sistema em 600 rpm (ralenti);
- Leitura de rotação do motor igual a zero;
- Motor não pode estar em parada de emergência;
- Bomba de resfriamento do motor ligada;
- A temperatura de da água de resfriamento do motor deverá estar entre 45 e 85°C; e

- Ausência de quaisquer alarmes advindos do controlador do motor:
 - Alarme de baixa pressão de óleo de controle da ER;
 - Alarme de falha no carregamento da bateria do MCP;
 - Alarme de alta temperatura da água de resfriamento do MCP ($>85^{\circ}\text{C}$);
 - Alarme de baixa pressão de óleo lubrificante do MCP;
 - Alarme de nível baixo da água de resfriamento do MCP;
 - Alarme de filtro de óleo lubrificante do MCP obstruído;
 - Alarme de alta temperatura do óleo lubrificante do MCP; e
 - Alarme do MCP de espera entre partidas.

2. Procedimento para partida normal

Para a partida normal do MCP é necessário que não tenha ocorrido uma partida no intervalo de 13 segundos e que a rotação do motor não tenha superado 590 rpm. Além disso, é necessário que as seguintes condições ocorram simultaneamente por duração mínima de 3 segundo:

- A ER deverá estar desacoplada;
- O starter do MCP deverá estar na posição "off";
- O motor deverá estar parado (rotação igual a zero);
- O intertravamento para partida do MCP deverá estar cumprido; e
- O comando de partir deverá estar ativo no supervisor.

3. Intertravamento para Parada Normal

Para que seja habilitado o comando de parar o motor normalmente, é estipulado pelo fabricante que o motor deve permanecer em ralenti por 4 minutos e o MCP deverá estar em modo de operação "remoto".

4. Procedimento para Parada em Emergência

O MCP estará parado se a condição parando estiver ativa e a rotação chegar a zero. Permanecerá em modo parando se o botão de

parada de emergência for pressionado, a Engrenagem Reversora for desacoplada e se for forçada uma demanda acima de 600 rpm por no mínimo 3 segundos simultâneamente ao menos uma vez, até que o MCP esteja na condição parado. Também forçará uma demanda no MCP se o botão de parada de emergência for pressionado e a ER estiver acoplada e se demanda estiver determinada para 600 rpm simultâneamente, até que o MCP esteja na condição parado.

5. Intertravamento para Acoplar

As seguintes condições devem ser atendidas:

- É necessário aguardar um intervalo de 2 segundos entre mudanças de estados;
- Operação em remoto;
- Somente um comando e um estado de acoplamento lidos pelo sistema;
- Motor em ralenti;
- Ausência de quaisquer alarmes advindos da ER:
 - Alarme de baixa pressão do óleo de controle da ER;
 - Alarme de baixa pressão do óleo lubrificante da ER; e
 - Alarme de alta temperatura do óleo de controle da ER.

6. Intertravamento para Desacoplamento:

As seguintes condições devem ser atendidas

- É necessário aguardar um intervalo de 2 segundos entre mudanças de estados;
- Operação em remoto;
- Somente um comando e um estado de acoplamento lidos pelo sistema; e
- Motor em ralenti.

4.1.1 A MCE

Levando-se em conta todas essas especificações fornecidas, é possível elaborar a MCE (Tabela 15) contendo todas essas especificações de segurança desejadas.

Tabela 15 - MCE da aplicação da MB.

AI	DI	VOTING	TAG	TAG	INTERLOCK PARTIDA_MCP	COMANDO_START_MCP	ALARME_MCP_ESPERA_ENTRE_PARTIDAS	INTERLOCK_PARADA_NORMAL_MCP	DEMANDA_FORCADA_MCP	MCP_STOPPING	MCP_OFF	ER_2SEG_MESMO_ESTADO	INTERLOCK_ACOPLAR_AV_ER_2SEG_AV_AR	INTERLOCK_DESCOPLAR	NOTES
	ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER			1	NAI								NAI		
	ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP			2	NAI										
	ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP			3	NAI										
	ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP			4	NAI										
	ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP			5	NAI										
	ALARME_FIEMO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO			6	NAI										
	ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP			7	NAI										
ER	ACOPLADA_AV			8								T2			T2
ER	ACOPLADA_AR			9								T2			T2
ER	DISACOPLADA			10	AI	AI T2		NAI	AI+			T2			T2
	ALARME_MCP_ESPERA_ENTRE_PARTIDAS			11	NAI	NAI T2									
	PRDA_EMERG_MCP			12	NAI			AI+	AI+				NAI		
	MCP_EM_OPERACAO_LOCAL			13	NAI			NAI					NAI		
	STARTER_MCP_DLOFF			14	AI	AI T2									
	BOMBA_AGUA_SALGADA_LIGADA			15	AI										
	VALVULA_BOMBA_AGUA_SALGADA_ABERTA			16	AI										
	MANETE_EXTERNA_HABILITADA			17	AI										
	CHAVE_BATERIA_MCP_LIGADA			18	AI										
	TEMP_H2O_RESF_MCP		>=45	19	AI										
	TEMP_H2O_RESF_MCP		<=85	20	AI										
	DEMANDA_MCP		<=600	21			AI						AI		AI
	DEMANDA_MCP		<=600	22				AI+							
	DEMANDA_MCP		<=700	23			AI								
	DEMANDA_MCP		<=700	24									AI		AI
	DEMANDA_MCP		<=600	25	AI										
	ROTACAO_MCP		<=8	26	AI	AI T2					AI				
	ROTACAO_MCP		<=50	27		AI									
	ROTACAO_MCP		>=600	28			AI T240						AI		AI
	ROTACAO_MCP		<=700	29			AI T240								
	ROTACAO_MCP		<=700	30									AI		AI
	INTERLOCK_PARTIDA_MCP			31		AI T2									
	START_MCP_SUPERVISORIO			32		AI T2									
	COMANDO_START_MCP			33			AI T15								
	MCP_OFF			34				X-	X-						
	DEMANDA_FORCADA_MCP			35						F3+	AI				
	ER_2SEG_MESMO_ESTADO			36									AI		
	ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER			37									NAI		
	ER_2SEG_AV_AR			38										AI	
	ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER			39									NAI		

4.2 DIAGRAMAS DE LÓGICA BINÁRIA DA MB

Os BLDs fornecidos pela MB estão apresentados nas Figura 19 até a Figura 24, que consistem de BLDs de especificação de:

- Intertravamento de partida do MCP;
- Procedimento para partida normal do MCP;
- Intertravamento para parada normal do MCP;
- Procedimento para Parada em Emergência do MCP;
- Intertravamento para acoplar a ER; e
- Intertravamento para descoplar a ER.

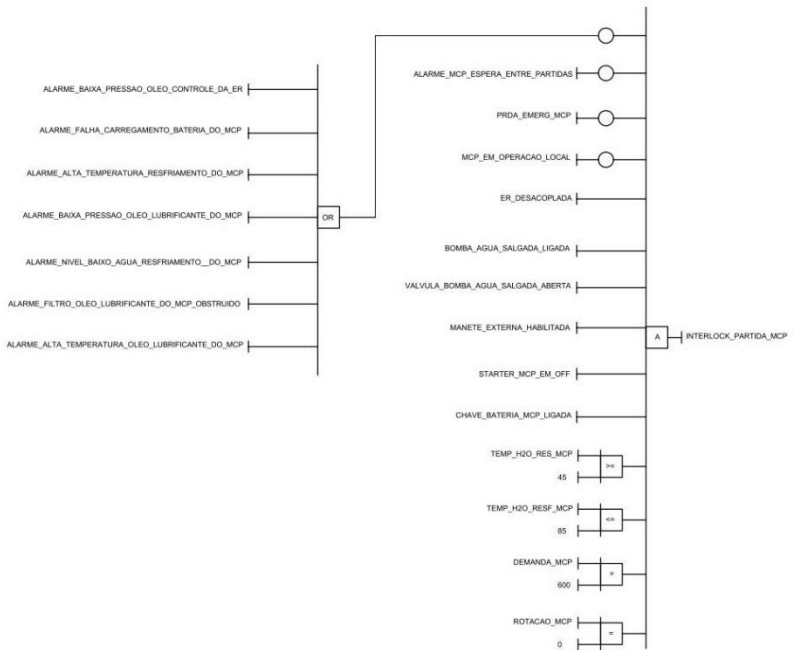


Figura 19 - BLD do Intertravamento de partida do MCP.

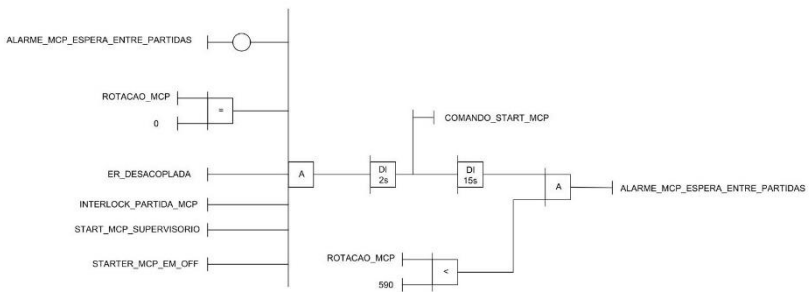


Figura 20 - BLD do procedimento para partida normal do MCP.

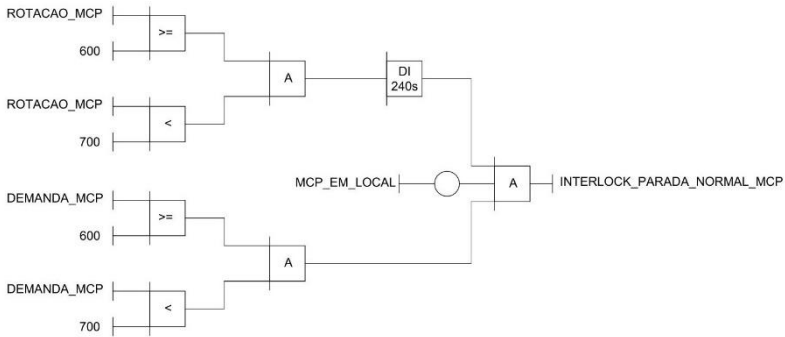


Figura 21 - BLD do Intertravamento para parada normal do MCP.

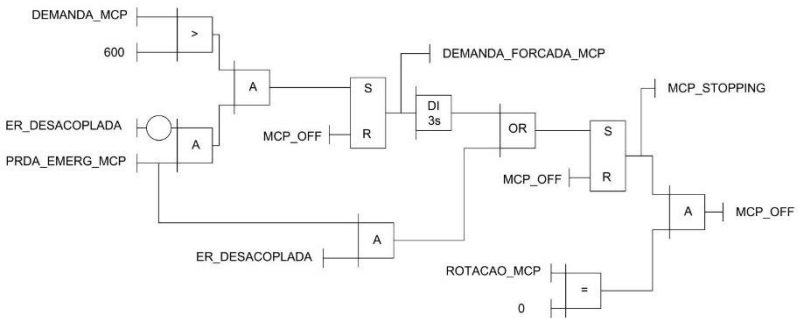


Figura 22 - BLD do procedimento para Parada em Emergência do MCP.

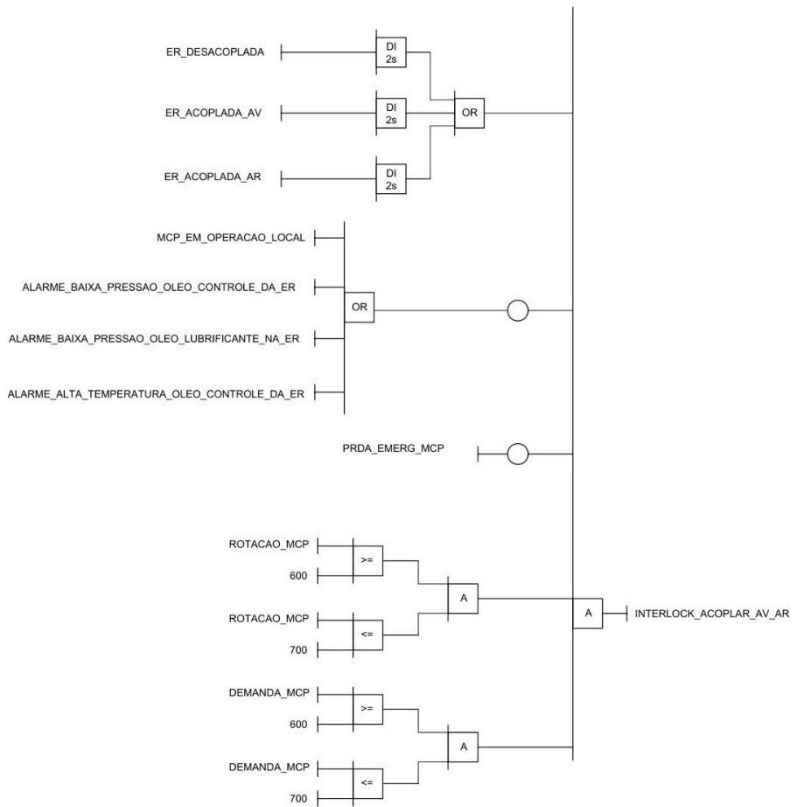


Figura 23 - BLD do Intertravamento para acoplar a ER.

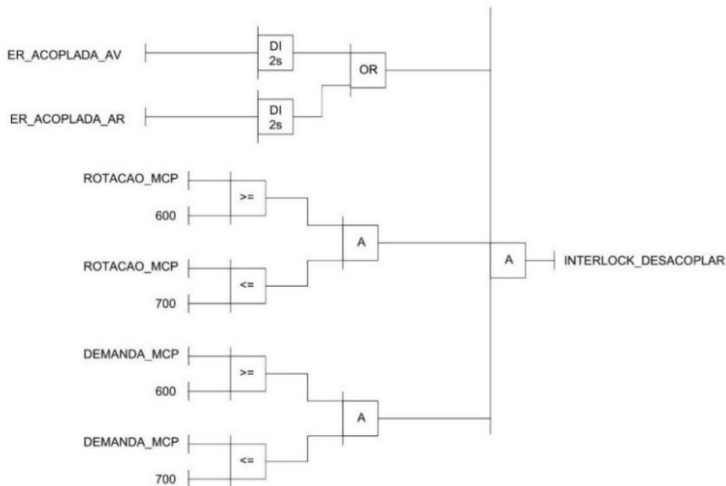


Figura 24 - BLD do Intertravamento para desacoplar a ER.

4.3 TRADUÇÃO DA MCE DA MB PARA LTL

Com as especificações de segurança expressadas na MCE da Tabela 15, pode-se apresentar as fórmulas LTL para cada efeito da MCE. Para o primeiro efeito da MCE da Tabela 15 (INTERLOCK_PARTIDA_MCP), pode-se expressar as propriedades de acordo com o Código 10.

property	causa_1	is	ltl
(not (ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER)			and
not (ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP)			and
not (ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP)			and
not (ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP)			and
not (ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP)			and
not (ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO)			and
not (ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP)			and
ER_DESACOPLADA and not (ALARME_MCP_ESPERA_ENTRE_PARTIDAS)			and
not (PRDA_EMERG MCP) and			
not (MCP_EM_OPERACAO_LOCAL)	and	STARTER_MCP_EM_OFF	and
BOMBA_AGUA_SALGADA_LIGADA			and
VALVULA_BOMBA_AGUA_SALGADA_ABERTA			and
MANETE_EXTERNA_HABILITADA and CHAVE_BATERIA_MCP_LIGADA			and
TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45			and
TEMP_H2O_RESF_MCP_MENOR_IGUAL_85 and DEMANDA_MCP_IGUAL_600 and ROTACAO_MCP_IGUAL_0)			and
property efeito_1 is ltl (INTERLOCK_PARTIDA_MCP)			

```

property Livre_Falha_Perigosa_1 is ltl [] not(probe and
causa_1 and not(efeito_1))
assert Livre_Falha_Perigosa_1
property Livre_Falha_Segura_1 is ltl [] not(probe and
not(causa_1) and efeito_1)
assert Livre Falha Segura 1

```

Código 10 - Código em FIACRE para *model checking* do primeiro efeito da MCE da Tabela 15.

Para o segundo efeito da MCE (COMANDO_START_MCP), pode-se expressar as propriedades de acordo com o Código 11.

```

property causa_2 is ltl (OBS_DIO)
property efeito_2 is ltl (COMANDO_START_MCP)

property Livre_Falha_Perigosa_2 is ltl [] not(probe and
causa_2 and not(efeito_2))
assert Livre_Falha_Perigosa_2
property Livre_Falha_Segura_2 is ltl [] not(probe and
not(causa_2) and efeito_2)
assert Livre Falha Segura 2

```

Código 11 - Código em FIACRE para *model checking* do segundo efeito da MCE da Tabela 15.

Para o terceiro efeito da MCE da Tabela 15 (ALARME_MCP_ESPERA_ENTRE_PARTIDAS), pode-se expressar as propriedades de acordo com o Código 12.

```

property causa_3 is ltl (ROTACAO_MCP_MENOR_590 and OBS_DI1)
property efeito_3 is ltl (ALARME_MCP_ESPERA_ENTRE_PARTIDAS)

property Livre_Falha_Perigosa_3 is ltl [] not(probe and
causa_3 and not(efeito_3))
assert Livre_Falha_Perigosa_3
property Livre_Falha_Segura_3 is ltl [] not(probe and
not(causa_3) and efeito_3)
assert Livre Falha Segura 3

```

Código 12 - Código em FIACRE para *model checking* do terceiro efeito da MCE da Tabela 15.

Para o quarto efeito da da Tabela 15 (INTERLOCK_PARADA_NORMAL_MCP), pode-se expressar as propriedades de acordo com o Código 13.

```

property causa_4 is ltl (not(MCP_EM_OPERACAO_LOCAL) and
DEMANDA_MCP_MAIOR_IGUAL_600 and DEMANDA_MCP_MENOR_700 and
((ROTACAO_MCP_MAIOR_IGUAL_600 and ROTACAO_MCP_MENOR_700) and
OBS_DI2))
property efeito_4 is ltl (INTERLOCK_PARADA_NORMAL_MCP)

property Livre_Falha_Perigosa_4 is ltl [] not(probe and
causa_4 and not(efeito_4))
assert Livre_Falha_Perigosa_4
property Livre_Falha_Segura_4 is ltl [] not(probe and
not(causa_4) and efeito_4)
assert Livre_Falha_Segura_4

```

Código 13 - Código em FIACRE para *model checking* do quarto efeito da MCE da Tabela 15.

Para o quinto efeito da MCE da Tabela 15 (DEMANDA_FORCADA_MCP), pode-se expressar as propriedades de acordo com o Código 14.

```

property causa_5 is ltl ((DEMANDA_FORCADA_MCP or
(not(ER_DESACOPLADA) and PRDA_EMERG_MCP and
DEMANDA_MCP_MAIOR_600)) and not(MCP_OFF))
property efeito_5 is ltl (DEMANDA_FORCADA_MCP)

property Livre_Falha_Perigosa_5 is ltl [] not(probe and
causa_5 and not(efeito_5))
assert Livre_Falha_Perigosa_5
property Livre_Falha_Segura_5 is ltl [] not(probe and
not(causa_5) and efeito_5)
assert Livre_Falha_Segura_5

```

Código 14 - Código em FIACRE para *model checking* do quinto efeito da MCE da Tabela 15.

Para o sexto efeito da MCE da Tabela 15 (MCP_STOPPING), pode-se expressar as propriedades de acordo com o Código 15.

```

property causa_6 is ltl ((MCP_STOPPING or ((ER_DESACOPLADA and
PRDA_EMERG_MCP) or (DEMANDA_FORCADA_MCP and OBS_DI3))) and
not(MCP_OFF))
property efeito_6 is ltl (MCP_STOPPING)

property Livre_Falha_Perigosa_6 is ltl [] not(probe and

```

```

causa_6 and not(efeito_6))
assert Livre_Falha_Perigosa_6
property Livre_Falha_Segura_6 is ltl [] not(probe and
not(causa_6) and efeito_6)
assert Livre Falha Segura 6

```

Código 15 - Código em FIACRE para *model checking* do sexto efeito da MCE da Tabela 15.

Para o sétimo efeito da MCE da Tabela 15 (MCP_OFF), pode-se expressar as propriedades de acordo com o Código 16.

```

property causa_7 is ltl (ROTACAO_MCP_IGUAL_0 and
DEMANDA_FORCADA_MCP)
property efeito_7 is ltl (MCP_OFF)

property Livre_Falha_Perigosa_7 is ltl [] not(probe and
causa_7 and not(efeito_7))
assert Livre_Falha_Perigosa_7
property Livre_Falha_Segura_7 is ltl [] not(probe and
not(causa_7) and efeito_7)
assert Livre Falha Segura 7

```

Código 16 - Código em FIACRE para *model checking* do sétimo efeito da MCE da Tabela 15.

O oitavo efeito da MCE da Tabela 15 se trata de uma variável auxiliar, logo será absorvido pelo efeito que a utiliza, que é o nono efeito (INTERLOCK_ACOPLAR_AV_AR), e pode-se expressar as propriedades de acordo com o Código 17.

```

property causa_8 is ltl ((OBS_DI4 or OBS_DI5 or OBS_DI6) and
not(ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER) and
not(PRDA_EMERG_MCP) and not(MCP_EM_OPERACAO_LOCAL) and
DEMANDA_MCP_MAIOR_IGUAL_600 and DEMANDA_MCP_MENOR_IGUAL_700
and ROTACAO_MCP_MAIOR_IGUAL_600 and
ROTACAO_MCP_MENOR_IGUAL_700 and
not(ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER) and
not(ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER))
property efeito_8 is ltl (INTERLOCK_ACOPLAR_AV_AR)

property Livre_Falha_Perigosa_8 is ltl [] not(probe and
causa_8 and not(efeito_8))
assert Livre_Falha_Perigosa_8
property Livre_Falha_Segura_8 is ltl [] not(probe and
not(causa_8) and efeito_8)
assert Livre Falha Segura 8

```

Código 17 - Código em FIACRE para *model checking* do nono efeito da MCE da Tabela 15.

O décimo efeito da MCE da Tabela 15 também se trata de uma variável auxiliar, logo será absorvido pelo efeito que a utiliza, que é o décimo primeiro efeito da MCE (INTERLOCK_DESACOPLAR), e pode-se expressar as propriedades de acordo com o Código 18.

```

property causa_9 is ltl ((OBS_DI7 or OBS_DI8) and
DEMANDA_MCP_MAIOR_IGUAL_600 and DEMANDA_MCP_MENOR_IGUAL_700
and ROTACAO_MCP_MAIOR_IGUAL_600 and
ROTACAO_MCP_MENOR_IGUAL_700 )
property efeito_9 is ltl (INTERLOCK_DESACOPLAR)

property Livre_Falha_Perigosa_9 is ltl [] not(probe and
causa_9 and not(efeito_9))
assert Livre_Falha_Perigosa_9
property Livre_Falha_Segura_9 is ltl [] not(probe and
not(causa_9) and efeito_9)
assert Livre_Falha_Segura_9

```

Código 18 - Código em FIACRE para *model checking* do décimo primeiro efeito da MCE da Tabela 15.

4.4 TRADUÇÃO DOS DIAGRAMAS LÓGICOS DA MB PARA FIACRE

O código completo da aplicação da MB, tando com os BLDs traduzidos para FIACRE quanto com a MCE traduzida, está apresentado no apêndice B por possuir um volume muito grande de informações.

Para apresentar a estratégia de tradução dos BLDs da MB, a apresentação se dará de forma particionada. O Código 19 apresenta o trecho da tradução para FIACRE do primeiro BLD (Intertravamento de partida do MCP). O estado *probe* é posicionado para verificar o efeito INTERLOCK_PARTIDA_MCP.

```

...
from UPDATE OUTPUTS0
INTERLOCK_PARTIDA_MCP:= (not (ALARME_BAIXA_PRESSAO_OLEO_CO
NTROLE_DA_ER or ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP
or ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP or
ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP or
ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP or
ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO or
ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP) and
not (ALARME_MCP_ESPERA_ENTRE_PARTIDAS) and
not (PRDA_EMERG_MCP) and not (MCP_EM_OPERACAO_LOCAL) and
ER_DESACOPLADA and BOMBA_AGUA_SALGADA_LIGADA and
VAIVULA_BOMBA_AGUA_SALGADA_ABERTA and
MANETE_EXTERNA_HABILITADA and STARTER_MCP_EM_OFF and

```

```

CHAVE_BATERIA_MCP_LIGADA and
TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45 and
TEMP_H2O_RESF_MCP_MENOR_IGUAL_85 and
DEMANDA_MCP_IGUAL_600 and ROTACAO_MCP_IGUAL_0);
wait[0,0];
to PROBE
from PROBE
wait[0,0];
to UPDATE_OUTPUTS1
...
from WAITING
to UPDATE_INPUTS0

```

Código 19 - Tradução para FIACRE do BLD da Figura 19.

O Código 20 apresenta o trecho da tradução para FIACRE do segundo BLD (procedimento para partida normal do MCP). O estado *probe* é posicionado para verificar o efeito COMANDO_START_MCP. São necessários dois estados, um para envio de mensagem para o processo do temporizador DI0, e um estado para receber mensagem do processo do temporizador DI0 com o valor da saída atualizado.

```

...
from UPDATE_OUTPUTS1
portDI0_IN! not(ALARME_MCP_ESPERA_ENTRE_PARTIDAS) and
ROTACAO_MCP_IGUAL_0 and ER_DESACOPLADA and
INTERLOCK_PARTIDA_MCP and START_MCP_SUPERVISORIO and
STARTER_MCP_EM_OFF;
to UPDATE_OUTPUTS2
from UPDATE_OUTPUTS2
portDI0_Q? COMANDO_START_MCP;
to PROBE
from PROBE
wait[0,0];
to UPDATE_OUTPUTS3
...
from WAITING
to UPDATE_INPUTS0

```

Código 20- Tradução para FIACRE da primeira saída do BLD da Figura 20.

O Código 21 apresenta o trecho da tradução para FIACRE também do segundo BLD (procedimento para partida normal do MCP), Figura 20. O estado *probe* é posicionado para verificar o efeito ALARME_MCP_ESPERA_ENTRE_PARTIDAS. São necessários dois estados, um para envio de mensagem para o processo do temporizador DI1, e um estado para receber mensagem do processo do temporizador DI1 e calcular o valor da saída.

```

...
from UPDATE_OUTPUTS3
  portDI1_IN! COMANDO_START_MCP;
  to UPDATE_OUTPUTS4
from UPDATE_OUTPUTS4
  portDI1_Q? DI1_Q;
  ALARME_MCP ESPERA_ENTRE_PARTIDAS:=          DI1_Q          and
  ROTACAO_MCP_MENOR_590;
  to PROBE
from PROBE
  wait[0,0];
  to UPDATE_OUTPUTS5
...
from WAITING
  to UPDATE_INPUTS0

```

Código 21- Tradução para FIACRE da segunda saída do BLD da Figura 20.

O Código 22 apresenta o trecho da tradução para FIACRE do terceiro BLD (Intertravamento para parada normal do MCP), Figura 21. O estado *probe* é posicionado para verificar o efeito INTERLOCK_PARADA_NORMAL_MCP. São necessários dois estados, um para envio de mensagem para o processo do temporizador DI2, e um estado para receber mensagem do processo do temporizador DI2 e calcular o valor da saída.

```

...
from UPDATE_OUTPUTS5
  portDI2_IN!          ROTACAO_MCP_MAIOR_IGUAL_600          and
  ROTACAO_MCP_MENOR_700;
to UPDATE_OUTPUTS6
  from UPDATE_OUTPUTS6
  portDI2_Q? DI2_Q;
  INTERLOCK_PARADA_NORMAL_MCP:=          DI2_Q          and
  not(MCP_EM_OPERACAO_LOCAL) and DEMANDA_MCP_MAIOR_IGUAL_600 and
  DEMANDA_MCP_MENOR_700;
  to PROBE
from PROBE
  wait[0,0];
  to UPDATE_OUTPUTS7
...
from WAITING
  to UPDATE_INPUTS0

```

Código 22 - Tradução para FIACRE do BLD da Figura 21.

O Código 23 apresenta um trecho da tradução para FIACRE do quarto BLD (procedimento para Parada em Emergência do MCP),

Figura 22. O estado *probe* é posicionado para verificar o efeito DEMANDA_FORCADA_MCP.

```

...
from UPDATE_OUTPUTS7
  DEMANDA_FORCADA_MCP:=((DEMANDA_FORCADA_MCP           or
(DEMANDA_MCP_MAIOR_600   and   not(ER_DESACOPLADA)   and
PRDA_EMERG_MCP)) and not(MCP_OFF));
  wait[0,0];
  to PROBE
from PROBE
  wait[0,0];
  to UPDATE_OUTPUTS8
...
from WAITING
  to UPDATE_INPUTS0

```

Código 23 - Tradução para FIACRE para a primeira saída do BLD da Figura 22.

O Código 24 apresenta o trecho da tradução para FIACRE do quarto BLD (procedimento para Parada em Emergência do MCP), Figura 22, porém o estado *probe* é posicionado para verificar o efeito MCP_STOPPING. São necessários dois estados, um para envio de mensagem para o processo do temporizador DI3, e um estado para receber mensagem do processo do temporizador DI3 e calcular o valor da saída.

```

...
from UPDATE_OUTPUTS8
  portDI3_IN! DEMANDA_FORCADA_MCP;
  to UPDATE_OUTPUTS9
from UPDATE_OUTPUTS9
  portDI3_Q? DI3_Q;
  MCP_STOPPING:= (MCP_STOPPING or (DI3_Q or (ER_DESACOPLADA
and PRDA_EMERG_MCP)) and not(MCP_OFF));
  to PROBE
from PROBE
  wait[0,0];
  to UPDATE_OUTPUTS10
...
from WAITING
  to UPDATE_INPUTS0

```

Código 24 - Tradução para FIACRE para a segunda saída do BLD da Figura 22.

O Código 25 apresenta o trecho da tradução para FIACRE do quarto BLD (procedimento para Parada em Emergência do MCP, Figura

22, porém o estado *probe* é posicionado para verificar o efeito MCP_OFF.

```

...
from UPDATE_OUTPUTS10
  MCP_OFF:= MCP_STOPPING and ROTACAO_MCP_IGUAL_0;
  wait[0,0];
  to PROBE
from PROBE
  wait[0,0];
  to UPDATE_OUTPUTS11
...
from WAITING
  to UPDATE_INPUTS0

```

Código 25 - Tradução para FIACRE para a terceira saída do BLD da Figura 22.

O Código 26 apresenta o trecho da tradução para FIACRE do quinto BLD (intertravamento para acoplar a ER), Figura 23. O estado *probe* é posicionado para verificar o efeito INTERLOCK_ACOPLAR_AV_AR. São necessários quatro estados para envio e recebimento de mensagens dos processos do temporizadores DI4, DI5 e DI6, e um estado para receber mensagem do processo do temporizador DI6 e para atualizar o valor da saída.

```

..
from UPDATE_OUTPUTS11
  portDI4_IN! ER_DESACOPLADA;
  to UPDATE_OUTPUTS12

from UPDATE_OUTPUTS12
  portDI5_IN! ER_ACOPLADA_AV;
  to UPDATE_OUTPUTS13

from UPDATE_OUTPUTS13
  portDI6_IN! ER_ACOPLADA_AR;
  to UPDATE_OUTPUTS14

from UPDATE_OUTPUTS14
  portDI4_Q? DI4_Q;
  to UPDATE_OUTPUTS15

from UPDATE_OUTPUTS15
  portDI5_Q? DI5_Q;
  to UPDATE_OUTPUTS16

from UPDATE_OUTPUTS16
  portDI6_Q? DI6_Q;
  INTERLOCK_ACOPLAR_AV_AR:=(DI4_Q or DI5_Q or DI6_Q) and

```

```

not (MCP_EM_OPERACAO_LOCAL) or
ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER or
ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER or
ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER) and
not (PRDA_EMERG_MCP) and ROTACAO_MCP_MAIOR_IGUAL_600 and
ROTACAO_MCP_MENOR_IGUAL_700 and DEMANDA_MCP_MAIOR_IGUAL_600
and DEMANDA_MCP_MENOR_IGUAL_700;
  to PROBE
  wait[0,0];
  to UPDATE_OUTPUTS17
  ...
from WAITING
to UPDATE_INPUTS0

```

Código 26- Tradução para FIACRE do BLD da Figura 23.

O Código 27 apresenta o trecho da tradução para FIACRE do sexto BLD ((intertravamento para desacoplar a ER), Figura 24. O estado *probe* é posicionado para verificar o efeito INTERLOCK_DESACOPLAR. São necessários quatro estados, para envio e recebimento de mensagens dos processos dos temporizadores DI7 e DI8, e um estado para receber mensagem do processo do temporizador DI8 e atualizar o valor da saída.

```

...
from UPDATE_OUTPUTS17
  portDI7_IN! ER_ACOPLADA_AV;
  to UPDATE_OUTPUTS18

from UPDATE_OUTPUTS18
  portDI8_IN! ER_ACOPLADA_AR;
  to UPDATE_OUTPUTS19
from UPDATE_OUTPUTS19
  portDI7_Q? DI7_Q;
  to UPDATE_OUTPUTS20
from UPDATE_OUTPUTS20
  portDI8_Q? DI8_Q;
  INTERLOCK_DESACOPLAR:=(DI7_Q or DI8_Q) and
ROTACAO_MCP_MAIOR_IGUAL_600 and ROTACAO_MCP_MENOR_IGUAL_700
and DEMANDA_MCP_MAIOR_IGUAL_600 and
DEMANDA_MCP_MENOR_IGUAL_700;
  to PROBE
from PROBE
  wait[0,0];
  to WAITING
from WAITING
to UPDATE_INPUTS0

```

Código 27- Tradução para FIACRE do BLD da Figura 24.

O Código 28 apresenta a estrutura FIACRE do *component* que é o componente responsável pela composição paralela das instâncias dos processos, descrevendo a interação entre as instâncias. O código não está contemplando o uso das estratégias de redução do espaços de estados.

```

component BLD is
  port portDI0_IN: in out bool in [0,0],
    portDI0_Q: in out bool in [0,0],
    portDI0_Timer: sync in [2,2],
    portDI1_IN: in out bool in [0,0],
    portDI1_Q: in out bool in [0,0],
    portDI1_Timer: sync in [15,15],
    portDI2_IN: in out bool in [0,0],
    portDI2_Q: in out bool in [0,0],
    portDI2_Timer: sync in [240,240],
    portDI3_IN: in out bool in [0,0],
    portDI3_Q: in out bool in [0,0],
    portDI3_Timer: sync in [3,3],
    portDI4_IN: in out bool in [0,0],
    portDI4_Q: in out bool in [0,0],
    portDI4_Timer: sync in [2,2],
    portDI5_IN: in out bool in [0,0],
    portDI5_Q: in out bool in [0,0],
    portDI5_Timer: sync in [2,2],
    portDI6_IN: in out bool in [0,0],
    portDI6_Q: in out bool in [0,0],
    portDI6_Timer: sync in [2,2],
    portDI7_IN: in out bool in [0,0],
    portDI7_Q: in out bool in [0,0],
    portDI7_Timer: sync in [2,2],
    portDI8_IN: in out bool in [0,0],
    portDI8_Q: in out bool in [0,0],
    portDI8_Timer: sync in [2,2]

  priority portDI0_Timer > portDI0_Q,
    portDI1_Timer > portDI1_Q,
    portDI2_Timer > portDI2_Q,
    portDI3_Timer > portDI3_Q,
    portDI4_Timer > portDI4_Q,
    portDI5_Timer > portDI5_Q,
    portDI6_Timer > portDI6_Q,
    portDI7_Timer > portDI7_Q,
    portDI8_Timer > portDI8_Q

  par * in
    diagram
  [portDI0_IN,portDI0_Q,portDI1_IN,portDI1_Q,portDI2_IN,
portDI2_Q,portDI3_IN,portDI3_Q,portDI4_IN,portDI4_Q,portDI5_IN,
portDI5_Q,portDI6_IN,portDI6_Q,portDI7_IN,portDI7_Q,portDI8_IN,
portDI8_Q]
  || DI [portDI0_IN,portDI0_Q,portDI0_Timer]

```

```

|| DI [portDI1_IN,portDI1_Q,portDI1_Timer]
|| DI [portDI2_IN,portDI2_Q,portDI2_Timer]
|| DI [portDI3_IN,portDI3_Q,portDI3_Timer]
|| DI [portDI4_IN,portDI4_Q,portDI4_Timer]
|| DI [portDI5_IN,portDI5_Q,portDI5_Timer]
|| DI [portDI6_IN,portDI6_Q,portDI6_Timer]
|| DI [portDI7_IN,portDI7_Q,portDI7_Timer]
|| DI [portDI8_IN,portDI8_Q,portDI8_Timer]
end

```

BLD

Código 28 - Trecho da composição paralela de instâncias dos processos.

Além disso, como apresentado na subseção 3.3, para cada temporizador existente na MCE deverá existir um contador temporal equivalente.

Com a tradução dos BLDs e da MCE do exemplo da MB para FIACRE, pode-se agora realizar o *model checking* com as estratégias de redução para minimizar o problema de explosão combinacional.

4.5 REALIZANDO *MODEL CHECKING*

As técnicas de redução se fazem necessárias devido ao grande volume de informações usualmente presentes nas MCEs. Sendo assim, a Figura 25 apresenta o cone de influência da MCE (Tabela 15). Exemplificando, para o efeito INTERLOCK_PARTIDA_MCP, pode-se identificar que, entre outras, as variáveis ER_ACOPLADA_AV e ER_ACOPLADA_AR não geram influência no efeito, portanto, tais variáveis são desprezadas no código.

A estratégia utilizada para desprezar as variáveis que não geram influência no efeito que se está verificando ocorre através de comentários no código, pelos símbolos // (exclui da compilação o que estiver na linha após as barras) e /* */ (que exclui da compilação todo código entre os símbolos de asterisco).

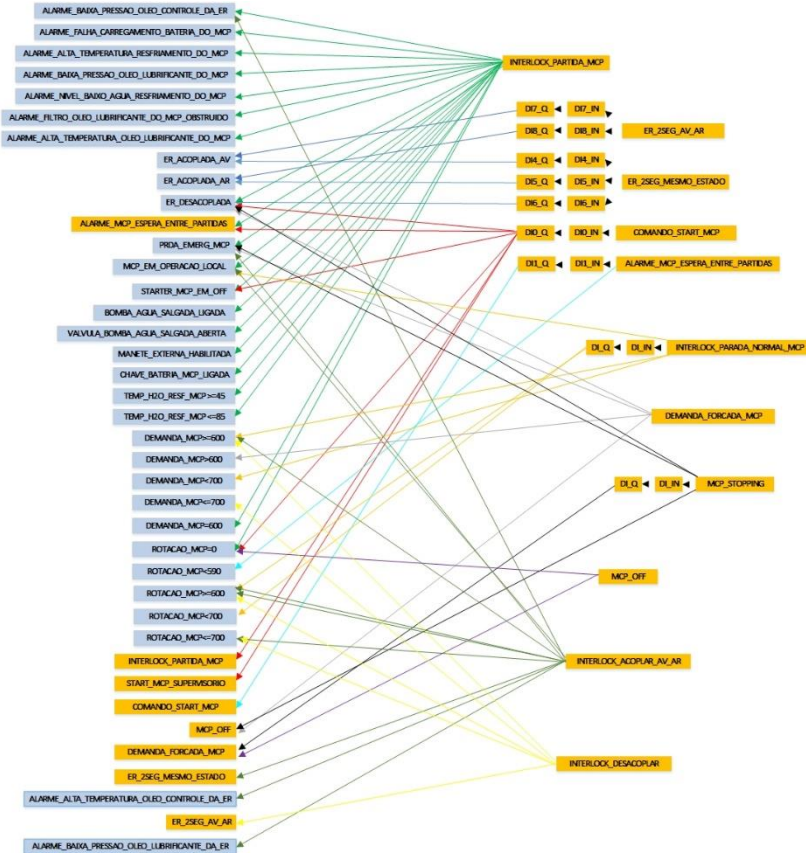


Figura 25 - Cone de influência da aplicação da MB.

Realizando *model checking* somente com a estratégia de redução por cone de influência, obtém-se a Tabela 16, onde verifica-se que não são todos os casos de aplicação que conseguem ser solucionados somente com a estratégia de redução por cone de influência, devido à ocorrência de explosão combinacional. Sendo assim, para verificação dos efeitos “INTERLOCK_PARTIDA_MCP”, “COMANDO_START_MCP”, “ALARME_MCP ESPERA ENTRE PARTIDAS” e “INTERLOCK_ACOPLAR_AV_AR” é necessário o uso de outra estratégia de redução para evitar explosão combinacional.

Tabela 16 - Resultado do *model checking* com a redução por cone de influência.

Efeitos	Estados	Tempo de exec.	FS/FP/OK
INTERLOCK_PARTIDA_MCP	∞	∞	-/-
COMANDO_START_MCP	∞	∞	-/-
ALARME_ESPERA_ENTRE PARTIDAS	∞	∞	-/-
INTERLOCK_PARADA_NORMAL_MCP	2507	0.008s	0/0/2
DEMANDA_FORCADA_MCP	6814	0.016s	0/0/2
MCP_STOPPING	4670	0.008s	0/0/2
MCP_OFF	4530	0.012s	0/0/2
INTERLOCK_ACOPLAR_AV_AR	∞	∞	-/-
INTERLOCK_DESACOPLAR	13393	0.024s	0/0/2
Legenda: FS = Falha Segura FP = Falha Perigosa OK = Propriedades aprovadas			

Apesar de não ser uma técnica de redução, e sim de implementação, uma alternativa para contornar o problema na verificação do efeito “INTERLOCK_ACOPLAR_AV_AR” é a adição de estados intermediário no estado de atualização das entradas. O estado concentrava um número grande de variáveis, gerando uma grande árvore de possibilidades, não sendo possível armazenar na memória do computador utilizado. Após a correção no código implementado, o *model checking* foi realizado e o código atualizado está apresentado no Código 29, com a adição dos estados A e B.

```

from UPDATE_INPUTSO
  ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER:=any;
  //ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP:=any;
  //ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP:=any;
  //ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP:=any;
  //ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO:=any;
  //ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP:=any;
  //STARTER_MCP_EM_OFF:=any;
  ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER:=any;
  ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER:=any;
  wait[0,0];
  to A
from A
  PRDA_EMERG_MCP:=any;
  MCP_EM_OPERACAO_LOCAL:=any;
  //BOMBA_AGUA_SALGADA_LIGADA:=any;
  //VALVULA_BOMBA_AGUA_SALGADA_ABERTA:=any;
  //MANETE_EXTERNA_HABILITADA:=any;
  //CHAVE_BATERIA_MCP_LIGADA:=any;
  //START_MCP_SUPERVISORIO:=any;

```

```

//DEMANDA_FORCADA_MCP:=any;
//it:=any;
id:=any;
wait[0,0];
to B
from B
ir:=any;
//STARTER_MCP_EM_OFF:=any;
ER:=any;
wait[0,0];
to UPDATE_INPUTS1

```

Código 29 - Estratégia para o *model checking* do efeito “INTERLOCK_ACOPLAR_AV_AR”.

Para o efeito “INTERLOCK_PARTIDA_MCP” a estratégia de redução por decomposição juntamente com a redução por cone de influência, foi suficiente para contornar o problema de explosão combinacional. A Figura 26 apresenta as variáveis que estão relacionadas com o efeito e a decomposição das propriedades em dois blocos.

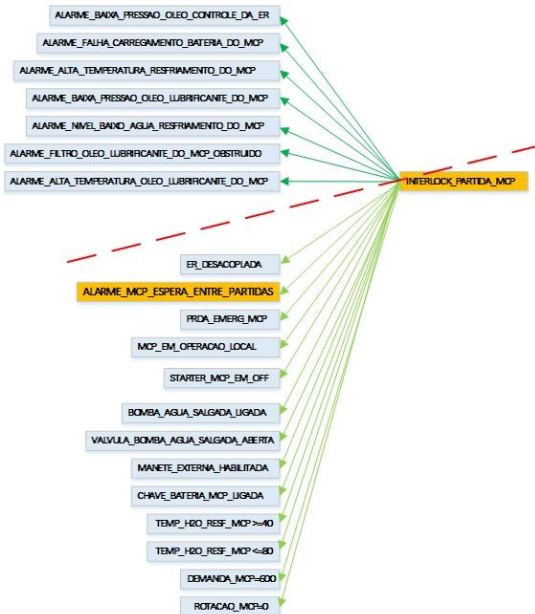


Figura 26 - Ilustração da redução por decomposição para o efeito
“INTERLOCK_PARTIDA_MCP”.

A estratégia de redução por decomposição para o efeito “INTERLOCK_PARTIDA_MCP” é aplicada dividindo as propriedades verificadas em blocos. No presente caso, em dois blocos:

$$\text{LFS} \equiv \left\{ \begin{array}{c} \boxed{\neg(\text{Efeito} \vee (A_1 \wedge A_2 \wedge \dots \wedge A_7))} \\ \wedge \\ \boxed{\neg(\text{Efeito} \vee (A_8 \wedge A_9 \wedge \dots \wedge A_{22}))} \end{array} \right\}$$

$$\text{LFP} \equiv \left\{ \begin{array}{c} \boxed{(\text{Efeito} \vee \neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_7)} \\ \wedge \\ \boxed{(\text{Efeito} \vee \neg A_8 \vee \neg A_9 \vee \dots \vee \neg A_{22})} \end{array} \right\}$$

sendo que:

$A_1 = \neg\text{ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER}$
 $A_2 = \neg\text{ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP}$
 $A_3 = \neg\text{ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP}$
 $A_4 = \neg\text{ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP}$
 $A_5 = \neg\text{ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP}$
 $A_6 = \neg\text{ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO}$
 $A_7 = \neg\text{ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP}$

e

$A_8 = \text{ER_DESACOPLADA}$
 $A_9 = \neg\text{ALARME_MCP_ESPERA_ENTRE_PARTIDAS} =$
 $= \neg(\text{ROTACAO_MCP_MENOR_590 and (COMANDO_START_MCP and (OBS_T3)))$
 $A_{10} = \neg\text{PRDA_EMERG_MCP}$
 $A_{11} = \neg\text{MCP_EM_OPERACAO_LOCAL}$
 $A_{12} = \text{STARTER_MCP_EM_OFF}$
 $A_{13} = \text{BOMBA_AGUA_SALGADA_LIGADA}$
 $A_{14} = \text{VALVULA_BOMBA_AGUA_SALGADA_ABERTA}$
 $A_{15} = \text{MANETE_EXTERNA_HABILITADA}$
 $A_{16} = \text{CHAVE_BATERIA_MCP_LIGADA}$
 $A_{17} = \text{TEMP_H2O_RESF_MCP_MAIOR_IGUAL_40}$
 $A_{18} = \text{TEMP_H2O_RESF_MCP_MENOR_IGUAL_80}$
 $A_{19} = \text{DEMANDA_MCP_IGUAL_600}$
 $A_{20} = \text{ROTACAO_MCP_IGUAL_0}$

Utilizando as mesmas estratégias de redução para os efeitos “COMANDO_START_MCP” e “ALARME_MCP_ESPERA_ENTRE_PARTIDAS”, é possível completar as informações da Tabela 16, gerando a Tabela 17, onde pode ser verificada a ausência de FS e FP. Cabe ressaltar que para os efeitos da MCE que utilizaram a redução por decomposição (efeitos que na Tabela 17 foram decompostos em índices), apesar do *model checking* ter resultado em LFS e LFP, somente a verificação de livre de falha segura é exaustivamente verificada, conforme apresentado na subseção 3.4.2.

Tabela 17 - Resultado do *model checking* com a redução por decomposição e por cone de influência.

Efeitos	Estados	Tempo de exec.	FS/FP/OK
INTERLOCK_PARTIDA_MCP_1	121850	0.208s	0/0/2
INTERLOCK_PARTIDA_MCP_2	134208	0.460s	0/0/2
COMANDO_START_MCP_1	11727	0.165s	0/0/2
COMANDO_START_MCP_2	250128	0.264s	0/0/2
COMANDO_START_MCP_3	58368	0.116s	0/0/2
ALARME_ESPERA_ENTRE PARTIDAS_1	8637	0.016s	0/0/2
ALARME_ESPERA_ENTRE PARTIDAS_2	189618	0.272s	0/0/2
ALARME_ESPERA_ENTRE PARTIDAS_3	58368	0.116s	0/0/2
INTERLOCK_PARADA_NORMAL_MCP	2507	0.008s	0/0/2
DEMANDA_FORCADA_MCP	6814	0.016s	0/0/2
MCP_STOPPING	4670	0.008s	0/0/2
MCP_OFF	4530	0.012s	0/0/2
INTERLOCK_ACOPLAR_AV_AR	509826	0.660s	0/0/2
INTERLOCK_DESACOPLAR	13393	0.024s	0/0/2
Legenda: FS = Falha Segura FP = Falha Perigosa OK = Propriedades aprovadas			

Como os BLDs da MB estavam LFS e LFP, portanto de acordo com o especificado na MCE, foi introduzida uma falha no BLD da Figura 24, para apresentação do contraexemplo. A Figura 27 apresenta o contraexemplo na forma de diagrama de sinais, para facilitar a interpretação do erro.

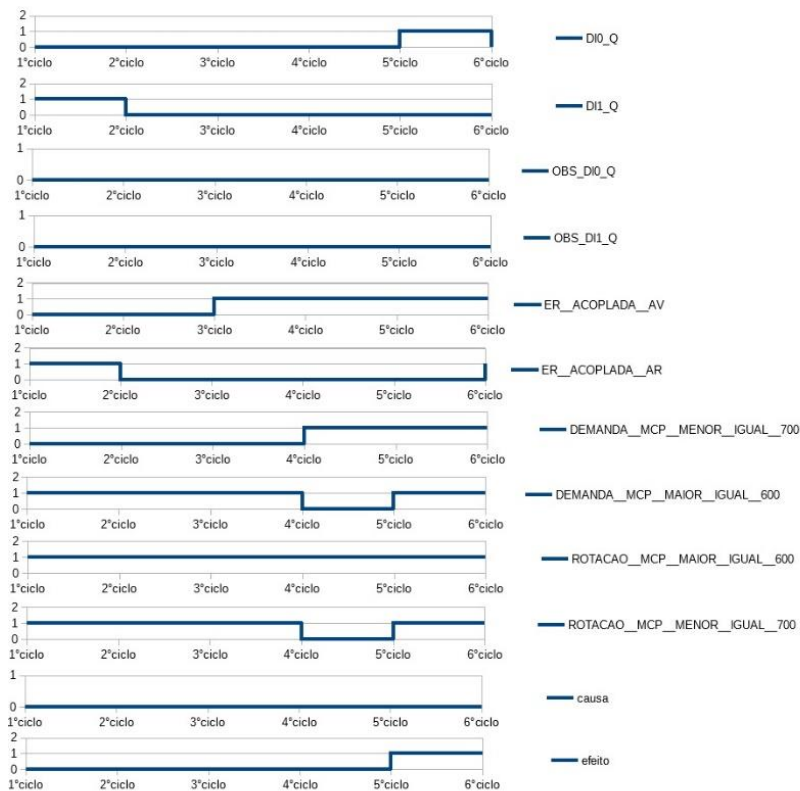


Figura 27 - Contraexemplo do erro forçado, para o efeito “INTERLOCK_DESACOPLAR”.

O diagrama de sinais (Figura 27) apresenta que no quinto ciclo há uma falha segura. Analisando os temporizadores do BLD (DIO_Q e DII_1) e o contador temporal (OBS_DIO_Q e OBS_DII_Q), identifica-se a falta de sincronismo entre os temporizadores do BLD e do especificado, gerado pela incoerência do valor especificado no BLD com o desejado na MCE. Corrigindo o tempo dos temporizadores do BLD de acordo com o especificado em sua MCE e realizando novamente o *model checking*, obtém-se um sistema LFS e LFP.

4.6 CONCLUSÃO DO CAPÍTULO

Com a documentação fornecida pela MB, uma aplicação real pôde ser utilizada para testar a aplicabilidade da metodologia proposta em um estudo de caso real e complexo. Os documentos fornecidos foram interpretados e a MCE foi gerada por ainda não ser um documento utilizado pela MB. Os modelos de tradução de BLDs e de MCEs para FIACRE foram utilizados, confirmando as vantagens de se utilizar uma linguagem intermediária formal.

As estratégias de redução foram aplicadas e as ferramentas de verificação foram utilizadas, obtendo a resposta de que os BLDs não continham FS e FP. Foi realizada uma simulação de falha na elaboração do BLDs para testar a identificação do erro pelo método. O *model checking* foi realizado e o erro foi identificado com auxílio do contraexemplo como diagrama de sinais. A apresentação do contraexemplo desta forma demonstrou a agilidade e eficácia na identificação de erros.

5 CONCLUSÃO

Esta dissertação contribuiu propondo a especificação tanto do código para o CLP quanto das propriedades de segurança em documentos normatizados e padronizados que são os BLDs e as MCEs, facilitando a compreensão das especificações e ressaltando sua importância por todas as equipes envolvidas no projeto de SCM da MB. A experiência de sucesso da Petrobras com o uso desses documentos de especificação em projetos de SIS corroboram com a utilização na MB.

Os modelos propostos de tradução dos BLDs e das MCEs para FIACRE viabilizaram a realização de verificação formal por *model checking*, atendendo à IEC 61508 (2010), aumentando a confiabilidade do sistema devido à busca exaustiva por erros e reduzindo os custos em correção de erros, pois a verificação ocorre já na etapa de projeto conceitual, etapa em que os custos de correção são relativamente baixos. Devido aos requisitos de segurança dos projetos de SCM, é muito importante se certificar que o sistema é mantido em condições de operação seguras. A utilização da metodologia garante que a especificação do código para o CLP respeita todas as especificações de segurança.

Este trabalho também contribuiu com a apresentação do contraexemplo em uma forma mais amigável aos projetistas do que o contraexemplo fornecido pela ferramenta verificadora SELT. A apresentação na forma de diagrama de sinais facilita e agiliza a identificação de erros, pois é apresentado o estado das variáveis (ativo ou inativo) a cada ciclo de leitura do BLD.

A estratégia de redução por decomposição proposta, possibilitou a redução do espaço de estados gerados, reduzindo o problema de explosão combinacional que é uma das limitações do método de verificação por *model checking*. Uma limitação da metodologia proposta pode ser ressaltada, que é a possibilidade de não realizar uma verificação exaustiva das propriedades LFP e LFS simultaneamente, nos casos em que a causa é formada somente por lógicas “or” ou lógicas “and”. Para esses casos pode-se realizar a verificação por *model checking*, porém, deve-se ter ciência de que grande parte das possibilidades são checadas mas não todas.

Para utilizarem a metodologia de desenvolvimento de SCM da MB assistido por *model checking*, os engenheiros e técnicos terão apenas que especificar o código do CLP e das propriedades de segurança na forma de BLDs e de MCE, respectivamente, que utilizam padrões lógicos conhecidos.

Apesar deste trabalho ter sido desenvolvido para aplicação na MB, ele pode ser aplicado facilmente em outros domínios de aplicação, necessitando apenas de adaptações às suas peculiaridades. Os modelos de tradução dos BLDs e da MCE podem ser utilizados de forma bem semelhante à proposta.

A aplicação real fornecida pela MB possibilitou avaliar a metodologia proposta em uma aplicação semelhante ao que usualmente se desejará verificar. Mesmo com um bom volume de BLDs (6 BLDs) e da reduzida dimensão da MCE (39x11), fruto da simbologia proposta, a verificação foi possível ser realizada, contornando problemas de explosão combinacional. A resposta de que a especificação era LFS e LFP, e também a identificação do erro forçado, contribui para concluir a eficácia da metodologia.

Levando-se em conta todos os resultados satisfatórios obtidos, inclusive satisfazendo todos os objetivos deste trabalho, a utilização da nova metodologia proposta na MB gerará significativas melhorias no desenvolvimentos de projetos de SCM, justificando a sua utilização em substituição à atualmente utilizada.

5.1 TRABALHOS FUTUROS

Apesar dos resultados satisfatórios obtidos e apresentados neste trabalho, alguns aprimoramentos são possíveis e desejáveis, e poderão ser fruto de trabalhos futuros, como:

- Novas estratégias de redução do problema de explosão combinacional, já que a estratégia de redução por decomposição proposta não garante a exaustão da verificação em casos específicos;
- Outra metodologia de verificação pode ser realizada comparando a especificação do código para o CLP com o código que efetivamente foi programado, buscando garantir que o que foi programado realmente foi o que se especificou no BLD. Este novo trabalho seria um complemento a este e a um trabalho em andamento, no grupo de trabalho de sistemas a eventos discretos do PGEAS, que realiza a verificação formal de códigos de CLP em LD com propriedades de segurança;
- Criação de ferramenta de tradução automática, com regras de tradução baseadas nos modelos propostos nesta dissertação,

para editores comerciais de BLDs, agilizando o processo de verificação e evitando erros de modelagem; e

- Criação de uma ferramenta computacional para preenchimento da MCE com o padrão proposto para a MB permitindo a tradução automática da MCE para FIACRE, agilizando o processo de verificação. Apesar de existir uma ferramenta para criação de MCE desenvolvida pelo grupo de trabalho, ela utiliza o padrão da Petrobras e não o padrão proposto para a MB.

REFERÊNCIAS

- ADIEGO, B. F., DARVAS, D., VIÑUELA, E. B., TOURNIER, J. C., BLIUDZE, S., BLECH, J. O., SUÁREZ, V. M. G. Applying model checking to industrial-sized PLC programs. **IEEE Transactions on Industrial Informatics**, v. 11, n. 6, p. 1400-1410, 2015.
- BARBOSA, L. P., GORGÔNIO, K., LIMA, A. M. N., PERKUSICH, A., DA SILVA, L. D. On the automatic generation of timed automata models from isa 5.2 diagrams. In: **Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on**. [S.l.], 2007. p. 406-412.
- BERHMANN, G.; DAVID, A.; LARSEN, K. G. A tutorial on uppaal, formal methods for the design of real-time systems. **SFM-RT, Springer-Verlag**, v. 965, p. 200-236, 2004.
- BERTHOMIEU, B., BODEVEIX, J. P., FARAIL, P., FILALI, M., GARAVEL, H., GAUFILLET, P., VERNADAT, F. Fiacre: an intermediate language for model verification in the topcased environment. In **ERTS 2008**. [S.l.], 2008.
- BERTHOMIEU, B., BODEVEIX, J. P., FILALI, M., GARAVEL, H., LANG, F., PERES, F., LANG, F. The Syntax and Semantics of FIACRE. **Report LAAS**, n. 07264, 2007.
- BERTHOMIEU, B.; VERNADAT, F. Time petri nets analysis with tina. In: IEEE. **Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on**. [S.l.], 2006. pp. 123-124.
- CANET, G., COUFFIN, S., LESAGE, J. J., PETIT, A., SCHNOEBELEN, P. Towards the automatic verification of PLC programs written in Instruction List. In: IEEE. **Systems, Man, and Cybernetics, 2000 IEEE International Conference on**. [S.l.], 2000. V4, p. 2449-2454.
- CAVADA, R., CIMATTI, A., DORIGATTI, M., GRIGGIO, A., MARIOTTI, A., MICHELI, A., TONETTA, S. The nuXmv symbolic model checker. In: SPRINGER. **International Conference on Computer Aided Verification**. [S.l.], 2014. p. 334-342.

CLARKE, E., BIÈRE, A., RAIMI, R., ZHU, Y. Bounded model checking using satisfiability solving. **Formal methods in system design**, Springer, v. 19, n. 1, p. 7-34, 2001.

CLARKE, E. M.; EMERSON, E. A. Synthesis of synchronization skeletons for branching time temporal logic. In: **Logic of programs: Workshop**. [S.l.: s.n.], 1981, v. 131, p. 244-263.

CLARKE, E. M.; EMERSON, E. A.; SISTLA, A. P. Automatic verification of finite-state concurrent systems using temporal logic specifications. **ACM Transactions on Programming Languages and Systems (TOPLAS)**, ACM, v. 8, n. 2, p. 244-263, 1986.

CLARKE, E. M., GRUMBERG, O.; PELED, D. **Model checking**. MIT press, 1999.

DA SILVA, L. D., BARBOSA, L. P., GORGÔNIO, K., PERKUSICH, A., LIMA, A. M. N. On the automatic generation of timed automata models from function block diagrams for safety instrumented systems. In: IEEE. **Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE**. [S.l.], 2008. p. 291-296.

DARVAS, D., ADIEGO, B. F., VÖRÖS, A., BARTHA, T., VIÑUELA, E. B., SUÁREZ, V. M. G. Formal verification of complex properties on plc programs. In: SPRINGER. **International Conference on Formal Techniques for Distributed Objects, Components, and Systems**. [S.l.], 2014. p. 284-299.

FARINES, J. M., DE QUEIROZ, M. H., DA ROCHA, V. G., CARPES, A. M. M., VERNADAT, F., CRÉGUT, X. A model-driven engineering approach to formal verification of PLC programs. In **Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on**. [S.l.], 2011, p. 1-8.

FREY, G.; LITZ, L. Formal methods in PLC programming. **Systems, Man, and Cybernetics, 2000 IEEE International Conference on**. [S.l.], 2000. v. 4, p. 2431-2436.

GERGELY, E. I.; COROIU, L.; POPENTIU-VLADICESCU, F. Methods for Validation of PLC Systems. **Journal of Computer Science and Control Systems**. University of Oradea, v. 4, n. 1, p. 47, 2011.

HAFER, T.; THOMAS, W. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. **Automata, Languages and Programming**. Springer, p. 269-279, 1987.

IEC 61508. Functional safety of electrical/electronic/programmable electronic safety related systems. **International Electrotechnical Commission**, 2010.

IEC 61131-1. IEC International Standard 61131-1: Programmable controllers - Part 1: General information. **International Electrotechnical Commission**, 2003.

IEC 61131-3. IEC International Standard 61131-3: Programmable Controllers Part 3: Programming Languages. **International Electrotechnical Commission**, 2013.

ISA 5.2. Binary Logic Diagrams for Process Operations. **International Society of Automation**. (RI992) edition, 1992.

LARSEN, K. G., MIKUCIONIS, M., NIELSEN, B., SKOU, A. Testing real-time embedded software using UPPAAL-TRON: an industrial case study. **In Proceedings of the 5th ACM international conference on Embedded software**. ACM, p. 299-306, 2005.

MCMILLAN, K. L. Symbolic model checking. In: **Symbolic Model Checking**. [S.l.], Springer, 1993, p. 25-60.

MOON, I. Modeling programmable logic controllers for logic verification. **IEEE Control Systems**, IEEE, v. 14, n. 2, p. 53-59, 1994.

OLIVEIRA, C. **Verificação de modelos aplicada ao projeto de sistemas industriais automatizados por controladores lógicos programáveis**. Dissertação de Mestrado. Instituto Militar de Engenharia, Rio de Janeiro, RJ, 2006.

OLIVEIRA, K. **Geração Automática de Testes de Conformidade para Programas de Controladores Lógicos Programáveis** - Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande. Centro de Engenharia Elétrica e Informática - Campina Grande, 2009.

PETROBRAS. Especificação Técnica para Implementação da lógica de intertravamento e controle - ET-3000.00-5500-800-PCI-002REVB, 2005.

PNUELI, A. The temporal logic of programs. In: IEEE. **Foundations of Computer Science, 18th Annual Symposium on.** [S.l.], 1977. p. 46-57.

SILVA, A. de M. **Aplicação de verificação de modelos a programas de clp: Explorando a temporização.** Dissertação (Mestrado) - Curso de Mestrado em Engenharia Elétrica do Instituto Militar de Engenharia, Rio de Janeiro, RJ, 2008.

SOUZA, M. F. d. **Modelagem e verificação de programas de CLP escritos em diagrama ladder.** Dissertação - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. Florianópolis, SC, 2010.

YOUNIS, M. B.; FREY, G. Formalization of existing PLC programs: A survey. In: **Proceedings of CESA.** [S.l.], 2003. p. 234-239.

**APÊNDICE A - Código FIACRE para *model checking* do BLD da
Figura 7 para o primeiro efeito da MCE da Tabela 10**

```

const DEMANDA_MCP: array 3 of nat is [599,600,601] //VAR id

process diagram
[portDI0_IN: out bool, portDI0_Q: in bool]
is
states UPDATE_INPUTS0, UPDATE_OUTPUTS0, UPDATE_OUTPUTS1,
UPDATE_OUTPUTS2, PROBE, WAITING

var id:0..2:=0,
ROTACAO_MCP_IGUAL_0: bool := false,
PRDA_EMERG_MCP: bool := false,
DEMANDA_FORCADA_MCP: bool := false,
ER_DESACOPLADA: bool := false,
DEMANDA_MCP_MAIOR_600: bool := false,
MCP_OFF: bool := false,
MCP_STOPPING: bool := false,
DI0_Q: bool := false

init to UPDATE_OUTPUTS0
from UPDATE_INPUTS0
PRDA_EMERG_MCP:=any;
id:=any;
ER_DESACOPLADA:=any;
DEMANDA_MCP_MAIOR_600:=(DEMANDA_MCP[id]>600);
ROTACAO_MCP_IGUAL_0:=any;
wait[0,0];
to UPDATE_OUTPUTS0

from UPDATE_OUTPUTS0
DEMANDA_FORCADA_MCP:=((DEMANDA_FORCADA_MCP or
(DEMANDA_MCP_MAIOR_600 and not(ER_DESACOPLADA) and
PRDA_EMERG_MCP)) and not(MCP_OFF));
portDI0_IN! ((DEMANDA_FORCADA_MCP or
(DEMANDA_MCP_MAIOR_600 and not(ER_DESACOPLADA) and
PRDA_EMERG_MCP)) and not(MCP_OFF));
to PROBE

from PROBE
wait[0,0];
to UPDATE_OUTPUTS1

from UPDATE_OUTPUTS1
portDI0_Q? DI0_Q;
MCP_STOPPING:= ((MCP_STOPPING or (DI0_Q and
(ER_DESACOPLADA and PRDA_EMERG_MCP))) and not(MCP_OFF));
to UPDATE_OUTPUTS2

from UPDATE_OUTPUTS2
MCP_OFF:= ((MCP_STOPPING or (DI0_Q and (ER_DESACOPLADA
and PRDA_EMERG_MCP))) and not(MCP_OFF)) and

```

```
ROTACAO_MCP_IGUAL_0);
  wait[0,0];
  to WAITING

from WAITING
  to UPDATE_INPUTS0

////////////////////////////////// DI //////////////////////////////////
process DI
  [portIN: in bool, portQ: out bool, portTimer: sync]
  is
  states idle, running, elapsed
  var IN: bool := false,
      Q: bool := false
  init
    to idle
  from idle
    select
      portIN? IN;
      if IN then
        to running
      else
        loop
      end
    []
      portQ! Q;
      loop
    end
  from running
    select
      portIN? IN;
      if not IN then
        to idle
      else
        loop
      end
    []
      portTimer;
      Q := true;
      to elapsed
    []
      portQ! Q;
      loop
    end
  from elapsed
    select
      portIN? IN;
      if not IN then
        Q := false;
        to idle
      else
        loop
      end
    end
```

```

        []
        portQ! Q;
        loop
        end

//////////////////////////////////// BLD //////////////////////////////////////

component BLD
is
port portDI0_IN: in out bool in [0,0],
portDI0_Q: in out bool in [0,0],
portDI0_Timer: sync in [3,3]

priority portDI0_Timer > portDI0_Q

par * in
diagram [portDI0_IN,portDI0_Q]
|| DI [portDI0_IN,portDI0_Q,portDI0_Timer]
end

BLD

//////////////////////////////////// OBSERVER////////////////////////////////////

process timed_input [ITrue,IFalse,T: sync] is
states varFalse , varTrueUntimed , varTrueTimed
init to
varFalse
from varFalse
ITrue; to varTrueUntimed
from varTrueUntimed select
T; to varTrueTimed
[] IFalse; to varFalse
end
from varTrueTimed
IFalse; to varFalse

// Here is every Fcrtimer info in this list...

component obs_DI0 is
port ITrue: sync in [0,0] is (BLD/1/value
DEMANDA_FORCADA_MCP),
IFalse: sync in [0,0] is not(BLD/1/value
DEMANDA_FORCADA_MCP),
T: sync in [3,3]
priority IFalse > T
par
timed_input [ITrue,IFalse,T]
end

obs_DI0 > BLD

```

```
//////////VERIFICAÇÃO ////////////
    property probe is ltl (BLD/1/state TEST)
    property OBS_DIO is ltl (obs_DIO/1/state varTrueTimed)
    property PRDA_EMERG_MCP is ltl (BLD/1/value
PRDA_EMERG_MCP)
    property ER_DESACOPLADA is ltl (BLD/1/value
ER_DESACOPLADA)
    property ROTACAO_MCP_IGUAL_0 is ltl (BLD/1/value
ROTACAO_MCP_IGUAL_0)
    property DEMANDA_FORCADA_MCP is ltl (BLD/1/value
DEMANDA_FORCADA_MCP)
    property DEMANDA_MCP_MAIOR_600 is ltl (BLD/1/value
DEMANDA_MCP_MAIOR_600)
    property MCP_OFF is ltl (BLD/1/value MCP_OFF)
    property MCP_STOPPING is ltl (BLD/1/value MCP_STOPPING)

    property causa_1 is ltl ((DEMANDA_FORCADA_MCP or
(not(ER_DESACOPLADA) and PRDA_EMERG_MCP and
DEMANDA_MCP_MAIOR_600)) and not(MCP_OFF))
    property efeito_1 is ltl (DEMANDA_FORCADA_MCP)

    property Livre_Falha_Perigosa_1 is ltl [] not(probe and
causa_1 and not(efeito_1))
    assert Livre_Falha_Perigosa_1

    property Livre_Falha_Segura_5 is ltl [] not(probe and
not(causa_1) and efeito_1)
    assert Livre Falha Segura 5
```

APÊNDICE B - Código FIACRE completo para *model checking* da aplicação da MB.

```

const    TEMP_H2O_RESF_MCP:    array    9    of    nat    is
[44,45,46,83,84,85,89,90,91]    // var it
const    DEMANDA_MCP:    array    6    of    nat    is    [599,600,601,699,700,701]
//VAR id
const    ROTACAO_MCP:    array    10    of    nat    is
[0,589,590,591,599,600,601,699,700,701] //VAR ir

process diagram
[portDI0_IN: out bool, portDI0_Q: in bool, portDI1_IN: out bool,
portDI1_Q: in bool, portDI2_IN: out bool, portDI2_Q: in
bool, portDI3_IN: out bool, portDI3_Q: in bool, portDI4_IN: out
bool, portDI4_Q: in bool, portDI5_IN: out bool, portDI5_Q: in
bool, portDI6_IN: out bool, portDI6_Q: in bool, portDI7_IN: out
bool, portDI7_Q: in bool, portDI8_IN: out bool, portDI8_Q: in
bool]
    is
        states    UPDATE_INPUTS0, UPDATE_INPUTS1,    UPDATE_OUTPUTS0,
UPDATE_OUTPUTS1, UPDATE_OUTPUTS2, UPDATE_OUTPUTS3, UPDATE_OUTPUTS4
, UPDATE_OUTPUTS5, UPDATE_OUTPUTS6, UPDATE_OUTPUTS7,
UPDATE_OUTPUTS8, UPDATE_OUTPUTS9, UPDATE_OUTPUTS10, UPDATE_OUTPUTS
11, UPDATE_OUTPUTS12, UPDATE_OUTPUTS13, UPDATE_OUTPUTS14, UPDATE OU
TPUTS15,    UPDATE_OUTPUTS16, UPDATE_OUTPUTS17,    UPDATE_OUTPUTS18,
UPDATE_OUTPUTS19, UPDATE_OUTPUTS20, PROBE, WAITING

        var    it:0..8:=0,
            id:0..5:=0,
            ir:0..9:=0,
            ER:0..2:=0, // [ACOPLADA AV, ACOPLADA AR, DESACOPLADA]
            STARTER_MCP_EM_OFF: bool := false,
            ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER:    bool    :=
false,
            ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP:    bool    :=
false,
            ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP:    bool    :=
false,
            ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP:    bool    :=
false,
            ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO:    bool
:= false,
            ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP:    bool
:= false,
            ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER:    bool    :=
false,
            ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER:    bool    :=
false,
            PRDA_EMERG_MCP: bool := false,

            MCP_EM_OPERACAO_LOCAL: bool := false,
            BOMBA_AGUA_SALGADA_LIGADA: bool := false,

```

```
VALVULA_BOMBA_AGUA_SALGADA_ABERTA: bool := false,
MANETE_EXTERNA_HABILITADA: bool := false,
CHAVE_BATERIA_MCP_LIGADA: bool := false,
START_MCP_SUPERVISORIO: bool := false,
DEMANDA_FORCADA_MCP: bool := false,
ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP: bool :=
false,
TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45: bool := false,
TEMP_H2O_RESF_MCP_MENOR_IGUAL_85: bool := true,
ER_ACOPLADA_AV: bool := false,
ER_ACOPLADA_AR: bool := false,
ER_DESACOPLADA: bool := true,
DEMANDA_MCP_MENOR_600: bool := false,
DEMANDA_MCP_MAIOR_IGUAL_600: bool := true,
DEMANDA_MCP_MAIOR_600: bool := false,
DEMANDA_MCP_MENOR_700: bool := false,
DEMANDA_MCP_MENOR_IGUAL_700: bool := false,
DEMANDA_MCP_IGUAL_600: bool := true,
ROTACAO_MCP_IGUAL_0: bool := true,
ROTACAO_MCP_MENOR_590: bool := false,
ROTACAO_MCP_MAIOR_IGUAL_600: bool := false,
ROTACAO_MCP_MENOR_700: bool := false,
ROTACAO_MCP_MENOR_IGUAL_700: bool := false,
INTERLOCK_PARTIDA_MCP: bool := false,
ALARME_MCP_ESPERA_ENTRE_PARTIDAS: bool := false,
COMANDO_START_MCP: bool := false,
INTERLOCK_PARADA_NORMAL_MCP: bool := false,
MCP_OFF: bool := false,
MCP_STOPPING: bool := false,
INTERLOCK_ACOPLAR_AV_AR: bool := false,
INTERLOCK_DESACOPLAR: bool := false,
(BLD/1/value aux_obs): bool := false,
DI1_Q: bool := false,
DI2_Q: bool := false,
DI3_Q: bool := false,
DI4_Q: bool := false,
DI5_Q: bool := false,
DI6_Q: bool := false,
DI7_Q: bool := false,
DI8_Q: bool := false
```

```
init to UPDATE_OUTPUTS0
```

```
from UPDATE_INPUTS0
```

```
ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER:=any;
ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP:=any;
ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP:=any;
ALARME_NIVEL_BAIIXO_AGUA_RESFRIAMENTO_DO_MCP:=any;
ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO:=any;
ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP:=any;
STARTER_MCP_EM_OFF:=any;
ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER:=any;
ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER:=any;
PRDA_EMERG_MCP:=any;
MCP_EM_OPERACAO_LOCAL:=any;
```

```

BOMBA_AGUA_SALGADA_LIGADA:=any;
VALVULA_BOMBA_AGUA_SALGADA_ABERTA:=any;
MANETE_EXTERNA_HABILITADA:=any;
CHAVE_BATERIA_MCP_LIGADA:=any;
START_MCP_SUPERVISORIO:=any;
DEMANDA_FORCADA_MCP:=any;
it:=any;
id:=any;
ir:=any;
STARTER_MCP_EM_OFF:=any;
ER:=any;
wait[0,0];

to UPDATE_INPUTS1

from UPDATE_INPUTS1

ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP:=(TEMP_H2O_RESF_MCP[it]>90);

TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45:=(TEMP_H2O_RESF_MCP[it]>=45);

TEMP_H2O_RESF_MCP_MENOR_IGUAL_85:=(TEMP_H2O_RESF_MCP[it]<=85);
ER_ACOPLADA_AV:=(ER=0);
ER_ACOPLADA_AR:=(ER=1);
ER_DESACOPLADA:=(ER=2);
DEMANDA_MCP_MENOR_600:=(DEMANDA_MCP[id]<600);
DEMANDA_MCP_MAIOR_IGUAL_600:=(DEMANDA_MCP[id]>=600);
DEMANDA_MCP_MAIOR_600:=(DEMANDA_MCP[id]>600);
DEMANDA_MCP_MENOR_700:=(DEMANDA_MCP[id]<700);
DEMANDA_MCP_MENOR_IGUAL_700:=(DEMANDA_MCP[id]<=700);
DEMANDA_MCP_IGUAL_600:=(DEMANDA_MCP[id]=600);
ROTACAO_MCP_IGUAL_0:=(ROTACAO_MCP[ir]=0);
ROTACAO_MCP_MENOR_590:=(ROTACAO_MCP[ir]<590);
ROTACAO_MCP_MAIOR_IGUAL_600:=(ROTACAO_MCP[ir]>=600);
ROTACAO_MCP_MENOR_700:=(ROTACAO_MCP[ir]<700);
ROTACAO_MCP_MENOR_IGUAL_700:=(ROTACAO_MCP[ir]<=700);
wait[0,0];
to UPDATE_OUTPUTS0

from UPDATE_OUTPUTS0
INTERLOCK_PARTIDA_MCP:=(not(ALARME_BAIXA_PRESSAO_OLEO_CONTR
OLE_DA_ER or ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP or
ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP or
ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP or
ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP or
ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO or
ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP) and
not(ALARME_MCP_ESPERA_ENTRE_PARTIDAS) and
not(PRDA_EMERG_MCP) and not(MCP_EM_OPERACAO_LOCAL) and
ER_DESACOPLADA and BOMBA_AGUA_SALGADA_LIGADA and
VALVULA_BOMBA_AGUA_SALGADA_ABERTA and
MANETE_EXTERNA_HABILITADA and STARTER_MCP_EM_OFF and

```

```

CHAVE_BATERIA_MCP_LIGADA                                and
TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45                      and
TEMP_H2O_RESF_MCP_MENOR_IGUAL_85 and DEMANDA_MCP_IGUAL_600
and ROTACAO_MCP_IGUAL_0);
wait[0,0];
to UPDATE_OUTPUTS1

from UPDATE_OUTPUTS1
portDI0_IN!      not(ALARME_MCP_ESPERA_ENTRE_PARTIDAS) and
ROTACAO_MCP_IGUAL_0      and      ER_DESACOPADA      and
INTERLOCK_PARTIDA_MCP   and      START_MCP_SUPERVISORIO and
STARTER_MCP_EM_OFF;
aux_obs:=      not(ALARME_MCP_ESPERA_ENTRE_PARTIDAS) and
ROTACAO_MCP_IGUAL_0      and      ER_DESACOPADA      and
INTERLOCK_PARTIDA_MCP   and      START_MCP_SUPERVISORIO and
STARTER_MCP_EM_OFF;
to UPDATE_OUTPUTS2

from UPDATE_OUTPUTS2
portDI0_Q? COMANDO_START_MCP;
to UPDATE_OUTPUTS3

from UPDATE_OUTPUTS3
portDI1_IN! COMANDO_START_MCP;
to UPDATE_OUTPUTS4

from UPDATE_OUTPUTS4
portDI1_Q? DI1_Q;
ALARME_MCP_ESPERA_ENTRE_PARTIDAS:=      DI1_Q      and
ROTACAO_MCP_MENOR_590;
to UPDATE_OUTPUTS5

from UPDATE_OUTPUTS5
portDI2_IN!      ROTACAO_MCP_MAIOR_IGUAL_600      and
ROTACAO_MCP_MENOR_700;
to UPDATE_OUTPUTS6

from UPDATE_OUTPUTS6
portDI2_Q? DI2_Q;
INTERLOCK_PARADA_NORMAL_MCP:=      DI2_Q      and
not(MCP_EM_OPERACAO_LOCAL) and DEMANDA_MCP_MAIOR_IGUAL_600
and DEMANDA_MCP_MENOR_700;
to UPDATE_OUTPUTS7

from UPDATE_OUTPUTS7
DEMANDA_FORCADA_MCP:=((DEMANDA_FORCADA_MCP or
(DEMANDA_MCP_MAIOR_600 and not(ER_DESACOPADA) and
PRDA_EMERG_MCP)) and not(MCP_OFF));
wait[0,0];
to UPDATE_OUTPUTS8

from UPDATE_OUTPUTS8
portDI3_IN! DEMANDA_FORCADA_MCP;
to UPDATE_OUTPUTS9

```

```
from UPDATE_OUTPUTS9
  portDI3_Q? DI3_Q;
  MCP_STOPPING:= (MCP_STOPPING or (DI3_Q or (ER_DESACOPLADA
and PRDA_EMERG_MCP)) and not(MCP_OFF));
  to UPDATE_OUTPUTS10

from UPDATE_OUTPUTS10
  MCP_OFF:= MCP_STOPPING and ROTACAO_MCP_IGUAL_0;
  wait[0,0];
  to UPDATE_OUTPUTS11

from UPDATE_OUTPUTS11
  portDI4_IN! ER_DESACOPLADA;
  to UPDATE_OUTPUTS12

from UPDATE_OUTPUTS12
  portDI5_IN! ER_ACOPLADA_AV;
  to UPDATE_OUTPUTS13

from UPDATE_OUTPUTS13
  portDI6_IN! ER_ACOPLADA_AR;
  to UPDATE_OUTPUTS14

from UPDATE_OUTPUTS14
  portDI4_Q? DI4_Q;
  to UPDATE_OUTPUTS15

from UPDATE_OUTPUTS15
  portDI5_Q? DI5_Q;
  to UPDATE_OUTPUTS16

from UPDATE_OUTPUTS16
  portDI6_Q? DI6_Q;
  INTERLOCK_ACOPLAR_AV_AR:=(DI4_Q or DI5_Q or DI6_Q) and
not(MCP_EM_OPERACAO_LOCAL or
ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER or
ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER or
ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER) and
not(PRDA_EMERG_MCP) and ROTACAO_MCP_MAIOR_IGUAL_600 and
ROTACAO_MCP_MENOR_IGUAL_700 and DEMANDA_MCP_MAIOR_IGUAL_600
and DEMANDA_MCP_MENOR_IGUAL_700;
  to UPDATE_OUTPUTS17

from UPDATE_OUTPUTS17
  portDI7_IN! ER_ACOPLADA_AV;
  to UPDATE_OUTPUTS18

from UPDATE_OUTPUTS18
  portDI8_IN! ER_ACOPLADA_AR;
  to UPDATE_OUTPUTS19

from UPDATE_OUTPUTS19
```

```

portDI7_Q? DI7_Q;
to UPDATE_OUTPUTS20

from UPDATE_OUTPUTS20
portDI8_Q? DI8_Q;
INTERLOCK_DESACOPRAR:=(DI7_Q or DI8_Q) and
ROTACAO_MCP_MAIOR_IGUAL_600 and ROTACAO_MCP_MENOR_IGUAL_700 and
DEMANDA_MCP_MAIOR_IGUAL_600 and DEMANDA_MCP_MENOR_IGUAL_700;
to PROBE

from PROBE
wait[0,0];
to WAITING
from WAITING
to UPDATE_INPUTS0

////////////////////////////////////
////////// DI:
process DI
[portIN: in bool, portQ: out bool, portTimer: sync]
is
states idle, running, elapsed
var IN: bool := false,
    Q: bool := false
init
to idle
from idle
select
portIN? IN;
if IN then
to running
else
loop
end
[]
portQ! Q;
loop
end

from running
select
portIN? IN;
if not IN then
to idle
else
loop
end
[]
portTimer;
Q := true;
to elapsed
[]
portQ! Q;
loop

```

```

end
from elapsed
select
  portIN? IN;
  if not IN then
    Q := false;
    to idle
  else
    loop
  end
[]
  portQ! Q;
  loop
end

////////////////////////////////////
//////////////////////////////////// BLD:

component BLD
is
  port portDI0_IN: in out bool in [0,0],
    portDI0_Q: in out bool in [0,0],
    portDI0_Timer: sync in [2,2],
    portDI1_IN: in out bool in [0,0],
    portDI1_Q: in out bool in [0,0],
    portDI1_Timer: sync in [15,15],
    portDI2_IN: in out bool in [0,0],
    portDI2_Q: in out bool in [0,0],
    portDI2_Timer: sync in [240,240],
    portDI3_IN: in out bool in [0,0],
    portDI3_Q: in out bool in [0,0],
    portDI3_Timer: sync in [3,3],
    portDI4_IN: in out bool in [0,0],
    portDI4_Q: in out bool in [0,0],
    portDI4_Timer: sync in [2,2],
    portDI5_IN: in out bool in [0,0],
    portDI5_Q: in out bool in [0,0],
    portDI5_Timer: sync in [2,2],
    portDI6_IN: in out bool in [0,0],
    portDI6_Q: in out bool in [0,0],
    portDI6_Timer: sync in [2,2],
    portDI7_IN: in out bool in [0,0],
    portDI7_Q: in out bool in [0,0],
    portDI7_Timer: sync in [2,2],
    portDI8_IN: in out bool in [0,0],
    portDI8_Q: in out bool in [0,0],
    portDI8_Timer: sync in [2,2]
priority portDI0_Timer > portDI0_Q,
  portDI1_Timer > portDI1_Q,
  portDI2_Timer > portDI2_Q,
  portDI3_Timer > portDI3_Q,
  portDI4_Timer > portDI4_Q,
  portDI5_Timer > portDI5_Q,

```

```

portDI6_Timer > portDI6_Q,
portDI7_Timer > portDI7_Q,
portDI8_Timer > portDI8_Q
    par * in
        diagram
[portDI0_IN,portDI0_Q,portDI1_IN,portDI1_Q,portDI2_IN,
portDI2_Q,portDI3_IN,portDI3_Q,portDI4_IN,portDI4_Q,portDI5_IN,
portDI5_Q,portDI6_IN,portDI6_Q,portDI7_IN,portDI7_Q,portDI8_IN,
portDI8_Q]
|| DI [portDI0_IN,portDI0_Q,portDI0_Timer]
|| DI [portDI1_IN,portDI1_Q,portDI1_Timer]
|| DI [portDI2_IN,portDI2_Q,portDI2_Timer]
|| DI [portDI3_IN,portDI3_Q,portDI3_Timer]
|| DI [portDI4_IN,portDI4_Q,portDI4_Timer]
|| DI [portDI5_IN,portDI5_Q,portDI5_Timer]
|| DI [portDI6_IN,portDI6_Q,portDI6_Timer]
|| DI [portDI7_IN,portDI7_Q,portDI7_Timer]
|| DI [portDI8_IN,portDI8_Q,portDI8_Timer]
end

```

BLD

```

////////////////////////////////////
////////// OBSERVER:

```

```

process timed_input [ITrue,IFalse,T: sync] is
    states varFalse , varTrueUntimed , varTrueTimed
    init to
        varFalse
    from varFalse
        ITrue; to varTrueUntimed
    from varTrueUntimed select
        T; to varTrueTimed
    [] IFalse; to varFalse
    end
    from varTrueTimed
        IFalse; to varFalse

```

```
// Here is every Fcrtimer info in this list...
```

```

component obs_DI0 is
    port ITrue: sync in [0,0] is (BLD/1/value aux_obs),
        IFalse: sync in [0,0] is not(BLD/1/value aux_obs),
        T: sync in [2,2]
    priority IFalse > T
    par
        timed_input [ITrue,IFalse,T]
    end

```

```
obs_DI0 > BLD
```

```
component obs_DI1 is
```

```

    port ITrue: sync in [0,0] is (BLD/1/value
COMANDO_START_MCP),
    IFalse: sync in [0,0] is not(BLD/1/value
COMANDO_START_MCP),
    T: sync in [15,15]
    priority IFalse > T
    par
        timed_input [ITrue,IFalse,T]
    end

```

```
obs_DI1 > BLD
```

```

component obs_DI2 is
    port ITrue: sync in [0,0] is (BLD/1/value
ROTACAO_MCP_MAIOR_IGUAL_600
ROTACAO_MCP_MENOR_700),
    IFalse: sync in [0,0] is not(BLD/1/value
ROTACAO_MCP_MAIOR_IGUAL_600
ROTACAO_MCP_MENOR_700),
    T: sync in [240,240]
    priority IFalse > T
    par
        timed_input [ITrue,IFalse,T]
    end

```

```
obs_DI2 > BLD
```

```

component obs_DI3 is
    port ITrue: sync in [0,0] is (BLD/1/value
DEMANDA_FORCADA_MCP),
    IFalse: sync in [0,0] is not(BLD/1/value
DEMANDA_FORCADA_MCP),
    T: sync in [3,3]
    priority IFalse > T
    par
        timed_input [ITrue,IFalse,T]
    end

```

```
obs_DI3 > BLD
```

```

component obs_DI4 is
    port ITrue: sync in [0,0] is (BLD/1/value ER_DESACOPLADA),
    IFalse: sync in [0,0] is not(BLD/1/value
ER_DESACOPLADA),
    T: sync in [2,2]
    priority IFalse > T
    par
        timed_input [ITrue,IFalse,T]
    end

```

```
obs_DI4 > BLD
```

```
component obs_DI5 is
  port ITrue: sync in [0,0] is (BLD/1/value ER_ACOPLADA_AV),
     IFalse: sync in [0,0] is not(BLD/1/value
ER_ACOPLADA_AV),
     T: sync in [2,2]
  priority IFalse > T
  par
    timed_input [ITrue,IFalse,T]
  end
```

```
obs_DI5 > BLD
```

```
component obs_DI6 is
  port ITrue: sync in [0,0] is (BLD/1/value ER_ACOPLADA_AR),
     IFalse: sync in [0,0] is not(BLD/1/value
ER_ACOPLADA_AR),
     T: sync in [2,2]
  priority IFalse > T
  par
    timed_input [ITrue,IFalse,T]
  end
```

```
obs_DI6 > BLD
```

```
component obs_DI7 is
  port ITrue: sync in [0,0] is (BLD/1/value ER_ACOPLADA_AV),
     IFalse: sync in [0,0] is not(BLD/1/value
ER_ACOPLADA_AV),
     T: sync in [2,2]
  priority IFalse > T
  par
    timed_input [ITrue,IFalse,T]
  end
```

```
obs_DI7 > BLD
```

```
component obs_DI8 is
  port ITrue: sync in [0,0] is (BLD/1/value ER_ACOPLADA_AR),
     IFalse: sync in [0,0] is not(BLD/1/value
ER_ACOPLADA_AR),
     T: sync in [2,2]
  priority IFalse > T
  par
    timed_input [ITrue,IFalse,T]
  end
```

```
obs_DI8 > BLD
```

```
////////// VERIFICAÇÃO //////////
property probe is ltl (BLD/1/state PROBE)
property OBS_DI0 is ltl (obs_DI0/1/state varTrueTimed)
property OBS_DI1 is ltl (obs_DI1/1/state varTrueTimed)
property OBS_DI2 is ltl (obs_DI2/1/state varTrueTimed)
property OBS_DI3 is ltl (obs_DI3/1/state varTrueTimed)
property OBS_DI4 is ltl (obs_DI4/1/state varTrueTimed)
property OBS_DI5 is ltl (obs_DI5/1/state varTrueTimed)
property OBS_DI6 is ltl (obs_DI6/1/state varTrueTimed)
property OBS_DI7 is ltl (obs_DI7/1/state varTrueTimed)
property OBS_DI8 is ltl (obs_DI8/1/state varTrueTimed)
property ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER is ltl
(BLD/1/value ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER )
property ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP is ltl
(BLD/1/value ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP)
property ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP is ltl
(BLD/1/value ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP)
property ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP is ltl
(BLD/1/value ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP)
property ALARME_NIVEL_BAIIXO_AGUA_RESFRIAMENTO_DO_MCP is ltl
(BLD/1/value ALARME_NIVEL_BAIIXO_AGUA_RESFRIAMENTO_DO_MCP)
property ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO is
ltl (BLD/1/value
ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO)
property ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP is
ltl (BLD/1/value
ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP)
property ER_ACOPLADA_AV is ltl (BLD/1/value ER_ACOPLADA_AV)
property ER_ACOPLADA_AR is ltl (BLD/1/value ER_ACOPLADA_AR)
property ALARME_MCP_ESPERA_ENTRE_PARTIDAS is ltl (BLD/1/value
ALARME_MCP_ESPERA_ENTRE_PARTIDAS)
property PRDA_EMERG_MCP is ltl (BLD/1/value PRDA_EMERG_MCP)
property ER_DESACOPLADA is ltl (BLD/1/value ER_DESACOPLADA)
property BOMBA_AGUA_SALGADA_LIGADA is ltl (BLD/1/value
BOMBA_AGUA_SALGADA_LIGADA)
property VALVULA_BOMBA_AGUA_SALGADA_ABERTA is ltl (BLD/1/value
VALVULA_BOMBA_AGUA_SALGADA_ABERTA)
property MANETE_EXTERNA_HABILITADA is ltl (BLD/1/value
MANETE_EXTERNA_HABILITADA)
property STARTER_MCP_EM_OFF is ltl (BLD/1/value
STARTER_MCP_EM_OFF)
property CHAVE_BATERIA_MCP_LIGADA is ltl (BLD/1/value
CHAVE_BATERIA_MCP_LIGADA)
property TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45 is ltl (BLD/1/value
TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45)
property TEMP_H2O_RESF_MCP_MENOR_IGUAL_85 is ltl (BLD/1/value
TEMP_H2O_RESF_MCP_MENOR_IGUAL_85)
property DEMANDA_MCP_IGUAL_600 is ltl (BLD/1/value
DEMANDA_MCP_IGUAL_600)
property ROTACAO_MCP_IGUAL_0 is ltl (BLD/1/value
ROTACAO_MCP_IGUAL_0)
property INTERLOCK_PARTIDA_MCP is ltl (BLD/1/value
INTERLOCK_PARTIDA_MCP)
```

```

property  ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER  is  ltl
(BLD/1/value  ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER)
property  ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER  is  ltl
(BLD/1/value  ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER)
property  COMANDO_START_MCP  is  ltl  (BLD/1/value
COMANDO_START_MCP)
property  DEMANDA_FORCADA_MCP  is  ltl  (BLD/1/value
DEMANDA_FORCADA_MCP)
property  DEMANDA_MCP_MAIOR_600  is  ltl  (BLD/1/value
DEMANDA_MCP_MAIOR_600)
property  DEMANDA_MCP_MAIOR_IGUAL_600  is  ltl  (BLD/1/value
DEMANDA_MCP_MAIOR_IGUAL_600)
property  DEMANDA_MCP_MENOR_600  is  ltl  (BLD/1/value
DEMANDA_MCP_MENOR_600)
property  DEMANDA_MCP_MENOR_700  is  ltl  (BLD/1/value
DEMANDA_MCP_MENOR_700)
property  DEMANDA_MCP_MENOR_IGUAL_700  is  ltl  (BLD/1/value
DEMANDA_MCP_MENOR_IGUAL_700)
property  INTERLOCK_ACOPLAR_AV_AR  is  ltl  (BLD/1/value
INTERLOCK_ACOPLAR_AV_AR)
property  INTERLOCK_DESACOPLAR  is  ltl  (BLD/1/value
INTERLOCK_DESACOPLAR)
property  INTERLOCK_PARADA_NORMAL_MCP  is  ltl  (BLD/1/value
INTERLOCK_PARADA_NORMAL_MCP)
property  MCP_EM_OPERACAO_LOCAL  is  ltl  (BLD/1/value
MCP_EM_OPERACAO_LOCAL)
property  MCP_OFF  is  ltl  (BLD/1/value  MCP_OFF)
property  MCP_STOPPING  is  ltl  (BLD/1/value  MCP_STOPPING)
property  ROTACAO_MCP_MAIOR_IGUAL_600  is  ltl  (BLD/1/value
ROTACAO_MCP_MAIOR_IGUAL_600)
property  ROTACAO_MCP_MENOR_590  is  ltl  (BLD/1/value
ROTACAO_MCP_MENOR_590)
property  ROTACAO_MCP_MENOR_700  is  ltl  (BLD/1/value
ROTACAO_MCP_MENOR_700)
property  ROTACAO_MCP_MENOR_IGUAL_700  is  ltl  (BLD/1/value
ROTACAO_MCP_MENOR_IGUAL_700)
property  START_MCP_SUPERVISORIO  is  ltl  (BLD/1/value
START_MCP_SUPERVISORIO)

```

```

////////////////////////////////////
property  causa_1  is  ltl
(not(ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER)  and
not(ALARME_FALHA_CARREGAMENTO_BATERIA_DO_MCP)  and
not(ALARME_ALTA_TEMPERATURA_RESFRIAMENTO_DO_MCP)  and
not(ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DO_MCP)  and
not(ALARME_NIVEL_BAIXO_AGUA_RESFRIAMENTO_DO_MCP)  and
not(ALARME_FILTRO_OLEO_LUBRIFICANTE_DO_MCP_OBSTRUIDO)  and
not(ALARME_ALTA_TEMPERATURA_OLEO_LUBRIFICANTE_DO_MCP)  and
ER_DESACOPLADA  and  not(ALARME_MCP_ESPERA_ENTRE_PARTIDAS)  and
not(PRDA_EMERG_MCP)  and
not(MCP_EM_OPERACAO_LOCAL)  and  STARTER_MCP_EM_OFF  and
BOMBA_AGUA_SALGADA_LIGADA  and  VALVULA_BOMBA_AGUA_SALGADA_ABERTA
and  MANETE_EXTERNA_HABILITADA  and  CHAVE_BATERIA_MCP_LIGADA  and
TEMP_H2O_RESF_MCP_MAIOR_IGUAL_45  and

```

```

TEMP_H2O_RESF_MCP_MENOR_IGUAL_85 and DEMANDA_MCP_IGUAL_600 and
ROTACAO_MCP_IGUAL_0)
property efeito_1 is ltl (INTERLOCK_PARTIDA_MCP)

property Livre_Falha_Perigosa_1 is ltl [] not(probe and causa_1
and not(efeito_1))
assert Livre_Falha_Perigosa_1

property Livre_Falha_Segura_1 is ltl [] not(probe and
not(causa_1) and efeito_1)
assert Livre_Falha_Segura_1

////////////////////////////////////
property causa_2 is ltl ((ER_DESACOPLADA and
not(ALARME_MCP_ESPERA_ENTRE_PARTIDAS) and STARTER_MCP_EM_OFF
and ROTACAO_MCP_IGUAL_0 and INTERLOCK_PARTIDA_MCP and
START_MCP_SUPERVISORIO) and OBS_DI0)
property efeito_2 is ltl (COMANDO_START_MCP)

property Livre_Falha_Perigosa_2 is ltl [] not(probe and causa_2
and not(efeito_2))
assert Livre_Falha_Perigosa_2

property Livre_Falha_Segura_2 is ltl [] not(probe and
not(causa_2) and efeito_2)
assert Livre_Falha_Segura_2
////////////////////////////////////
property causa_3 is ltl ((ROTACAO_MCP_MENOR_590 and OBS_DI1)
property efeito_3 is ltl (ALARME_MCP_ESPERA_ENTRE_PARTIDAS)

property Livre_Falha_Perigosa_3 is ltl [] not(probe and causa_3
and not(efeito_3))
assert Livre_Falha_Perigosa_3

property Livre_Falha_Segura_3 is ltl [] not(probe and
not(causa_3) and efeito_3)
assert Livre_Falha_Segura_3
////////////////////////////////////
property causa_4 is ltl (not(MCP_EM_OPERACAO_LOCAL) and
DEMANDA_MCP_MAIOR_IGUAL_600 and DEMANDA_MCP_MENOR_700 and
((ROTACAO_MCP_MAIOR_IGUAL_600 and ROTACAO_MCP_MENOR_700) and
OBS_DI2))
property efeito_4 is ltl (INTERLOCK_PARADA_NORMAL_MCP)

property Livre_Falha_Perigosa_4 is ltl [] not(probe and causa_4
and not(efeito_4))
assert Livre_Falha_Perigosa_4

property Livre_Falha_Segura_4 is ltl [] not(probe and
not(causa_4) and efeito_4)
assert Livre_Falha_Segura_4
////////////////////////////////////
property causa_5 is ltl ((DEMANDA_FORCADA_MCP or
(not(ER_DESACOPLADA) and PRDA_EMERG_MCP and

```

```

DEMANDA_MCP_MAIOR_600)) and not(MCP_OFF))
property efeito_5 is ltl (DEMANDA_FORCADA_MCP)

property Livre_Falha_Perigosa_5 is ltl [] not(probe and causa_5
and not(efeito_5))
assert Livre_Falha_Perigosa_5

property Livre_Falha_Segura_5 is ltl [] not(probe and
not(causa_5) and efeito_5)
assert Livre_Falha_Segura_5
////////////////////////////////////
property causa_6 is ltl ((MCP_STOPPING or ((ER_DESACOPLADA and
PRDA_EMERG_MCP) or (DEMANDA_FORCADA_MCP and OBS_DI3))) and
not(MCP_OFF)) //
property efeito_6 is ltl (MCP_STOPPING)

property Livre_Falha_Perigosa_6 is ltl [] not(probe and causa_6
and not(efeito_6))
assert Livre_Falha_Perigosa_6

property Livre_Falha_Segura_6 is ltl [] not(probe and
not(causa_6) and efeito_6)
assert Livre_Falha_Segura_6
////////////////////////////////////
property causa_7 is ltl (ROTACAO_MCP_IGUAL_0 and
DEMANDA_FORCADA_MCP)
property efeito_7 is ltl (MCP_OFF)

property Livre_Falha_Perigosa_7 is ltl [] not(probe and causa_7
and not(efeito_7))
assert Livre_Falha_Perigosa_7

property Livre_Falha_Segura_7 is ltl [] not(probe and
not(causa_7) and efeito_7)
assert Livre_Falha_Segura_7
////////////////////////////////////
property causa_8 is ltl ((OBS_DI4 or OBS_DI5 or OBS_DI6) and
not(ALARME_BAIXA_PRESSAO_OLEO_CONTROLE_DA_ER) and
not(PRDA_EMERG_MCP) and not(MCP_EM_OPERACAO_LOCAL) and
DEMANDA_MCP_MAIOR_IGUAL_600 and DEMANDA_MCP_MENOR_IGUAL_700 and
ROTACAO_MCP_MAIOR_IGUAL_600 and ROTACAO_MCP_MENOR_IGUAL_700 and
not(ALARME_ALTA_TEMPERATURA_OLEO_CONTROLE_DA_ER) and
not(ALARME_BAIXA_PRESSAO_OLEO_LUBRIFICANTE_DA_ER)) ///
property efeito_8 is ltl (INTERLOCK_ACOPLAR_AV_AR)

property Livre_Falha_Perigosa_8 is ltl [] not(probe and causa_8
and not(efeito_8))
assert Livre_Falha_Perigosa_8

property Livre_Falha_Segura_8 is ltl [] not(probe and
not(causa_8) and efeito_8)
assert Livre_Falha_Segura_8
////////////////////////////////////
property causa_9 is ltl ((OBS DI7 or OBS DI8) and

```

```
DEMANDA_MCP_MAIOR_IGUAL_600 and DEMANDA_MCP_MENOR_IGUAL_700 and  
ROTACAO_MCP_MAIOR_IGUAL_600 and ROTACAO_MCP_MENOR_IGUAL_700 )  
property efeito_9 is ltl (INTERLOCK_DESACOPLAR)
```

```
property Livre_Falha_Perigosa_9 is ltl [] not(probe and causa_9  
and not(efeito_9))  
assert Livre_Falha_Perigosa_9
```

```
property Livre_Falha_Segura_9 is ltl [] not(probe and  
not(causa_9) and efeito_9)  
assert Livre_Falha_Segura_9
```
