SISTEMA ESPECIALISTA DE TEMPO REAL PARA AUXILIAR O DIAGNÓSTICO DAS CONDIÇÕES LIMITE DE OPERAÇÃO DA USINA NUCLEAR DE ANGRA 2

André Diamante Schechter

Projeto de Graduação apresentado ao Curso de Engenharia Nuclear da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários a obtenção do título de Engenheiro.

Orientador: Roberto Schirru

Rio de Janeiro

Setembro de 2017

SISTEMA ESPECIALISTA DE TEMPO REAL PARA AUXILIAR O DIAGNÓSTICO DAS CONDIÇÕES LIMITE DE OPERAÇÃO

DA USINA NUCLEAR DE ANGRA 2

André Diamante Schechter

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA NUCLEAR DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO NUCLEAR.

Examinado por:

_____

Prof. Roberto Schirru, D. Sc.

_____

Prof. Eduardo Gomes Dutra do Carmo, D. Sc.

_____

Prof. Claudio Marcio de Nascimento Abreu Pereira, D. Sc.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2017

Schechter, André D.

Sistema especialista de tempo real para auxiliar no diagnóstico de condições limites de operação da usina nuclear de Angra 2 / André D. Schechter – Rio de Janeiro: UFRJ / ESCOLA POLITÉCNICA, 2017.

VIII, 52 p.: il.; 29,7 cm.

Orientador: Roberto Schirru

Projeto de Graduação – UFRJ/ POLI/ Engenharia Nuclear, 2017.

Referências Bibliográficas: p. 40-41.

1. Sistema Especialista. 2. Condições Limites de Operação. 3. Árvore de Falha. I. Schirru, Roberto. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Nuclear. III. Sistema especialista de tempo real para auxiliar no diagnóstico de condições limites de operação da usina nuclear de Angra 2.

AGRADECIMENTO

Agradeço a todos que me auxiliaram na realização deste trabalho, da equipe de limpeza até a equipe do restaurante universitário. Sem eles, o dia a dia na universidade não seria possível. Também gostaria de agradecer a todos os meus colegas de curso que me apoiaram e me auxiliaram a chegar até aqui. Como são muitos, os representarei nomeando o Robson e o João.

Agradeço a Andressa, que me ajudou muito na revisão deste trabalho, e a todos os colegas do LMP, principalmente ao professor e orientador Roberto Schirru, que soube me ensinar e me cobrar de acordo com o que minha vida pessoal permitia no momento. Os ensinamentos foram muito além do escopo do curso de engenharia nuclear e foram, sem dúvidas, muito valiosos.

Agradeço à Déborah, minha mãe, por ter me formado como pessoa, por toda paciência e incentivo para que esse trabalho fosse feito da melhor maneira possível. Sem essa ajuda não conseguiria chegar ao fim do curso de engenharia nuclear.

Agradeço ao Hélio, meu pai, por ter me incentivado em todos os momentos da faculdade, me passando ensinamentos que foram muito além das fórmulas de física. Esses ensinamentos que moldaram meu jeito de ver o mundo.

Agradeço à Domenica, por estar sempre ao meu lado para me apoiar e me incentivar nos momentos mais importantes. Tudo que foi feito para me ajudar a concluir este trabalha ficará para sempre em minha memória.

Por fim, agradeço ao Fábio, à minha família e a todos os professores que tive na minha formação desde o meu primeiro dia de aula no pré-maternal.

“Educar é impregnar de sentido o que fazemos a cada instante! ”

Paulo Freie

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Nuclear.

SISTEMA ESPECIALISTA DE TEMPO REAL PARA AUXILIAR O DIAGNÓSTICO DAS CONDIÇÕES LIMITE DE OPERAÇÃO DA USINA NUCLEAR DE ANGRA 2

André Diamante Schechter

Setembro / 2017

Orientador: Roberto Schirru

Curso: Engenharia Nuclear

Os Sistemas Especialistas (SE) são amplamente utilizados para auxiliar o usuário comum a atuar como especialista em um assunto específico. A informação é salva no banco de dados do programa e o usuário introduz os fatos a serem interpretados em tempo real. O motor de inferência analisa esses fatos com base no conhecimento retido e retorna a resposta ao usuário. Neste trabalho, um SE, de tempo real, é usado para tratar as Condições Limite de Operação (CLO) da Usina Nuclear de Angra 2. Nesta usina PWR, todas as informações das CLOs provêm de árvores de falha e o SE foi adaptado para lidar com fatos de valores Verdadeiro, Falso, Nulo e Inválido, da árvore de falha. A literatura mostra que experimentos com SE aplicado em CLOs de usinas nucleares e simulações feitas para Angra 2 mostram que o SE é um modelo computacional eficiente para lidar com este tipo de problema.

Palavras-chave: sistema especialista, condições limite de operação, PWR, árvores de falha, tempo real.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Nuclear Engineer.

REAL-TIME EXPERT SYSTEM FOR ASSISTANCE IN THE DIAGNOSIS OF LIMITING CONDITIONS OF OPERATION IN THE ANGRA 2 NUCLEAR POWER PLANT

André Diamante Schechter

September / 2017

Advisor: Roberto Schirru

Course: Nuclear Engineering

Expert Systems (ES) are widely used to help a common user act as an expert on a particular subject. The information is saved in the program database and the user introduces facts to be interpreted in real time. The inference engine analyze these facts based on the knowledge retained and return the answer of the expert. In this article, an ES, in real time, is used to treat the Limiting Conditions for Operation (LCO) of the Angra 2 Nuclear Power Plant. In this PWR plant, all information from the LCOs comes from failure trees and the ES was adapted to be able to handle the facts with the values True, False, None and Null of the tree. The literature shows that experiments with ES applied in LCOs of nuclear power plant and simulations made for Angra 2 show that the ES is an efficient computational model to deal with this type of problem.

Keywords: Expert Systems, Limiting Conditions for Operation, PWR, Failure Trees, Real Time.

Sumário

CAPÍTULO 1
INTRODUÇÃO

A energia elétrica provinda de usinas nucleares é de fundamental importância para alguns países que não dispõe de outros recursos naturais, como França e Japão. Além disso, ela é uma fonte de energia que não lança gases tóxicos ao meio ambiente e não é sazonal, ou seja, pode ter uma produção contínua ao longo do ano. A Agência internacional de Energia Atômica (IAEA) informa que, em junho de 2017, existiam 449 reatores nucleares em operação no mundo para fins de produção de energia elétrica, sendo 2 deles no Brasil, com um potencial elétrico de 392 GWe.

Um dos grandes problemas do setor nuclear no mundo é a aceitação pública. Isso se deve ao fato de que os acidentes que já ocorreram, como Chernobyl e Fukushima, geraram um grande impacto ambiental. Hoje em dia, as grandes empresas do setor fazem consideráveis investimentos financeiros para tentar conter o preconceito da população com essa fonte de energia.

Como os investimentos para evitar acidentes também são muito altos, com o tempo foram desenvolvidos novos sistemas de segurança, como os passivos, que funcionam de maneira intrínseca quando alguma falha ocorre. Aliado a isso, é necessário evitar ao máximo o erro humano, visto que foi provado que em alguns acidentes, como Chernobyl e Three Mile Island, essa foi uma das principais causas dos problemas ocorridos.

Operadores de usinas nucleares trabalham, em caso de anormalidade, sob pressão, o que pode levar a tomadas de decisões equivocadas em determinadas situações. Tendo em vista a periculosidade que um acidente nuclear tem, esses possíveis equívocos devem ser evitados ou minimizados. Para isso, programas de computador tem sido cada vez mais introduzidos no dia a dia dos operadores, com objetivo de dar suporte na operação da planta em situações normais, mas principalmente em situações de emergência, facilitando a tomada de decisão e minimizando o risco de ações equivocadas. A maioria desses *softwares* faz análises em tempo real da situação da usina facilitando a interpretação dos dados disponíveis e, consequente a tomada de decisão por parte do operador, evita que o mesmo tenha que interpretar sozinho uma infinidade de dados e acabe tomando uma decisão equivocada.

Uma situação onde sistemas de suporte à operação tem sido empregados para auxiliar o operador na tomada de decisão é no controle das Condições Limites de Operação (CLO) (PAIVA, 2015), onde muitas informações têm que ser interpretadas em paralelo. Neste caso, o sistema recebe as informações sobre o status da planta e após interpretá-las, auxilia o operador nas ações a serem tomadas para que a usina continue operando de forma segura. Dessa forma, pode-se dizer que a carga sobre o operador é amenizada e a tomada de decisão é feita de forma mais clara diminuindo a chance de ocorrência de um erro humano.

CLO de uma usina nuclear são situações em que alguma situação anormal na operação está ocorrendo e o operador precisa tomar providencias para controlá-la. Todas essas situações estão definidas em documentos com o formato apresentado na figura 1, que exemplifica uma CLO da usina de Angra 1, que tem o texto similar ao de Angra 2, foco deste trabalho. Cada CLO possui ao menos uma condição, ações requeridas referentes a cada condição e tempo para conclusão para cada ação. Se todas as ações apresentadas na CLO forem seguidas corretamente, a usina voltará para operação normal ou será desligada com segurança, evitando qualquer tipo de acidente.

16.3.4 SISTEMA DE REFRIGERAÇÃO DO REATOR (SRR)

16.3.4.2 Temperatura Mínima do SRR para Criticalidade

CLO 16.3.4.2 A temperatura média (T_{med}) de ambos os circuitos do SRR deve ser $\geq$ 283 °C (541 °F).

APLICABILIDADE: MODO 1,
MODO 2, com $k_{eff} \geq 1,0$.

AÇÕES

CONDIÇÃO	AÇÃO REQUERIDA	TEMPO PARA CONCLUSÃO
A. A T_{med} de um ou ambos circuitos do SRR fora do limite.	A.1 Esteja em MODO 3	30 minutos

Figura 1 - Exemplo de texto de uma Condição Limite de Operação

As CLOs são muito importantes para a segurança da central nuclear e indicam ao operador na sala de controle tudo que ele precisa fazer para levar a usina a uma condição segura em caso de ocorrência de situações anormais. Elas são classificadas por sistema da usina e, de acordo com o modo de operação, outros parâmetros permitem o operador saber se uma CLO é aplicável ou não. No caso de ocorrência de uma situação que leve a abertura de uma CLO e, consequentemente a realização de suas respectivas ações, a agência reguladora (a CNEN no caso do Brasil) precisa ser informada. Todas as CLOs estão presentes no relatório final de análise de segurança da usina (RFAS capítulo 16).

O objetivo desse trabalho é desenvolver um sistema de apoio à decisão que seja capaz de absorver em tempo real as informações das CLOs, vindas dos sensores definidos nas árvores de falha da usina nuclear de Angra 2, interpretá-las e informar ao operador qual ação precisa ser tomada e quanto tempo ele tem para isso. A CLO de Angra 2 é baseada na árvore de falha da planta e está e está desenhada na linguagem do programa CAFTA, criado pelo grupo *Eletric Power Research Institute* (EPRI). Sendo assim, o sistema protótipo desenvolvido neste trabalho deverá ser capaz de interpretar as informações vindas da linguagem CAFTA.

Para isso, a interpretação das árvores de falhas foi tratada pelo valor dos ramos, como “VERDADEIRO”, “FALSO”, “NULO” ou “INVÁLIDO” e pelos conectores lógicos “E”, “OU”, “IGUAL”, “INVERSO”, “COMBINAÇÃO DE 2”, “COMBINAÇÃO DE 3” e “A E NÃO B”. Com essas ferramentas a árvore pode ser desenhada para conter todos os casos possíveis de abertura e violações das CLOs. Cada valor colocado em algum ramo levará a árvore para um determinado caminho até chegar a uma resposta, que no caso das CLOs é a violação ou não das mesmas.

Em função do modo de representar o conhecimento das CLOs, escolheu-se para aplicação neste trabalho o Sistema Especialista (SE), que é uma técnica de inteligência artificial, onde a informação adquirida é levada para o motor e inferência (MI), que fará a interpretação usando a base de conhecimento (BC) e devolverá a resposta para o operador. Nesse trabalho foi desenvolvido um SE na linguagem computacional Python que é capaz gerenciar as CLOs de Angra 2 através das informações vindas diretamente da base de dados do CAFTA. Essa técnica também se destaca pelo fato de conseguir absorver informações, em tempo real ou *on-line*, específicas de cada usina, o que deixa o programa muito mais preparado e adaptado para funcionar em qualquer central nuclear, podendo ser considerado um sistema computacional genérico de controle de CLOs.

A linguagem de programação Python foi usada no desenvolvimento SE, sendo escolhida por poder ser adquirida gratuitamente na internet, ser de alto nível e multi-plataforma. Além de ter fácil compreensão e facilidade para trabalhar com classes, listas e funções recursivas. Todas essas características tornam a linguagem Python uma boa ferramenta de programação para a elaboração do sistema especialista em questão e para o trabalho com árvores lógicas.

No ano de 2016 foi desenvolvido pelo Laboratório de Monitoramento de Processos COPPE/UFRJ, a partir de um contrato com a Eletronuclear, um *software*, similar ao desenvolvido neste trabalho, para a usina de Angra 1. Nele também são tratadas, em tempo real, as CLOs da usina através de um sistema especialista, porém as informações não vêm de uma árvore de falhas e sim de um texto (RFAS de Angra 1) onde são descritas as CLOs. Neste sistema computacional não foi feito um tratamento para a leitura do CAFTA e sim um tratamento para a leitura de um banco de dados baseados em regras que foram inseridas pelos próprios operadores para adaptar o sistema ao funcionamento da usina.

O sistema protótipo apresentado neste trabalho, similarmente ao sistema desenvolvido para Angra 1, usa informações da usina vindas em tempo real do Sistema Integrado de Computadores de Angra – SICA (SCHIRRU e PEREIRA, 2004), sendo cada usina com seu SICA independente. Isso o torna muito mais prático, dinâmico e integrado ao dia a dia da sala de controle da usina. Este sistema protótipo, desenvolvido na lógica de um sistema especialista, tem como característica a capacidade de ler árvores lógicas vindas da linguagem do CAFTA. Essa leitura será feita para carregar o banco de regras do SE. Após esse carregamento, as CLOs inseridas são gerenciadas em tempo real. As usinas mais recentes em operação no mundo têm suas CLOs escritas no CAFTA, sendo assim o sistema protótipo proposto pode ser visto como sendo compatível com a maioria das usinas atualmente em operação, inclusive Angra 2.

O uso de SEs para diagnosticar problemas em usinas nucleares é facilmente encontrado na literatura. Muitas vezes eles são aplicados devido ao seu alto potencial de fazer diagnósticos de problemas complexos e por se adaptar bem a especificidade de cada usina.

Salmon D. R., em sua dissertação de mestrado submetida à COPPE UFRJ, apresenta um sistema especialista que trabalha em tempo real para fazer diagnósticos em

níveis progressivos de acidentes em usinas nucleares do tipo PWR. Esse sistema identifica funcionamento anormal em 64 parâmetros da usina e faz o diagnóstico em 4 níveis progressivos até que seja identificado o acidente.

Saludes *et al*, propuseram um modelo de detecção de falhas em usinas hidrelétricas. Uma rede neural foi criada e treinada com base em dados coletados durante um ano de operação. Além disso, um sistema especialista foi criado para armazenar o conhecimento adquirido pelos operadores e auxiliar nas conclusões do diagnóstico. A combinação desses sistemas é capaz de definir se a operação está normal ou não.

Angeli e Atherton, usando conexão online com sensores, desenvolveram um sistema especialista capaz de detectar falhas em uma central hidrelétrica. Também foram usados métodos de análise de sinais e estratégias baseadas em técnicas de raciocínio profundo. Depois da interação dessas variadas fontes de informação, o sistema especialista consegue realizar o diagnóstico.

Yu Quian *et al*, propuseram um sistema especialista para realizar diagnóstico em tempo real de falhas em processos químicos. Esse sistema fornece uma sugestão ao operador de como agir quando uma situação sai da condição considerada normal.

Paiva G. V., em seu trabalho de conclusão da graduação em Engenharia Nuclear, desenvolveu um protótipo de um sistema especialista que trabalha em tempo real com o objetivo de gerenciar as condições limites de operação de uma usina nuclear, utilizando SE de tempo real, como feito para o programa da CLO de Angra 1.

Este trabalho está dividido em 7 capítulos, que estão organizados da seguinte forma:

No capítulo 2 é apresentada a fundamentação teórica deste trabalho. Nele será feita uma apresentação do conceito de inteligência artificial e será introduzida a lógica de sistema especialista, que é fundamental para a compreensão do trabalho. Será apresentada sua estrutura e como é feita a inserção de dados no mesmo, que podem ser por regras ou árvores, lógicas que serão explicadas ao longo do capítulo.

O capítulo 3 tem como objetivo apresentar o problema proposto neste trabalho. Será feita a apresentação do relatório final de análise de segurança (RFAS), que é a documento de onde serão retiradas as informações da usina nuclear de Angra 2. Em seguida, será apresentado o conceito de CLO, fundamental para auxiliar o operador da

usina a evitar ou mitigar possíveis acidentes. Para melhor compreensão, será apresentada a linguagem como é descrita a CLO no RFAS de Angra 1, apresentando exemplos. Posteriormente, a CLO de Angra 2 será apresentada e será feito um comparativo entre as CLOs das duas usinas. Por fim será apresentado o programa CAFTA, mostrando sua linguagem de escrever árvores de falha e como ele pode ser usado para representar um conjunto de CLOs.

No capítulo 4 será apresentada toda a metodologia utilizada no desenvolvimento do protótipo, demonstrando como o sistema especialista será adaptado para trabalhar em tempo real e receber informações das árvores de falha retiradas do CAFTA. Em seguida será feita uma apresentação da linguagem de programação Pyhthon juntamente com uma justificativa do motivo de sua escolha para implementação neste trabalho. Neste mesmo capítulo serão apresentadas as características internas do sistema protótipo. Nele será abordada cada parte do sistema especialista, como a base de conhecimento e o motor de inferência, descrevendo como foram adaptadas para receber as árvores de falha vindas da linguagem do CAFTA. As características internas do programa também serão apresentadas nesse capítulo, como as informações de cada parte da árvore que o operador poderá retirar do sistema protótipo assim que necessário. Serão introduzidos conceitos fundamentais para a compreensão do programa, como o conceito de profundidade de uma árvore.

No capítulo 5 é feita uma apresentação do sistema protótipo proposto, e os resultados obtidos com testes feitos com a adaptação de uma CLO da usina nuclear de Angra 2. O que comprova o bom funcionamento do sistema protótipo e a eficácia dos métodos utilizados.

O capítulo 6 apresenta as conclusões do trabalho e propostas para novas pesquisas e novos *softwares* a serem desenvolvidos.

As referências são apresentadas no capítulo 7.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Esse capítulo apresentará a fundamentação teórica desse trabalho. Começando pelo conceito de inteligência artificial, muito usado para resolver problemas, onde se precisa de grande armazenamento de informações e capacidade de realizar rapidamente cálculos e relações complexas. Também será abordado o conceito de sistema especialista, ferramenta utilizada para execução desse trabalho, apresentando sua base de dados e motor de inferência.

Para a resolução do problema proposto, controle de CLOs, é necessário compreender o conceito de árvores de falha e todos os seus detalhes. Nesse capítulo será feita uma explicação sobre esse tema e serão apresentados todos os operadores lógicos presentes na árvore de falha da usina nuclear de Angra 2, que é o foco desse trabalho.

A escolha de uma linguagem computacional adequada é de fundamental importância para o sistema computacional apresentar um bom desempenho funcionando em tempo real. Por isso, serão apresentados os motivos pelo qual o Python foi a linguagem de programação escolhida.

2.1 INTELIGÊNCIA ARTIFICIAL

Sempre foi desejo do ser humano criar uma mente que pudesse pensar sozinha e tivesse um desempenho muito melhor que a mente do homem comum. Com o início do desenvolvimento dos computadores na década de 1950, esse desejo começou a se tornar mais factível, já que com essa tecnologia seria possível testar algumas teorias. O termo inteligência artificial começou a ser utilizado em 1956 (STUART *et al*, 1995), mas antes disso já existiam pesquisas nessa área.

O primeiro a articular uma visão completa da inteligência artificial, mesmo ainda não utilizando esse termo, foi Alan Turing em seu artigo "*Computing Machinery and Intelligency*". Nesse artigo foi proposto o que hoje é conhecido como teste de Turing (RUSSEL, 2004). Esse teste consiste em realizar algumas perguntas por escrito para um computador e, se quem fizer as perguntas não souber distinguir se as respostas vieram de

uma máquina ou de um ser humano, o programa pode ser considerado como uma inteligência artificial.

Existem 4 linhas de pensamento para definir a inteligência artificial. A primeira delas diz que são sistemas que pensam como seres humanos, a segunda diz que são sistemas que atuam como seres humanos, a terceira fala em sistemas que pensam racionalmente e a última em sistemas que atuam racionalmente (STUART *et al*, 1995). A primeira e a terceira linha de pensamento referem-se ao processo de pensamento e raciocínio, já a segunda e a quarta ao comportamento. Além disso, as duas primeiras medem a capacidade de imitar o ser humano, já as duas últimas usam o conceito ideal de racionalidade.

Hoje em dia, existem muitas áreas em que a inteligência artificial é aplicada, sendo o objetivo final criar um sistema computacional que consiga simular perfeitamente a mente humana, por isso é aplicada em outras áreas, como jogos, robótica, programas de diagnóstico médico, resolução de equações matemáticas, reconhecimento de som e imagem e até os corretores ortográficos dos celulares.

2.2 SISTEMA ESPECIALISTA

Um dos ramos da inteligência artificial é conhecido como sistema especialista. A ideia desse tipo de sistema é simular o trabalho de um especialista em uma determinada área, armazenando todo o conhecimento dele no sistema e que servirá para o usuário encontrar respostas sobre assuntos em que o mesmo pode não ter total domínio.

A história do sistema especialista começa na década de 1960 na *Stanford Heuristics Programming Project*, onde o desejo era criar um sistema usando inteligência artificial que fosse capaz de lidar com problemas complexos, como no ramo médico no diagnóstico de doenças. O professor Edward Feigenbaum, da Universidade de Stanford, é considerado o precursor dessa tecnologia e a define como “um programa de computador inteligente que utiliza conhecimentos e procedimentos de inferência para resolver problemas que são difíceis o suficiente para requerer significativa perícia humana para solução”. Ou seja, o sistema especialista serve para emular a capacidade de decisão de um especialista humano em determinada área.

Na década de 1980 os sistemas especialistas se tornaram comerciais e acabaram se popularizando. Alguns deles são consagrados hoje em dia, como o PROSPECTOR, sistema aplicado em prospecção de minério que descobriu um depósito de molibdênio com valor de 100 milhões de dólares; o R1, sistema que selecionava componentes eletrônicos para sistemas de computador VAX de acordo com o pedido do cliente; o CADUCEUS, sistema usado para diagnosticar doenças humanas e o PUFF, que é um sistema de diagnóstico e consultas de doenças pulmonares.

A principal característica que diferencia o sistema especialista das outras técnicas de inteligência artificial é a independência entre seus módulos internos, que são a base de conhecimento, onde todas as informações dadas pelo especialista são armazenadas, e o motor de inferência, que utiliza os fatos dados pelo usuário, consulta a base de conhecimento e devolve a perícia. Essa independência acaba sendo uma grande vantagem, visto que é possível alterar a base de fatos e/ ou a base de conhecimentos sem precisar alterar o motor de inferência. A estrutura do sistema especialista é representada na figura 2.

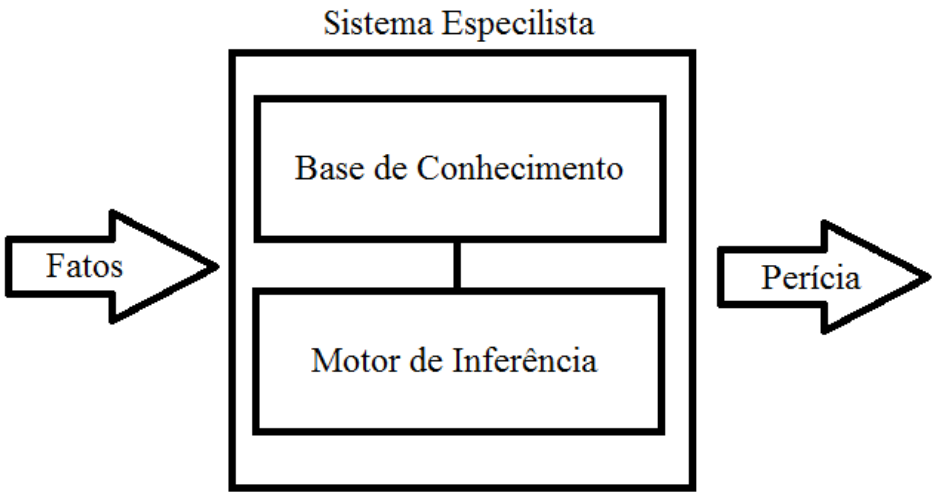

Figura 2- Estrutura de um Sistema Especialista

Uma boa representação do conhecimento deve ser capaz de dominar todas as questões existentes no problema. Para tal, existem diversas maneiras de se representar o conhecimento do especialista, sendo uma das mais usadas as regras, e também as árvores

lógicas, que serão melhor apresentadas ao longo deste trabalho. Além delas, existe a representação por *frames*, orientação a objetos e redes semânticas, entre outras.

O motor de inferência funciona recebendo os fatos e tentando encontrar informações na base de conhecimento para que tais fatos possam ser interpretados e a perícia encontrada. Existem duas maneiras de o motor de inferência funcionar, que são seus encadeamentos, sendo eles:

- Encadeamento para frente
- Encadeamentos para trás

No encadeamento para frente, o processo de inferência leva em conta os fatos iniciais colocados pelo operador e trabalha para obter uma perícia consultando a base de conhecimento. Já no encadeamento para trás o motor de inferência começa com a escolha de uma solução e realiza o processo de busca para trás para que evidências que comprovem essa solução sejam encontradas. Em outras palavras, o encadeamento para frente é guiado pelos fatos e o para trás é guiado pelo objetivo.

2.3 REPRESENTAÇÃO DO CONHECIMENTO

Cada tipo de problema tem uma maneira de representar o conhecimento que o consegue definir nos mínimos detalhes. Quanto mais precisa for a representação, melhor será a perícia que o sistema computacional fornecerá. A seguir serão apresentadas as formas mais utilizadas de se representar o conhecimento.

2.3.1 REPRESENTAÇÃO POR REGRAS

Representar o conhecimento usando regras é o meio mais natural que o especialista tem para expor suas ideias. As regras conseguem definir de forma adequada tudo que o motor de inferência precisa para poder fazer a perícia. Elas têm os seguintes formatos:

SE (antecedente) **ENTÃO** (consequente)

SE (antecedente) **ENTÃO** (consequente$_1$) **SENÃO** (consequente$_2$)

É importante ressaltar que essa regra tem sentido único, o que faz o inverso não necessariamente ser válido, ou seja, SE (consequente) ENTÃO (antecedente) não necessariamente é uma verdade, esse conceito é conhecido como *Modus Ponens*.

Para exemplificar essa teoria, será trabalhada a seguinte afirmação: "A praia fica cheia quando faz sol no final de semana". Essa afirmação, quando colocada na forma de regra, ficará da seguinte forma:

SE SOL **E** FINAL DE SEMANA **ENTÃO** PRAIA CHEIA

Com essa regra fica estabelecido que, sabendo o dia da semana e a condição do tempo, é possível estabelecer se a praia estará cheia ou não. Porém, não necessariamente se a praia estiver cheia quer dizer que é final de semana e está sol.

Pegando um exemplo mais inserido na área nuclear, é possível transformar em regra a seguinte afirmação: "Se a aplicabilidade for atendida e a contenção estiver inoperável, a CLO 3.6.1 precisa ser aberta". Na forma de regra ficaria:

SE APLICABILIDADE **E** CONTENÇÃO INOPERÁVEL **ENTÃO** CLO 3.6.1 ABERTA

Regras desse tipo também foram usadas no sistema computacional para controle das CLOs de Angra 1. A escolha de representação do conhecimento por regras para o sistema especialista dessa usina se mostrou muito eficiente, já que a mesma não possui uma árvore de falhas na forma de banco de dados para ser analisada.

2.3.2 REPRESENTAÇÃO POR ÁRVORES

Uma outra maneira de representar o conhecimento é através de árvores lógicas. Ela funciona atribuindo valores para algumas afirmações, como verdadeiro (*true*), falso (*false*), nulo (*none*) e inválido (*null*). Um exemplo é a afirmação "temperatura da água maior do que 50°C". Se a temperatura for maior que esse valor, será atribuído o valor verdadeiro, se for menor ou igual, será atribuído falso. Se a informação sobre o valor da temperatura não chegar ao programa, será atribuído o valor inválido, e se essa informação ainda não foi trabalhada pelo programa, será atribuído o valor nulo.

Essas afirmações são conectadas através de portões lógicos que farão com que se combinem e possam fazer sentido juntas umas das outras. Existem muitos tipos de portões lógicos e a figura 3 apresenta os principais deles, que aparecem na árvore de falha da

usina nuclear de Angra 2. Os ramos que se conectam com outro através de um portão lógico são chamados de filhos, e o ramo que os une, de pai.

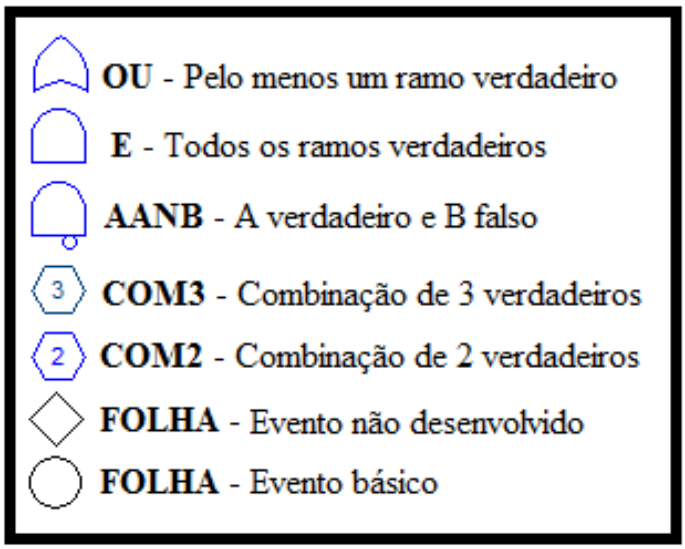

Figura 3 - Portões lógicos presentes na árvore de Angra 2

Cada portão funciona como se fosse uma regra, mas quando uma regra vai levando a outra, criando uma cadeia lógica entre elas, é mais conveniente o uso de árvores. A seguir, na figura 4, será apresentado um exemplo de árvore lógica que explica quais são os caminhos e o que é necessário acontecer para que a população consiga ser vacinada contra a febre amarela.

A árvore da figura 4 indica que para a população ficar informada sobre a importância de se vacinar, é preciso que a informação chegue por propaganda ou por conversas informais entre as pessoas. Além disso, para que todos sejam efetivamente vacinados, é preciso que estejam informados, decidam ir até o posto de saúde e que o governo tenha estoque de vacinas o suficiente para poder aplicar em todos que forem aos postos de saúde.

Percebe-se no exemplo que um portão lógico leva a outro, fazendo que a opção pelo uso de árvores em detrimento ao uso de regras foi acertada para esse exemplo.

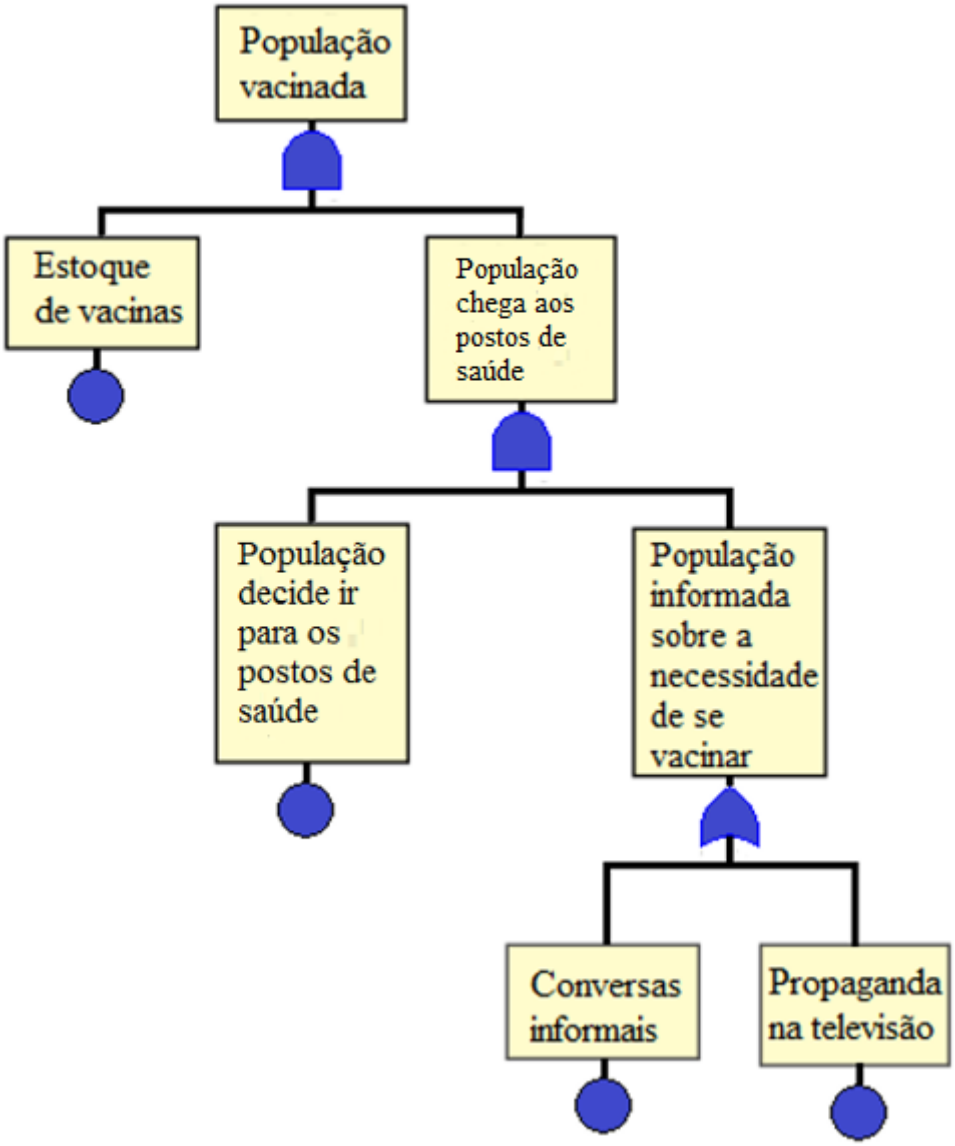

Figura 4 - Exemplo de árvore lógica

CAPÍTULO 3

APRESENTAÇÃO DO PROBLEMA

Para uma pessoa se tornar operador de uma usina nuclear, é preciso passar por um extenso curso de formação. Tudo que for aprendido durante esses anos será aplicado no dia a dia da sala de controle. É preciso entender uma infinidade de variáveis e situações que são praticamente inviáveis de serem aprendidas em sua totalidade e, além disso, um operador tem que lidar com a pressão de controlar uma usina nuclear e toda a sua periculosidade em caso de um acidente grave.

A solução para facilitar o trabalho dos operadores vem, não só dos procedimentos escritos, mas também dos sistemas computacionais que visam armazenar informações e apresentar caminhos para a resolução de problemas. Hoje em dia esses sistemas já estão completamente inseridos na sala de controle das usinas, seja para controlar variáveis ou apresentar soluções em caso de incidentes ou acidentes.

Um dos principais documentos em que o operador baseia suas decisões é o relatório de análise de segurança da usina. Ele é base para o capítulo 16 do RFAS (CNEN NE 1.04, 2002), que, além de outros assuntos, contém todo o conhecimento sobre as condições limites de operação da usina, que é o foco deste trabalho. Os itens a seguir irão apresentar esse documento e suas especificações.

3.1 RELATÓRIO DE ANÁLISE DE SEGURANÇA

A Comissão Nacional de Energia Nuclear (CNEN) é a agência reguladora das usinas nucleares brasileiras. Para que uma usina nuclear entre em operação, é necessário que a CNEN faça o seu licenciamento e isso só ocorre quando todos os documentos forem apresentados. Um dos principais documentos é o relatório de análise de segurança, onde fatores técnicos e humanos da planta são apresentados. Esse relatório tem duas etapas:

1. Relatório Preliminar de Análise de Segurança (RPAS)
2. Relatório Final de Análise de Segurança (RFAS)

O RPAS é feito antes de começar a construção da usina. Nele estão presentes dados importante como informações da construção, possível impacto ambiental gerado, local em que será construída, quanto pretende gerar de energia e tempo de operação até o descomicionamento. Com esse documento em mãos, a CNEN, fazendo o seu papel de órgão regulador, vai avaliar se concede o aval para liberar o início das obras.

Por sua vez, o RFAS apresenta características da operação da usina, como as bases de projeto, os limites de operação e uma análise de segurança completa. Esse relatório deve ser atualizado de acordo com o dia a dia da usina. A CNEN tomará posse desse documento e vai avaliar se a planta está apta a começar a funcionar, dando o aval para operação inicial e permanente da mesma (CNEN NE 1.04, 2002). Alguns dos capítulos do RFAS mais importantes para esse trabalho são o 15 (Análise de acidentes), de onde foi tirada a árvore de falha, o 16 (Especificações técnicas), que mostra as características da usina para conter acidentes e suas situações padrões de operação e o 18 (Engenharia de fatores humanos). Mais informações sobre esses dois relatórios de análise de segurança podem ser encontradas na norma NE 1.04 da CNEN nos itens 6.4 (RPAS) e 8.4 (RFAS).

3.2 CONDIÇÃO LIMITE DE OPERAÇÃO (CLO)

As duas usinas nucleares de Angra, atualmente em operação, apesar de ambas serem do tipo PWR, são muito diferentes uma da outra em termos operacionais. Uma das diferenças encontradas é na apresentação das suas respectivas CLOs. Será feita uma explicação sobre a CLO de Angra 1 e como seu sistema computacional foi feito e depois será apresentada a CLO de Angra 2, que é o foco do trabalho.

3.2.1 CLO ANGRA 1

A CLO de Angra 1 é apresentada na forma de textos e tabelas, sendo que atualmente seu formato foi atualizado para ficar similar ao de Angra 2. Sua aplicabilidade, que é o que diz se a CLO pode ser aberta ou não, vem apresentada na forma de texto e suas variáveis mais comuns são os modos de operação. A figura 5 apresenta todos os modos de operação existentes de acordo com as especificações técnicas feitas pela ELETRONUCLEAR em 2012.

MODO	TÍTULO	CONDIÇÃO DE REATIVIDADE (k_{eff})	% POTÊNCIA TÉRMICA NOMINAL[a]	TEMPERATURA MÉDIA DO REFRIGERANTE DO REATOR °C (°F)
1	Operação à Potência	> 0,99	> 2%	NA
2	Partida	> 0,984	≤ 2%	NA
3	Prontidão Quente	≤ 0,984	NA	≥ 177°C (350°F)
4	Desligado Quente[b]	≤ 0,984	NA	93°C (200°F) < T_{med} < 177°C (350°F)
5	Desligado Frio[b]	≤ 0,99	NA	≤ 93°C (200°F)
6	Recarrega-mento[c]	≤ 0,95	NA	≤ 60°C (140°F)

(a) Excluindo o calor de decaimento.

(b) Todos os parafusos de fechamento da cabeça do vaso do reator totalmente tensionados.

(c) Um ou mais parafusos de fechamento da cabeça do vaso do reator não totalmente tensionado(s), todas as barras inseridas e concentração de boro mantida dentro do limite especificado no Relatório de Projeto Nuclear e Termo-hidráulico RPNT.

Figura 5 - Características dos Modos de Operação

Após analisar se uma CLO está dentro de sua aplicabilidade, é preciso avaliar as suas condições. Se alguma condição for aplicável, deve-se começar a execução de suas ações requeridas. Cada condição pode requerer mais de uma ação e elas são ligadas por conectores lógicos. A figura 6 explica como se faz a relação entre os conectores de acordo com os exemplos explicativos presentes nas especificações técnicas da ELETRONUCLEAR (2012).

Para esse exemplo, o documento apresenta a seguinte explicação: “Este exemplo representa um uso mais complicado de conectores lógicos. As Ações Requeridas A.1, A.2 e A.3 são escolhas alternativas; somente uma delas tem que ser executada, como indicado pelo uso do conector lógico OU e por seu posicionamento em relação à margem esquerda. Qualquer uma destas três Ações pode ser escolhida. Se A.2 for escolhida, então ambas as ações A.2.1 e A.2.2 devem ser executadas como indicado pelo conector lógico E. A Ação Requerida A.2.2 é cumprida pela execução de A.2.2.1 ou A.2.2.2. A posição do conector

lógico OU mais afastado da margem esquerda indica que A.2.2.1 e A.2.2.2 são escolhas alternativas, portanto somente uma delas tem que ser executada. ”.

AÇÕES

CONDIÇÃO	AÇÃO REQUERIDA	TEMPO PARA CONCLUSÃO
A. CLO não cumprida.	A.1 Desarme . . . OU A.2.1 Verifique . . . E A.2.2.1 Reduza . . . OU A.2.2.2 Execute . . . OU A.3 Alinhe . . .	

Figura 6 - Relação Entre Conectores Lógicos nas Ações Requeridas

Na coluna “TEMPO PARA CONCLUSÃO” é apresentado o tempo que o operador terá para cumprir cada ação requerida na tentativa de solucionar o problema. Além disso, o texto da CLO também apresenta o sistema que ela faz parte, em qual parte do sistema ela atua e quais são as suas características. A figura 7 representa a CLO 16.3.6.4, que faz parte do sistema de contenção, onde é possível verificar todas as informações anteriormente apresentadas.

Na figura 7, é possível verificar que a CLO 16.3.6.4 diz respeito a pressão na contenção e, se o modo de operação da usina for do tipo 1, 2, 3 ou 4, a CLO será aplicável. Se a pressão na contenção estiver fora dos limites, entra-se na condição A e o tempo para o cumprimento de suas ações começará a ser contado. Se eventualmente a ação A.1 não for feita no tempo requerido (1 hora), a condição B é acionado e terá sua contagem de tempo iniciada. No caso de não conclusão de suas ações, a CLO será considerada como

violada. Para cada CLO aberta é exigido ao operador que faça um relatório com todas as ações realizadas para ser enviado à CNEN relatando o acontecido.

16.3.6 SISTEMAS DA CONTENÇÃO

16.3.6.4 Pressão da Contenção

CLO 16.3.6.4 A pressão da contenção deve estar $\geq$ -0,021 kg/cm^2 (-0.3 psig) e $\leq$ 0,105 kg/cm^2 (1,5 psig).

APLICABILIDADE: MODOS 1, 2, 3 e 4.

AÇÕES

CONDIÇÃO		AÇÃO REQUERIDA		TEMPO PARA CONCLUSÃO
A.	A pressão da contenção não está dentro dos limites.	A.1	Restaure a pressão da contenção para dentro dos limites.	1 hora
B.	A Ação Requerida e o Tempo para Conclusão associado não cumpridos.	B.1	Esteja em MODO 3.	6 horas
		E		
		B.2	Esteja em MODO 5.	36 horas

Figura 7 - Texto da CLO 16.3.6.4 do Sistema de Contenção

3.2.2 CLO ANGRA 2

As condições limites de operação da usina nuclear de Angra 2 se assemelham à maioria das usinas do mundo. Ela tem o texto similar ao de Angra 1, mas que se diferencia dessa usina por ter suas ações diretamente ligadas a árvore de falha retiradas do programa CAFTA, do grupo americano EPRI. Todas as CLOs de Angra 2 estão inseridas e interligadas por conectores lógicos.

A árvore de falha estudada nesse trabalho teve sua última atualização feita em 2010, por conta disso, alguns ramos podem estar diferentes daqueles usados atualmente, mas constam como um exemplo para o estudo de viabilidade no desenvolvimento do protótipo deste trabalho. Um exemplo são os ramos que dizem respeito aos modos de operação, que foram atualizados durante o desenvolvimento deste trabalho e atualmente são iguais aos usados na usina de Angra 1.

As árvores de falha apresentadas no CAFTA representam a CLO e todas as suas características. A aplicabilidade, condição, sistema, ação requerida e tempo para conclusão estão inseridos nela. A figura 8 demonstra como é representada a aplicabilidade de uma CLO nesse tipo de linguagem.

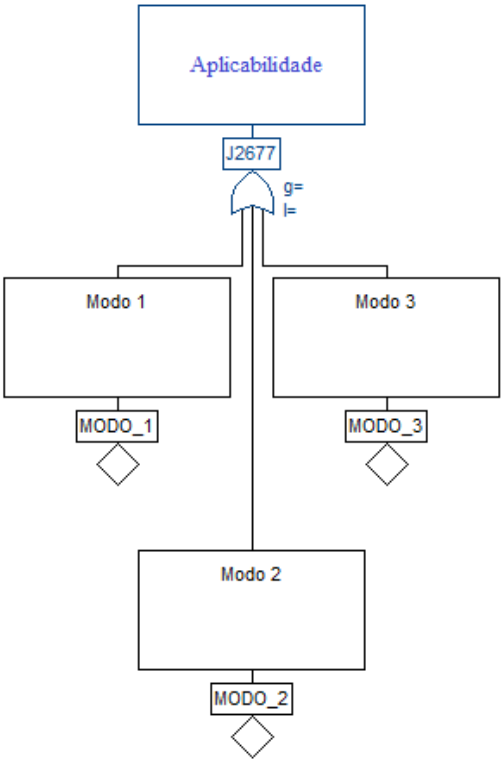

Figura 8 - Aplicabilidade de uma CLO demonstrada na linguagem do CAFTA

Como demonstrado na figura 3, o conector lógico, logo abaixo da aplicabilidade, é o "OU", isso quer dizer que se a usina estiver operando nos modos 1, 2 ou 3, o portão do modo em questão receberá o valor "verdadeiro", com isso o portão da aplicabilidade também receberá "verdadeiro", fazendo com que a CLO seja verdadeira e, consequentemente, considerada aplicável.

Além da aplicabilidade, outra parte importante a ser estudada é a entrada ou não em determinada condição. Levando como exemplo a CLO 16.3.6.4 de Angra 2, a sua condição "A" diz que a pressão interna da contenção está fora dos limites. Com isso, se a pressão realmente estiver fora dos limites estabelecidos, a condição "A" será aberta e dará

início ao contador da ação requerida referente a essa condição. A figura 9 mostra esse esquema apresentado na linguagem de árvores de falha do CAFTA.

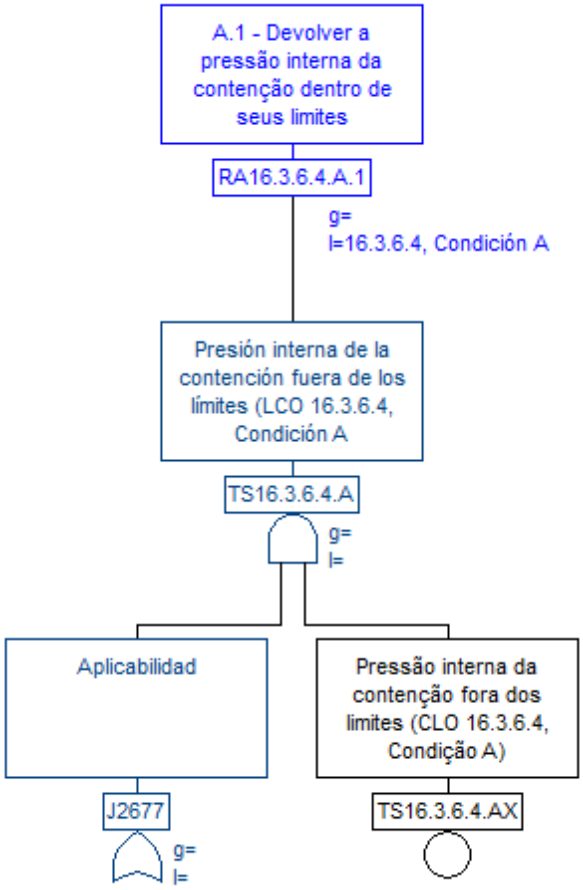

Figura 9 - Condição e Ação Requerida de uma CLO na linguagem do CAFTA

Na linguagem CAFTA, e de acordo com a figura 9, tem-se que se a aplicabilidade estiver com valor “verdadeiro” e a folha TS16.3.6.4.AX também for verdadeira, a condição “A” receberá “verdadeiro” e será aberta, começando a contagem do contador de tempo da ação requerida “A.1” no ramo RA16.3.6.4.A.1. É de maneira similar que todos as outras condições são expressas no CAFTA. Deste modo, todas as CLOs podem ser escritas nessa linguagem. Para que a leitura das árvores seja feita de maneira correta, é preciso começar a ler de baixo para cima, começando da causa até chegar ao efeito e às possíveis resoluções do problema.

3.3 CAFTA

O grupo americano EPRI desenvolveu o programa CAFTA para ser uma linguagem universal na elaboração de árvores lógicas. Atualmente, muitas usinas nucleares do mundo organizaram suas árvores de falha nesse programa, como a usina nuclear de Angra 2. Uma linguagem universal faz com que a interpretação do conteúdo seja facilitada e padronizada e o operador / engenheiro tenha menos dificuldades na hora de trabalhar e analisar os documentos.

Algumas informações não são apresentadas diretamente pelo programa, sendo necessário abrir a janela de propriedades de cada ramo para adquiri-las. A informação mais importante desse tipo é o tempo para conclusão de uma ação. A figura 10 mostra parte da tela do CAFTA (para a condição A da CLO 16.3.6.4) com a janela de propriedades aberta, aparecendo o tempo de conclusão (1 hora) para o ramo RA16.3.6.4.A.1. Esse tempo para conclusão da ação requerida está descrito como "*Action Ti*" na janela em destaque.

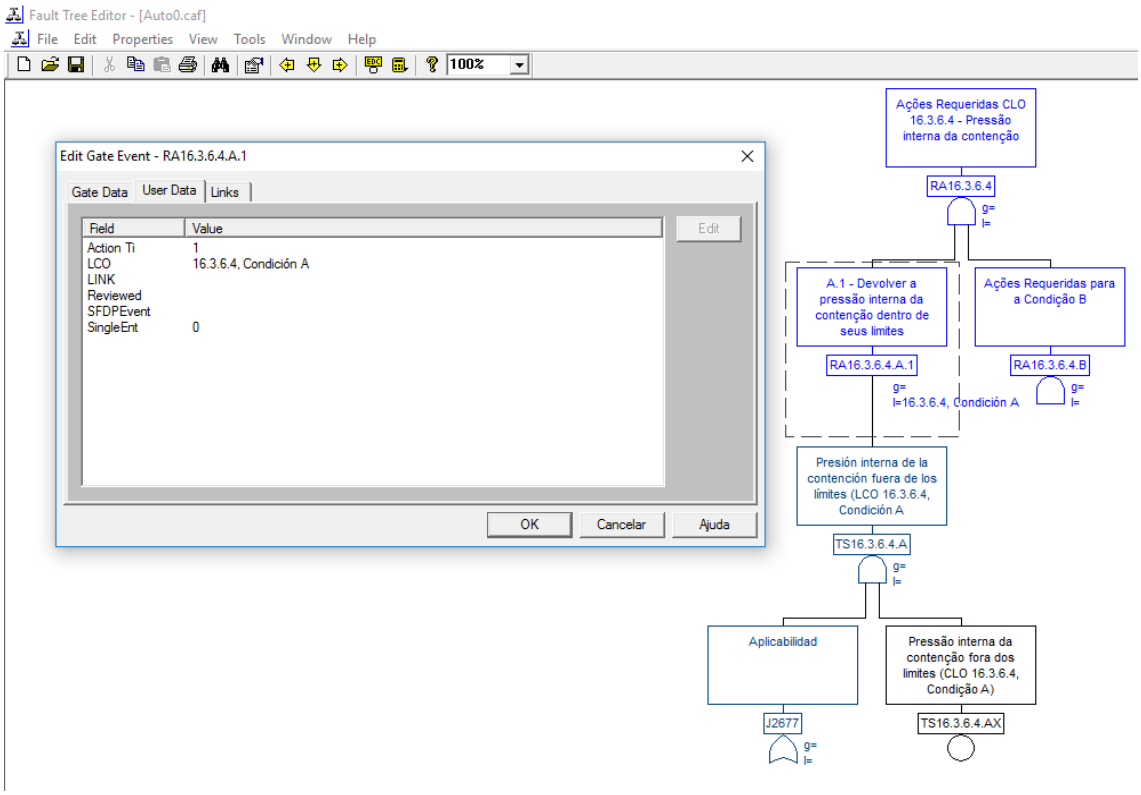

Figura 10 - Parte da tela do CAFTA mostrando o tempo de conclusão de um ramo

Apesar de não ser um programa para fazer o controle das árvores, o CAFTA permite colocar os valores “verdadeiro”, “falso” ou “nulo / inválido” nos ramos. Desta forma, é possível verificar a sequência lógica que eles seguirão. Esses valores são colocados com a lógica de cores, distribuída da seguinte forma:

- “Verdadeiro” → Vermelho
- “Falso” → Verde
- “Nulo” / “Inválido” → Branco

A figura 11 demonstra como um ramo de conector “OU” se comporta se cada um dos seus 3 filhos estiver com um valor diferente, sendo um “verdadeiro”, um “falso” e um “nulo” / “inválido”. Nela podemos observar que, por ser um operador “OU”, é necessário que apenas um dos seus filhos seja “verdadeiro” para que ele receba esse mesmo valor, fazendo com que os outros filhos não alterem o resultado.

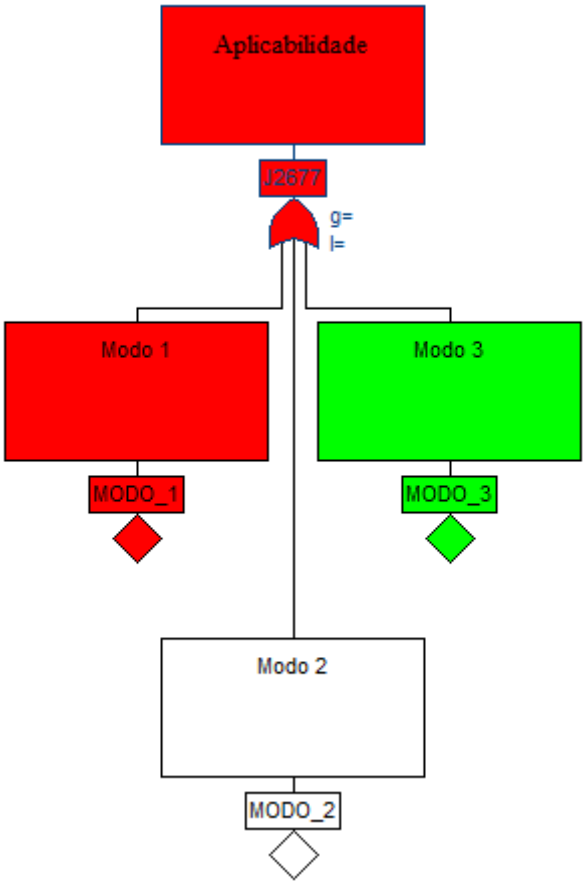

Figura 11 - Exemplo de uso do CAFTA para ver o resultado do portão lógico

O CAFTA possibilita que suas árvores sejam salvas em uma série de extensões diferentes. Uma delas é a "*Free-formated File* (*.Fre)", que retorna um arquivo, compatível com o bloco de notas do *Windows*, permitindo ao usuário saber o nome do ramo, seu operador lógico e seus filhos. Esse tipo de arquivo pode ser lido por outros programas que querem interpretar as árvores feitas na linguagem CAFTA. A figura 12 mostra uma parte da árvore de falhas da usina nuclear de Angra 2 salva nesse tipo de extensão.

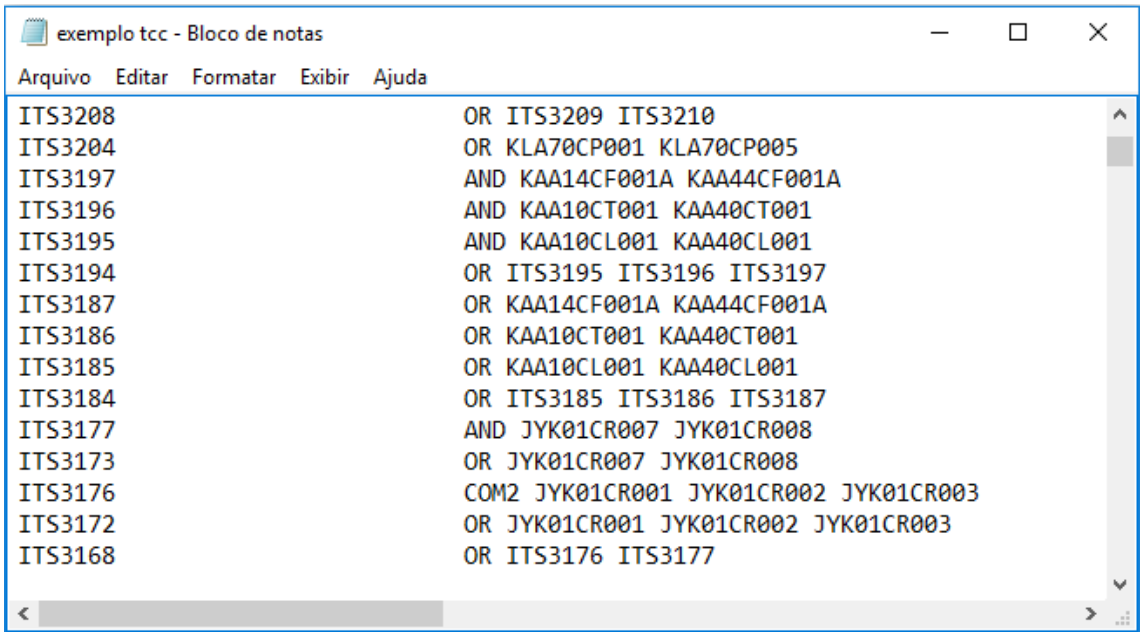

Figura 12 - Parte da árvore de falha de Angra 2 salva na extensão ".fre"

A usina nuclear de Angra 2 possui o programa AutoSpec que faz a leitura dos dados do CAFTA e realiza o controle das árvores. Esse programa não atende todas as necessidades das salas de operação das usinas em que está presente, por apresentar alguns problemas que dificultavam um bom uso pelo operador. A função desse trabalho é propor um sistema protótipo em Python que realize função similar ao que o AutoSpec se propõe. O AutoSpec é de propriedade da Areva e sua utilização é permitida apenas com licença de uso. Seu código fonte não está disponível para ser estudado, sendo tecnologia proprietária.

No caso específico de Angra 2, a principal função que os operadores necessitam, e o AutoSpec não oferece, é a ligação direta com o SICA. O SICA de Angra 2 é responsável pelo monitoramento em tempo real dos parâmetros essenciais para a determinação do estado de segurança da usina no caso de uma situação de emergência,

bem como no acompanhamento do funcionamento da mesma durante sua operação normal (NICOLAU, 2014). Atualmente a comunicação das informações entre esses dois programas é feita manualmente pelo operador. O sistema protótipo proposto neste trabalho tem como característica a comunicação direta com o SICA, o que o torna um possível candidato para substituir o AutoSpec na operação de Angra 2.

CAPÍTULO 4

METODOLOGIA E DEMONSTRAÇÃO DO PROTÓTIPO

A metodologia desenvolvida para o protótipo foi elaborada para atender todas as necessidades que um operador venha a ter, caso uma CLO precise ser aberta. Juntamente a isso, também foi elaborado para que possa, conforme necessário, ser modificado de maneira simplificada em situações em que a árvore de falha da usina sofra alguma alteração.

4.1 METODOLOGIA

Após um longo estudo sobre as CLOs de Angra 2 e sua adaptação à linguagem do CAFTA, foi percebido que a melhor maneira de se fazer um Sistema de Tempo Real (STR), para esta aplicação, era através de um sistema especialista. Com isso, um protótipo foi criado com o intuito de ser um sistema computacional genérico de controle de CLOs em tempo real, servindo para qualquer usina do mundo que tenha suas CLOs escritas na linguagem do CAFTA.

O sistema especialista proposto foi adaptado para que seu motor de inferência seja capaz de interpretar uma árvore de falha escrita, em arquivo de texto, fornecido pelo CAFTA. A leitura dessa árvore é feita de baixo para cima usando funções recursivas. Como o SE tem sua base de conhecimento completamente independente do motor de inferência, uma alteração na árvore de falha pode ser feita sem precisar alterar o corpo do programa.

Para uma precisa interpretação da árvore, o motor de inferência foi criado com a capacidade de interpretar todos os portões lógicos apresentados na figura 2, que são os presentes na CLO da usina nuclear de Angra 2. O correto entendimento e interpretação desses portões é que dão a precisão na perícia especialista do programa, já que o operador não precisará mais verificar manualmente os documentos de segurança toda vez que a usina apresentar alguma situação anormal, para detectar a abertura de uma CLO.

O controle das CLOs é feito em tempo real, mas nem tudo no sistema protótipo tem essa característica. Existem tarefas periódicas, como o controle de variáveis, e aperiódicas, como a estrutura da árvore. O ciclo escolhido nesse programa foi de 4 segundos, que é um tempo superior ao do processamento do programa, e considerado

razoável para que o valor de uma variável seja tratado como verdadeiro durante toda a duração do ciclo.

A cada ciclo, os dados são coletados de um arquivo de texto que simula o SICA de Angra 2. Caso um contador seja iniciado, o sistema protótipo fornecerá, a cada ciclo, a contagem regressiva para o fim do tempo de conclusão.

O protótipo ainda tem uma função que apresenta um resumo da situação atual da árvore, informando como está cada ramo e suas características. Esse controle é importante para saber se a lógica dos portões está funcionando e qual o ramo de maior profundidade que está com o valor de “verdadeiro”.

A junção de todos esses fatores, no programa escrito na linguagem Python, forma o sistema especialista protótipo proposto neste trabalho. Após a elaboração do mesmo, foi selecionada a CLO 16.3.6.4, da usina nuclear de Angra 2, para a realização dos testes para verificar a eficácia do sistema protótipo proposto. Foram testados fatores como os portões lógicos, folhas, tempo para conclusão e compatibilidade com o CAFTA. A CLO 16.3.6.4, na linguagem do CAFTA está representada nas figuras 13 e 14.

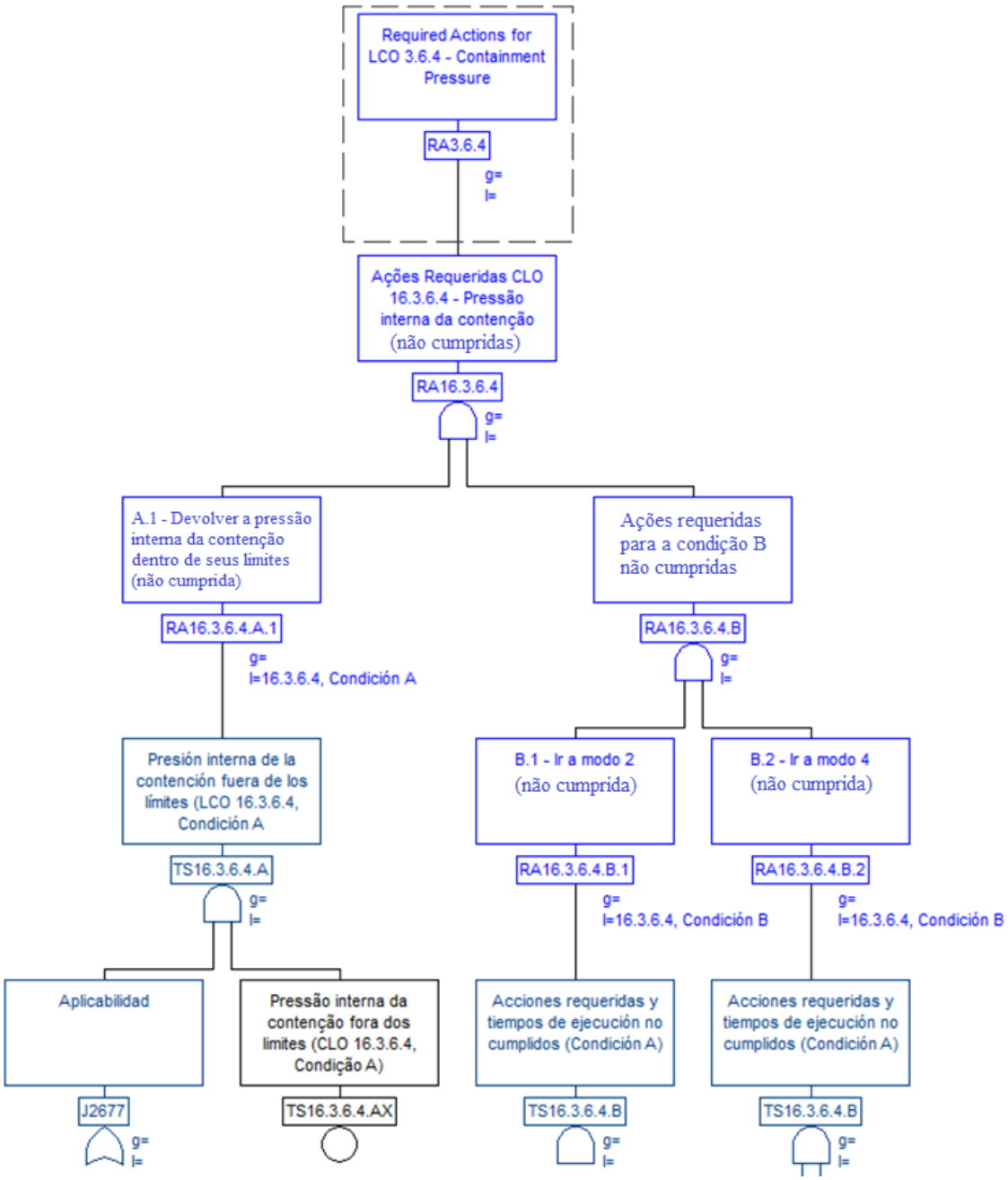

Figura 13 - CLO 16.3.6.4 na linguagem do CAFTA (primeira parte)

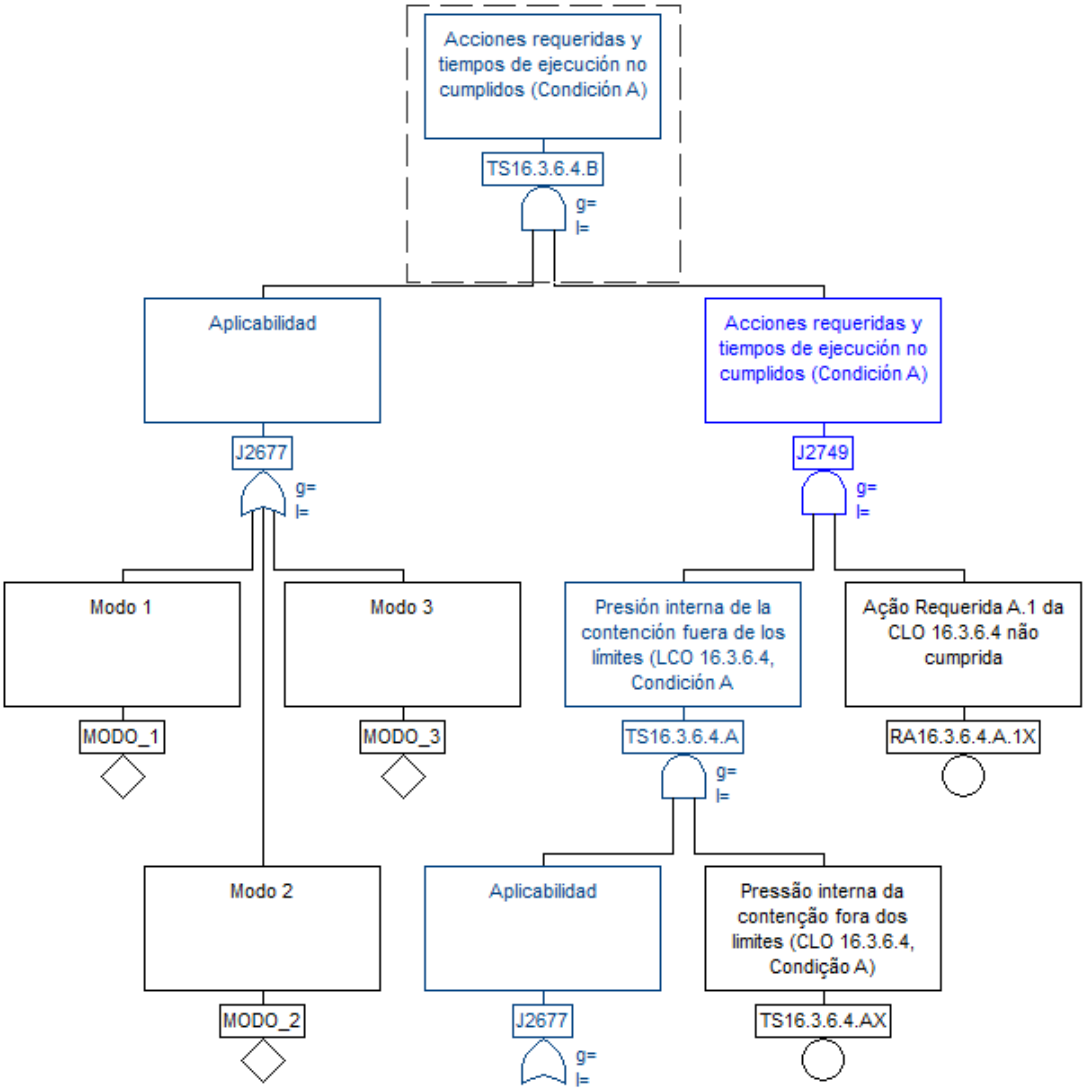

Figura 14 - CLO 16.3.6.4 na linguagem do CAFTA (segunda parte)

4.2 APRESENTAÇÃO DO PROTÓTIPO

O sistema protótipo proposto neste trabalho apresenta todas as características de um sistema especialista, com a independência entre o motor de inferência, a base de fatos e a base de regras. Neste capítulo todas as suas outras funções do sistema serão apresentadas.

4.2.1 MOTOR DE INFERÊNCIA

Como apresentado anteriormente, o motor de inferência pode fazer dois tipos de encadeamento, o para frente e o para trás. Como a lógica do sistema protótipo exige que

os fatos estejam inseridos para depois o mesmo fazer a perícia especialista, o encadeamento usado foi o para frente.

O motor de inferência do protótipo será alimentado com uma adaptação do arquivo de texto retirado do CAFTA, apresentado na figura 11. Essa adaptação foi feita de forma automática por uma rotina criada no Python, que faz parte do sistema protótipo, para se adequar a leitura do programa. Além disso, nesse mesmo arquivo de texto, foram inseridos os tempos para conclusão das ações requeridas. Eles estão assinalados entre colchetes logo depois da descrição de seus ramos. O modelo de como as árvores são inseridas está representado na figura 15. A CLO selecionada para testes foi a 16.3.6.4, que diz respeito à pressão na contenção.

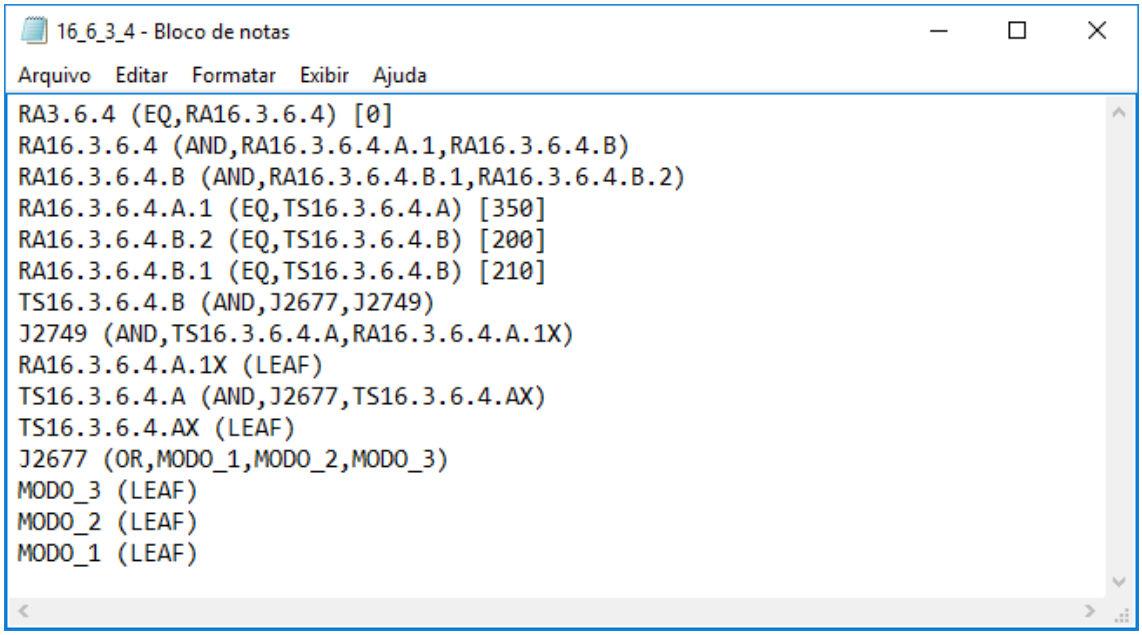

Figura 15 - Arquivo com a representação de um CLO na linguagem do protótipo

4.2.2 BASE DE CONHECIMENTO

O arquivo (16_6_3_4.txt) que será lido pelo motor de inferência fornece os dados necessários para interpretar corretamente a árvore de falha, sendo eles o: nome do ramo, seus filhos, seu operador lógico e o tempo para conclusão da ação. A lógica desguida em cada linha é dada por:

RAMO PAI (OPERADOR, FILHO 1, FILHO 2) [TEMPO PARA CONCLUSÃO]

Na lógica descrita anteriormente, foram apresentados 2 ramos filhos, porém isso não é uma regra, já que o ramo pode ter quantos filhos forem necessários e no caso de ser uma folha da árvore, ele não terá filho algum. Além disso, é importante ressaltar que o tempo para conclusão está em segundos, mas pode ser facilmente alterado para minutos ou horas de acordo com a especificação da CLO. Se um ramo não diz respeito à uma ação requerida, ele não terá tempo associado, ou seja, não será necessário apresentar o valor de tempo entre colchetes.

A leitura da árvore é feita com funções recursivas que percorrem a árvore de baixo para cima. Essa recursividade é importante para garantir que todos os ramos sejam lidos e avaliados de maneira correta. A leitura no sistema protótipo ocorre uma vez por ciclo para garantir que qualquer alteração feita no valor de um ramo seja interpretada e que a alteração seja feita em outros ramos, caso necessário.

4.2.3 BASE DE FATOS

A base de fatos do sistema protótipo faz uma simulação dos dados vindos do SICA de Angra 2. Como o controle precisa ser em tempo real, a cada ciclo que o sistema protótipo executar, os valores de todas as variáveis serão relidos e reenviados ao motor de inferência. No protótipo, existe um arquivo de texto que contém uma lista de todas as variáveis que podem receber valores, que são as folhas e as referentes ações requeridas, e ao lado estão seus respectivos valores, que podem ser verdadeiro (*TRUE*), falso (*FALSE*), nulo (sem resposta) ou inválido (*NULL*). Nos dois últimos casos o operador será informado e poderá mudar o valor da variável para condizer com a realidade da operação da usina no momento. A figura 16 representa esse arquivo de texto referente a CLO 16.3.6.4.

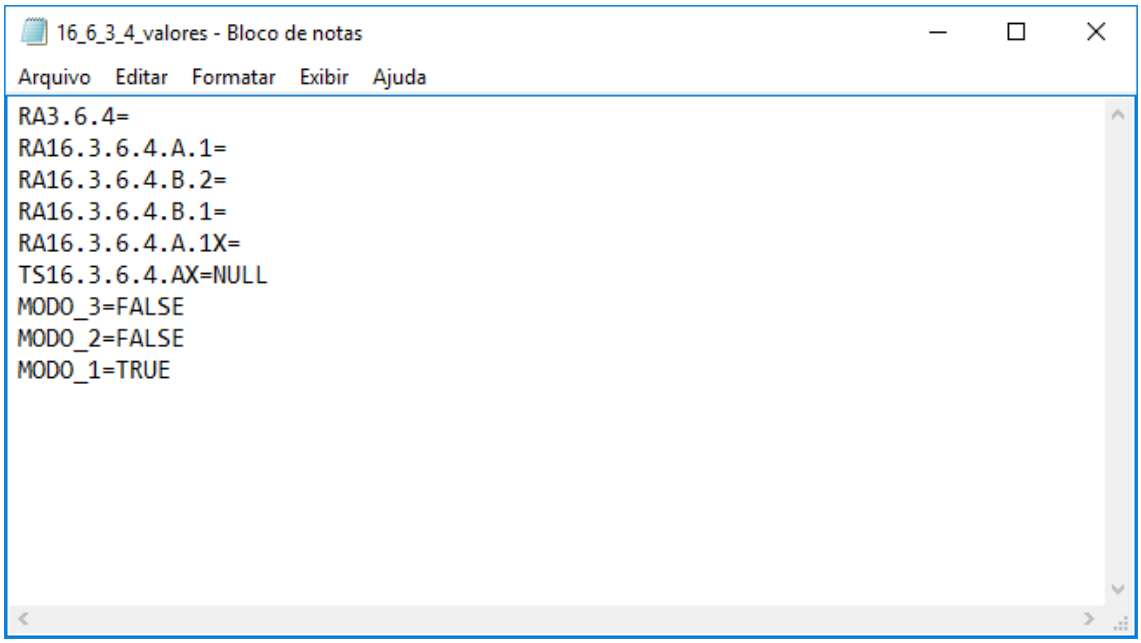

Figura 16 - Arquivo com a representação do valor das variáveis

Na figura 16 é possível avaliar que a usina está operando no modo 1 (MODO_1 = TRUE) e que outras variáveis estão sem valor. O que representa que as mesmas estão com o valor “nulo”. Esse valor é utilizado quando um ramo ainda não tem resposta. O ramo TS16.3.6.4.AX está com o valor “NULL”, que é considerado inválido. Isso ocorre quando o valor dessa variável deveria vir de alguma fonte externa, como o SICA, mas por algum motivo acabou não sendo adquirida pelo sistema. Em casos como esse o operador será informado do ocorrido e terá que informar o valor do ramo manualmente.

4.3 ATRIBUTOS DOS RAMOS

Visando uma boa organização do programa, foi criada uma classe que define cada ramo em diversos atributos, como código, texto, objeto, filhos, valor, operador, profundidade e tempo para conclusão. Esses atributos são atualizados a cada ciclo e assim que o operador necessitar, pode solicitar, pela interface do sistema, um texto que apresenta todas as características de cada ramo. Parte desse texto referente a CLO 16.3.6.4 está representado na figura 17.

O conceito de profundidade (*depth*), apresentado no parágrafo anterior, diz respeito à maior distância que um ramo precisa descer para chegar a uma folha. O texto

com o resumo de cada ramo está dividido por profundidades para facilitar a localização de parte da árvore que o operador necessitar.

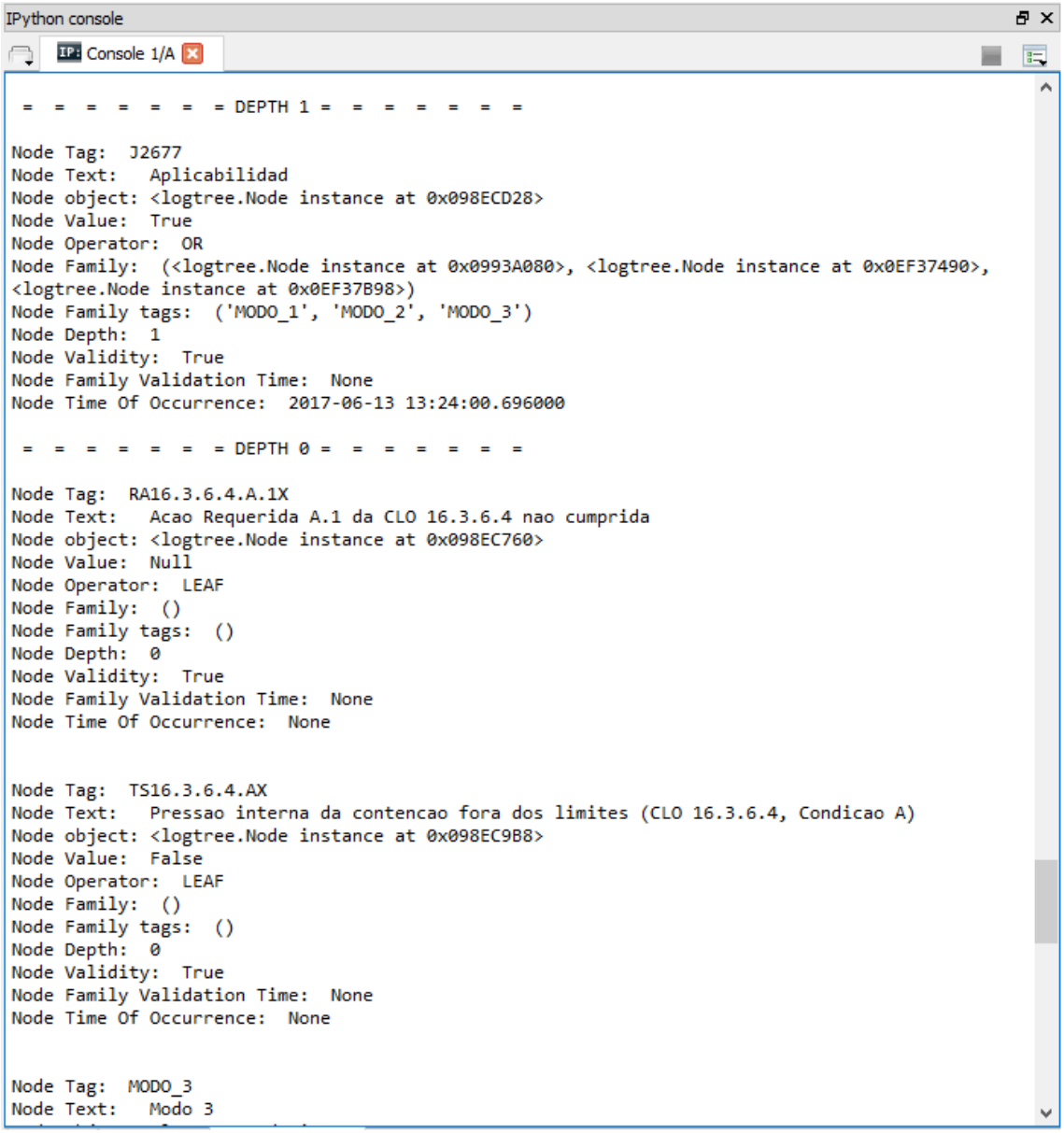

Figura 17 - Parte do texto com os atributos de cada ramo

Todos os atributos apresentados na figura 17 têm uma função importante para a elaboração do código e para a operação. A criação dessa classe permitirá ao operador obter todas as informações de forma sintetizada e direta, e a facilidade de fazer pesquisas pontuais sobre determinado ramo. Por exemplo, é permitido ao operador, durante a operação do sistema, pesquisar o tempo total que uma determinada ação tem para ser

executada, isso poderá auxiliar e facilitar a tomada de decisão dos operadores sobre como resolver a situação de maneira eficaz.

CAPÍTULO 5

PROTÓTIPO E RESULTADOS

Todo sistema protótipo precisa passar por uma etapa fundamental que é a de testes. Sem ela, seus resultados não poderão ser considerados válidos. O sistema protótipo desenvolvido neste trabalho foi testado usando CLOs adaptadas da usina nuclear de Angra 2. No apêndice encontra-se a parte principal do código fonte, em Python, desenvolvida neste trabalho. Foram testados todos os tipos de portões lógicos e de valores para cada ramo, além de verificar se todos os avisos que o operador precisa receber, como no caso de um ramo com o valor "inválido", estão aparecendo de maneira correta no programa.

O sistema protótipo divide sua operação por ciclos com tempo pré-determinado pelo operador. Em cada ciclo, será apresentada a data, a hora, o tempo de processamento e, se for o caso, o contador do algum ramo da árvore.

Neste capítulo, os principais testes realizados serão apresentados com o intuito de apresentar o funcionamento do protótipo. Além disso, será mostrada a interface que será apresentada ao operador com meio de interação com o mesmo. Observa-se que a interface desenvolvida neste trabalho tem por objetivo verificar a correta execução do programa.

5.1 FUNCIONAMENTO DO PROTÓTIPO

Para testar a eficácia do protótipo, como citado anteriormente, a CLO 16.3.6.4 de Angra 2 foi inserida na base de dados do sistema especialista proposto, conforme apresentado na figura 15. Essa CLO é considerada como um bom caso de teste, pois contém 3 ações com tempo para conclusão e 3 tipos diferentes de conectores lógicos. A árvore de falha dessa CLO, adaptada da linguagem do CAFTA, está representada nas figuras 13 e 14.

No teste em questão, a usina estará operando no modo 1 (MODO_1 = verdadeiro; MODO_2 = falso e MODO_3 = falso), fazendo com que a aplicabilidade da CLO seja verdadeira, e apresentando a pressão interna fora dos limites (ramo TS16.3.6.4.AX = verdadeiro). Todos os outros ramos estão com o valor nulo. Essa configuração abre a CLO na condição A e, então, seu contador começa a ser contado, como é possível

verificar na figura 18. Além disso, pode-se verificar nesta figura que o tempo de processamento de cada ciclo é calculado e apresentado na tela.

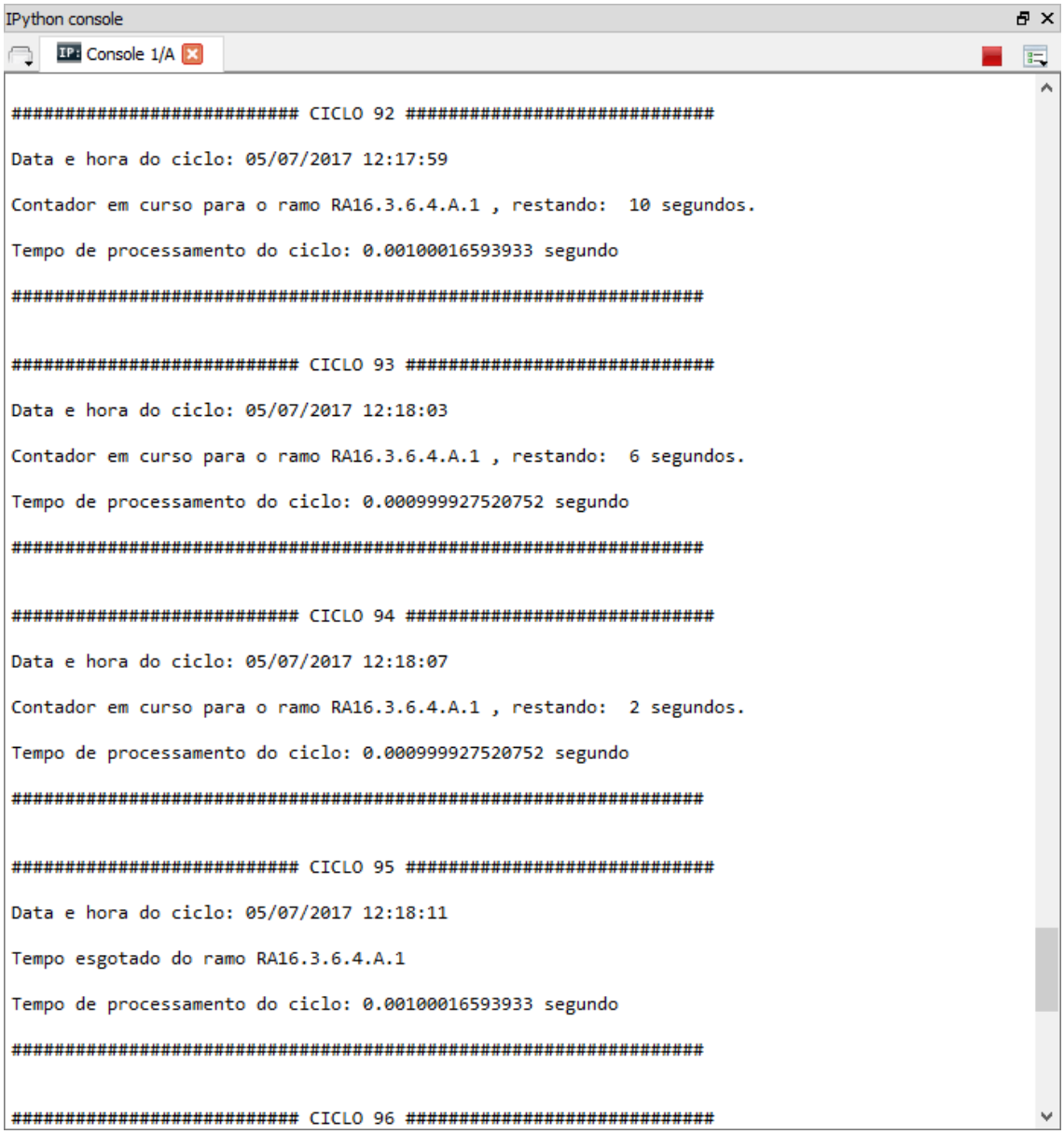

Figura 18 - Demonstração do contador da condição A da CLO 16.3.6.4

Se a ação não for cumprida dentro do prazo determinado, o sistema protótipo assumirá, automaticamente, o valor verdadeiro para o ramo. Isso se faz devido a lógica do protótipo de sempre atribuir o valor verdadeiro para a situação anormal da operação e o falso para o bom funcionamento da usina. Se a ação for concluída dentro do prazo, o operador deverá atribuir o valor falso para o ramo, fazendo com que a CLO receba o status de resolvida.

Com a ação da condição A não cumprida, o valor do ramo RA16.3.6.4.1X precisará ser alterado, pelo operador, para verdadeiro. Esse fato fará com que a condição B seja aberta e seus contadores iniciados, como pode ser visto na figura 19.

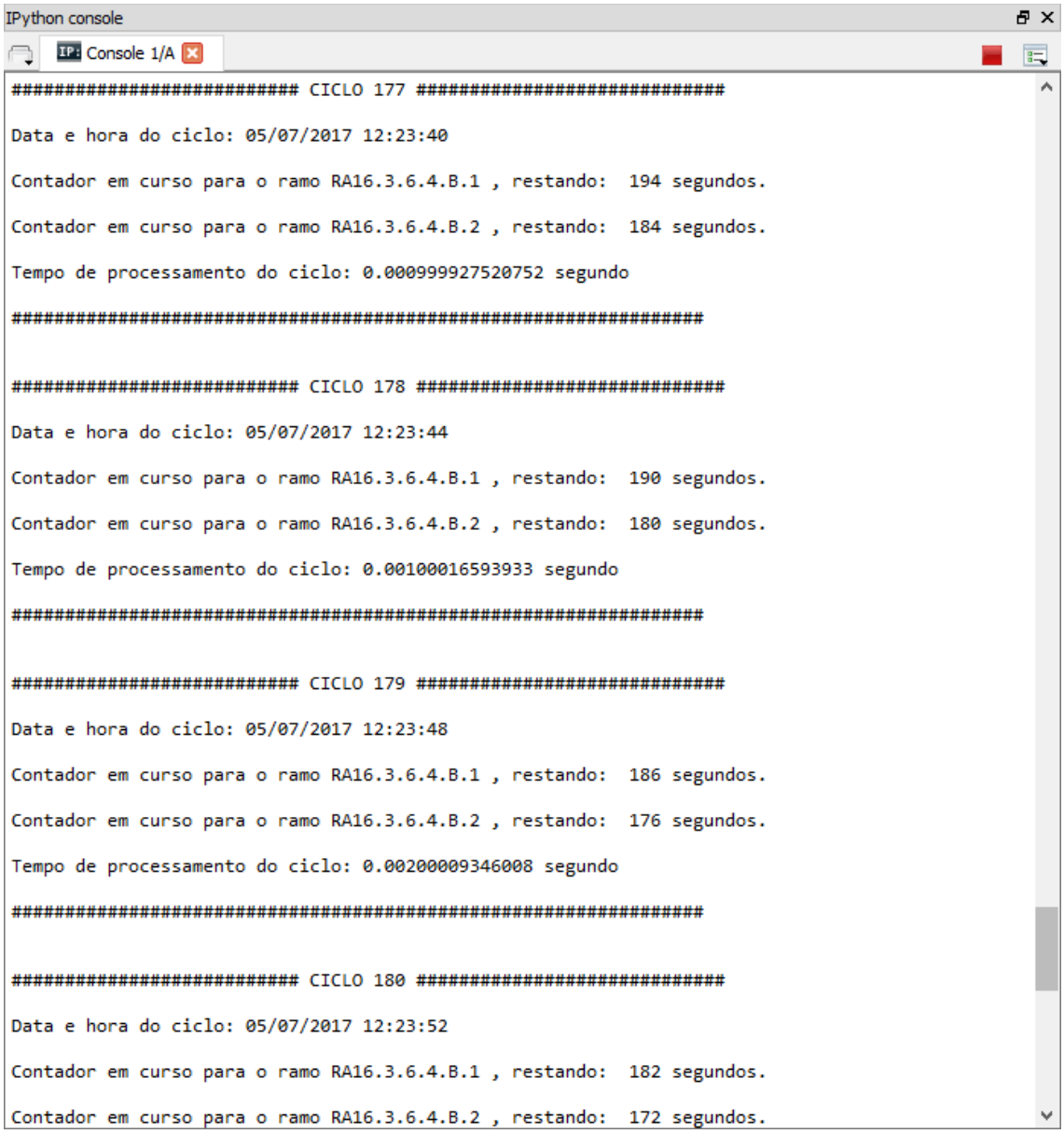

Figura 19 - Demonstração dos contadores da condição B da CLO 16.3.6.4

Se as duas ações (RA16.3.6.4.B.1 e RA16.3.6.4B.2) forem cumpridas, será atribuído o valor falso para ambos os ramos. Deste modo, o ramo do topo da CLO receberá o mesmo valor falso, o que significa que a CLO foi aberta, porém suas ações foram resolvidas e ela não foi violada. Outra possibilidade é que pelos menos uma ação não seja cumprida, o que dará o valor verdadeiro para a condição B e esse mesmo valor será atribuído ao topo da CLO. Esse caso significa que a CLO está aberta e foi violada.

Outro caso de relevância é quando um ramo assume o valor “inválido”. Isso pode acontecer quando a informação deveria vir de uma fonte externa e, por alguma anormalidade, não foi recebida pelo sistema. Na linguagem do protótipo será atribuído o valor “Null” ao ramo TS16.3.6.4.AX. A figura 20 demonstra, no ciclo 254, que o sistema protótipo envia um aviso ao operador (‘O ramo TS16.3.6.4.Ax esta com o valor NULL, inserir valor atual!’), informando-o que existe um ramo com valor “Null” e sugerindo que ele atribua, manualmente, um outro valor para esse ramo. Essa atribuição será feita no arquivo de texto apresentado na figura 16.

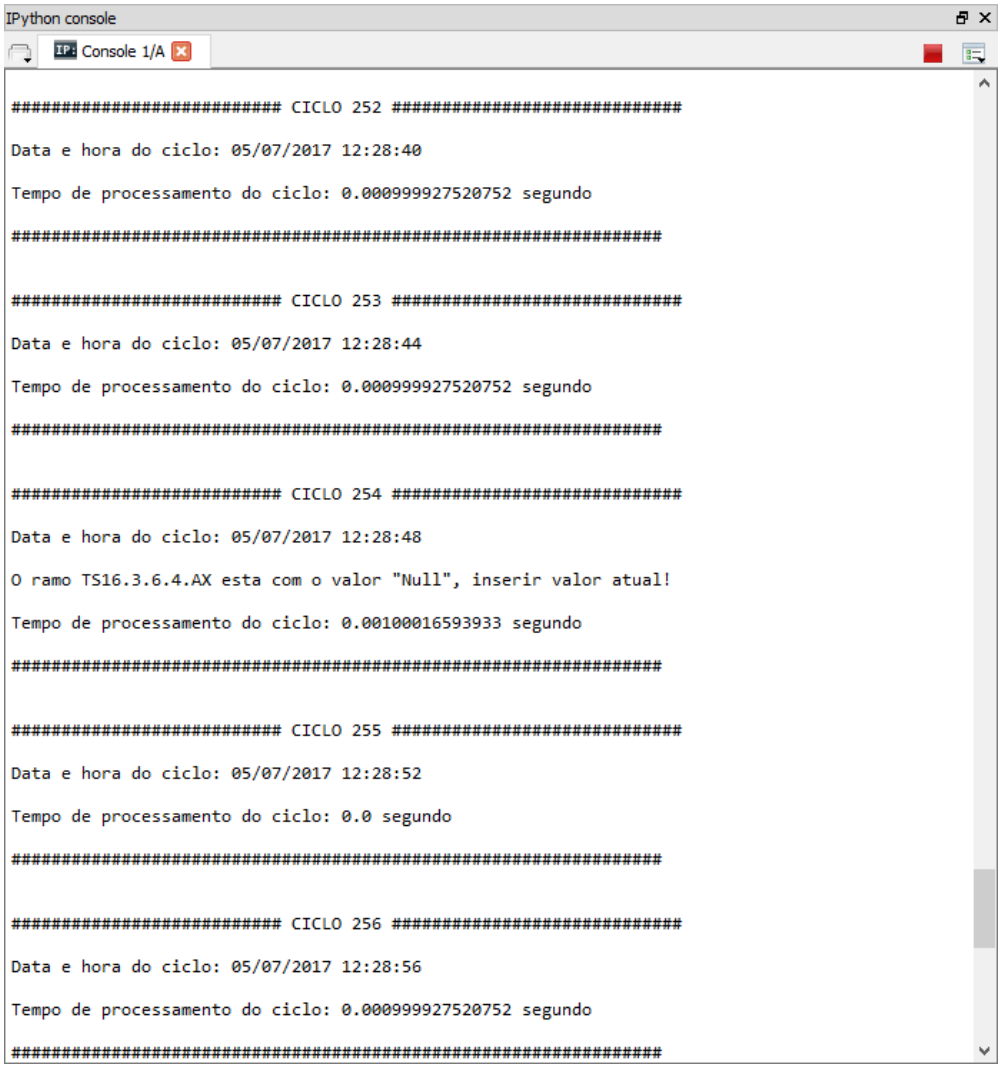

Figura 20 - Demonstração do aviso de ramo inválido

Um evento que leve à perda de contato do sistema protótipo com o SICA é bastante improvável, visto que o SICA funciona por duas linhas, se uma cair a outra continuará funcionando. Neste trabalho não foi feito nenhum estudo em relação a esse

fato, mas entende-se que o sistema protótipo precisa estar preparado para esse tipo de anormalidade, já que, se tratando da área nuclear, os requisitos de segurança são altamente rigorosos.

5.2 RESULTADOS

O objetivo deste trabalho era desenvolver um protótipo que facilitasse o controle das condições limites de operação da usina nuclear de Angra 2, adotando como fonte da base de conhecimentos as árvores de falha vindas do CAFTA. Todas as funções do sistema protótipo foram elaboradas com objetivo de auxiliar o operador no dia a dia da operação visando aumentar a segurança da usina e, consequentemente, evitar ou mitigar a ocorrência de possíveis acidentes ou incidentes.

Após a realização dos testes, foi comprovado que o protótipo atendeu perfeitamente todos os requisitos propostos. Onde a leitura da árvore foi feita de maneira correta e todos os avisos consideráveis suficientes para o suporte ao operador foram enviados de maneira precisa e no momento esperado.

Os testes foram elaborados com uma árvore adaptada, que é parte da árvore de falha original das CLOs de Angra 2, e podem ser considerados válidos para qualquer outro tipo ou tamanho de árvore. Deste modo, pode-se concluir que a árvore de falha geral da usina poderá ser facilmente lida e interpretada pelo sistema protótipo proposto.

CAPÍTULO 6

CONCLUSÃO E RECOMENDAÇÃO PARA TRABALHOS FUTUROS

O resultado do trabalho é o desenvolvimento de um sistema protótipo, que é um sistema especialista que trabalha em tempo real, que tem a função de auxiliar a tomada de decisão dos operadores de usinas nucleares do tipo PWR, com suas condições limites de operação escritas no formato de árvores de falha na linguagem do CAFTA. O SE de tempo real foi escolhido para estre trabalho por ser uma maneira eficiente de armazenar conhecimento e fornecer ao usuário no momento que for necessário. A linguagem de programação escolhida foi o Python por ser de auto nível e fornecida gratuitamente, além de ter facilidade de trabalhar com classes, listas e funções recursivas.

Após a realização dos testes e a comprovação da eficácia do sistema protótipo, é possível adaptar toda a árvore de Angra 2 para rodar no sistema protótipo sem mudar o bom desempenho do mesmo. Deste modo, o protótipo passará a atender todas as necessidades da operação de Angra 2, sendo esperado que todos os comportamentos apresentados nos testes sejam repetidos para a operação com a árvore completa da usina.

Para que o sistema protótipo proposto neste trabalho possa ser implementado em uma sala de controle de uma usina nuclear, é necessário elaborar uma interface gráfica mais amigável e que leve em conta os requisitos impostos pelo projeto de fatores humanos de usinas nucleares para salas de controle. O sistema proposto poderá ser muito útil para o controle das CLOs de Angra 2, melhorando o processo de tomada de decisões dos operadores e auxiliando na identificação de situações anormais, assim como o sistema computacional já elaborado para a usina de Angra 1 está sendo.

Como trabalhos futuros, após a elaboração da IHM, um modo de verificação e validação do sistema proposto seria a utilização do simulador real de Angra 2, usado para treinamento dos operadores da sala de controle, com objetivo de avaliar o comportamento do sistema visando sua implementação real na sala de controle.

CAPÍTULO 7

REFERÊNCIAS

ANGELI, C., **Diagnostic Expert Systems: From Expert's Knowledge to Real-Time Systems**, 2010. Disponível em: http://www.tmrfindia.org/eseries/ebookv1-c4.pdf. Acesso em: junho de 2017.

CAFTA Fault Tree Analysis, Electric Power Research Institute. Disponível em: http://www.epri.com/abstracts/Pages/ProductAbstract.aspx?ProductId=000000000001015514. Acesso em: junho de 2017

CNEN NE 1.04, **Licenciamento de Instalações Nucleares, Comissão Nacional de Energia Nuclear**, Brasil, 2002.

Especificações Técnicas, Central Nuclear Almirante Álvaro Alberto unidade 1, Eletrobrás Termonuclear S.A. – Eletronuclear, Rev.0, Rio de Janeiro, Brasil, 2012.

FARINES, J., FRAGA, J., OLIVEIRA, R., **Sistemas de Tempo Real**, Disponível em: http://www.das.ufsc.br/~romulo/livro-tr.pdf. Acesso em: julho de 2017.

International Atomic Energy Agency, Power Reactor Information System. https://www.iaea.org/pris/.

MACHADO, F. T. S., ***Desenvolvendo um Sistema Especialista baseado em regras para resolução de problemas na conexão de Internet no Software ExpertSinta,*** Disponível em: http://sites.setrem.com.br/stin/2012/anais/Fhabiana.pdf. Acesso em: julho de 2017.

Ministério de Minas e Energia, Balanço Energético Nacional, Relatório Final 2016. https://ben.epe.gov.br.

NUREG-0492, **Fault Tree Handbook**, US Nuclear Regulatory Commission, Washington, DC, 1981.

PASSOS, Emmanuel Lopes, 1989, **Inteligência Artificial e Sistemas Especialistas Ao Alcance de Todos**. Rio de Janeiro, Livros Técnicos e Científicos Editora LTDA.

Relatório Final de Análise de Segurança, Central Nuclear Almirante Álvaro Alberto unidade 2, Eletrobrás Termonuclear S.A. – Eletronuclear, Rev.: 10, Maio de 2007.

RUSSELL, S. e NORVIG, P., 1995, **Artificial Intelligence A Modern Approach.** New Jersey, Prentice Hall.

RUSSEL, S., NORVIG, P., **Inteligência Artificial**. 2ed, Elsevier, 2004

SALMON, D. R., ***Sistema Especialista Baseado em Níveis Progressivos de Diagnóstico para Identificação de Acidentes em Usina Nuclear PWR.*** Dissertação de Mestrado, Abril de 2013. Disponível em: http://antigo.nuclear.ufrj.br/MSc%20Dissertacoes /2013/dissertacao%20Douglas%20Ribeiro%20Salmon.pdf. Acesso em: junho de 2017.

VESELY, Bill, **Fault Tree Analysis (FTA): Concepts and Applications**. Disponível em: http://www.hq.nasa.gov/office/codeq/risk/docs/ftacourse.pdf. Acesso em: julho de 2017.

Apêndice A

Trecho do protótipo na linguagem de programação Python

```
import logtree as pt
import clo_expert
import time
import datetime
on_off = liga_desliga('C:\Users\Suporte\Desktop\liga_desliga.txt')
i=1
while on_off == True:
    comeco = time.time()
    print '############################ CICLO', i, '############################'
    print ''
    print 'Data e hora do ciclo:', datetime.datetime.now().strftime('%d/%m/%Y
%H:%M:%S:%f')
    wm = clo_expert.workmemory('C:/Users/Suporte/Desktop/16_6_3_4_valores.txt')
    pt.atualiza_valores_arvore(arvore,wm)
    pt.solve_from_bottom(wm,ciclo,arvore,True)
    i+=1
    fim = time.time()
    processamento = fim - comeco
    print ''
    if processamento < 1:
        print 'Tempo de processamento do ciclo:', processamento, "segundo"
    else:
        print 'Tempo de processamento do ciclo:', processamento, "segundos"
    print ''
    print '############################################################'
```

```
    print ''
    print ''
    on_off = liga_desliga('C:\Users\Suporte\Desktop\liga_desliga.txt')
    fim = time.time()
    processamento_final = fim - comeco
    time.sleep(ciclo-processamento_final)



def solve_from_bottom(wm,ciclo,TreeStructure, NoneOperation=False):
    TreeStructure = list(TreeStructure)
    TreeStructure.reverse()
    for eachNode in TreeStructure:
        familyValues = []
        for node in eachNode.family:
            if node.value == "Null":
                familyValues.append(None)
            elif node.validity == True:
                familyValues.append(node.value)
            elif node.validity == False and NoneOperation:
                familyValues.append(None)
            elif node.validity == False and NoneOperation == False:
                pass
        operator = eachNode.operator
        if eachNode.operator != 'LEAF':
```

```
            memoria_valor = eachNode.value
            Answer = solve_arc(wm,ciclo,eachNode,familyValues, operator,
ContainNone=NoneOperation)
            if NoneOperation == False and Answer == None:
                pass
            else:
                eachNode.value = Answer
            if (memoria_valor != eachNode.value) and (eachNode.value == True):
                eachNode.time_of_occurrence = dt.datetime.now()    #.strftime('%d/%m/%Y
%H:%M:%S')
    TreeStructure.reverse()



def atualiza_valores_arvore(TS, workmemory):
    keys = workmemory.keys()
    for eachNode in TS:
        memoria_valor = eachNode.value
        if eachNode.tag in keys:
            if eachNode.operator != 'EQ' or workmemory[eachNode.tag] != None:

                value = workmemory[eachNode.tag]
                eachNode.set_value(value)
        if memoria_valor != eachNode.value and eachNode.value == True:
```

```
            eachNode.time_of_occurrence = dt.datetime.now()     #.strftime('%d/%m/%Y
%H:%M:%S')
        if memoria_valor != eachNode.value and eachNode.value == "Null":
            print "
            print 'O ramo', eachNode.tag, 'esta com o valor "Null", inserir valor atual!'



def solve_arc(wm,ciclo,eachNode,familyValues, operator, ContainNone = False):
    if ContainNone:

        if operator == 'AANB':
            if familyValues[0]==True and familyValues[-1]==False:
                return True
            elif familyValues[0]==None and familyValues[-1]==False:
                return None
            elif familyValues[0]==True and familyValues[-1]==None:
                return None
            elif familyValues[0]==None and familyValues[-1]==None:
                return None
            else:
                return False

        if operator == 'COM2':
            conta_true = 0
            conta_false = 0
```

```
    for eachValue in familyValues:
        if eachValue == True:
            conta_true+=1
        if eachValue == False:
            conta_false+=1
    if conta_true >= 2:
        return True
    else:
        if conta_false>=(len(familyValues)-1):
            return False
        else:
            return None



if operator == 'COM3':
    conta_true = 0
    conta_false = 0
    for eachValue in familyValues:
        if eachValue == True:
            conta_true+=1
        if eachValue == False:
            conta_false+=1
    if conta_true >= 3:
        return True
```

```
    else:
        if conta_false>=(len(familyValues)-2):
            return False
        else:
            return None

if operator == 'AND':
    ValuesAuxiliar = []
    for eachValue in familyValues:
        if eachValue != False:
            ValuesAuxiliar.append(eachValue)
    ValuesAuxiliar = [False]*familyValues.count(False) + ValuesAuxiliar
    familyValues = ValuesAuxiliar
    expression = ""
    for Index in range(0,len(familyValues)-1):
        expression += str(familyValues[Index]) + " and "
    expression += str(familyValues[-1])
    Answer = eval(expression)
    return Answer

elif operator == 'OR':
    ValuesAuxiliar = []
    for eachValue in familyValues:
        if eachValue != None:
```

```
            ValuesAuxiliar.append(eachValue)
    ValuesAuxiliar = ValuesAuxiliar + [None]*familyValues.count(None)
    familyValues = ValuesAuxiliar
    expression = ""
    for Index in range(0,len(familyValues)-1):
        expression += str(familyValues[Index]) + " or "
    expression += str(familyValues[-1])
    Answer = eval(expression)
    return Answer


elif operator == 'NOT':
    if familyValues[0] == None:
        return None
    else:
        return (not familyValues[0])


elif operator == 'EQ':
    if wm[eachNode.tag] == True:
        return True
    elif wm[eachNode.tag] == False:
        return False
    else:
        return roda_contador (eachNode,ciclo)
    #return familyValues[0]
```

```
    else:
        raise Exception("Cant solve a "+str(operator)+" node")


else:
    if None not in familyValues and len(familyValues) > 0:
        if operator == 'AND':
            Answer = all(familyValues)
        elif operator == 'OR':
            Answer = any(familyValues)
        elif operator == 'NOT':
            Answer = not familyValues[0]
        elif operator == 'EQ':
            Answer = familyValues[0]
        elif operator == 'AANB':
            if familyValues[0]==True and familyValues[-1]==False:
                return True
            else:
                return False
        elif operator == 'COM2':
            conta_true = 0
            for eachValue in familyValues:
                if eachValue == True:
                    conta_true+=1
```

```
        if conta_true >= 2:
            return True
        else:
            return False
    elif operator == 'COM3':
        conta_true = 0
        for eachValue in familyValues:
            if eachValue == True:
                conta_true+=1
        if conta_true >= 3:
            return True
        else:
            return False
    elif operator == 'LEAF':
        raise Exception('Cant solve a LEAF')
    else:
        expression = ""
        for Index in range(0,len(familyValues)-1):
            expression += str(familyValues[Index]) + operator
        expression += str(familyValues[-1])
        Answer = eval(expression)
    return Answer


else:
```

```
return None
```