PROCESSAMENTO DE EVENTOS EM UM SISTEMA DE MONITORAÇÃO DE UM SISTEMA ELÉTRICO

Rafael Jorge Csura Szendrodi

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

Orientador: Jorge Lopes de Souza Leão

Rio de Janeiro
Setembro de 2013

PROCESSAMENTO DE EVENTOS EM UM SISTEMA DE MONITORAÇÃO DE UM SISTEMA ELÉTRICO

Rafael Jorge Csura Szendrodi

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

Examinada por:

_____

Prof. Jorge Lopes de Souza Leão, Dr. Ing. (orientador)

_____

Prof. Mauricio Aredes, Dr.-Ing.

_____

Prof. Geraldo Bonorino Xexéo, D. Sc.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2013

Szendrodi, Rafael Jorge Csura

Processamento de Eventos em um Sistema de Monitoração de um Sistema Elétrico / Rafael Jorge Csura Szendrodi. – Rio de Janeiro: UFRJ/COPPE, 2013.

XIV, 179 p.: il.; 29,7 cm.

Orientador: Jorge Lopes de Souza Leão

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia Elétrica, 2013.

Referências Bibliográficas: p. 77-79.

1. Processamento de Eventos Baseado em Conhecimento. 2. Processamento de Eventos Baseado em Modelo. 3. Cálculo de Eventos. 4. Reconhecimento de Crônicas. 5. *Frame Problem*. 6. ASP. 7. Proteção. 8. Sistema Elétrico. 9. Alarmes. I. Leão, Jorge Lopes de Souza. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

A Marcus Tulius Cicerus e aos deuses imortais.

Agradecimentos

Agradeço muito as pessoas que me ajudaram a avançar em mais uma etapa da minha vida, neste trabalho que despendeu muito esforço para ser concluído.

Agradeço aos meus pais, Ildiko Szendrodi e Gyorgy Ferenc Szendrodi, pelo apoio dado para o desenvolvimento do meu mestrado.

Agradeço de coração ao meu orientador, professor Jorge Lopes Souza Leão, pela ajuda inestimável que ele me deu, orientando-me de forma maravilhosa na conclusão desta dissertação e pela convivência proporcionada no dia a dia, durante esses quase dois anos, desde os primeiros rascunhos até a conclusão desta tese.

Agradeço ao meu colega de P&D, Alberto Wagner Collavizza, pela ajuda fornecida para o desenvolvimento e conclusão deste trabalho.

Agradeço ao professor Maurício Aredes, pela oportunidade de trabalhar no projeto de P&D da ANEEL, nº 001/11 que acabou gerando, como um de seus frutos, esta dissertação de mestrado.

Agradeço aos meus colegas do LEMT, pela ajuda nas horas de maior sufoco, em especial ao amigo Juliano Freitas Caldeira, Thiago Americano Brasil e Emanuel Leonardus van Emmerik.

Agradeço aos engenheiros Ricardo Teixeira Lima e Gabriella Gomide, da empresa ELECNOR Concessões Ltda., assim como ao engenheiro Flavio Luciano A. de Souza, da empresa Auctoritas, pelo suporte técnico dado durante o desenvolvimento do projeto de P&D.

Finalmente, agradeço de coração a ajuda da secretária acadêmica do Programa de Engenharia Elétrica, Daniele Cristina Oliveira da Silva que, sempre muito prestativa, ajudou-me nos procedimentos burocráticos para a continuidade e conclusão do meu mestrado.

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

PROCESSAMENTO DE EVENTOS EM UM SISTEMA DE MONITORAÇÃO DE UM SISTEMA ELÉTRICO

Rafael Jorge Csura Szendrodi

Setembro/2013

Orientador: Jorge Lopes de Souza Leão

Programa: Engenharia Elétrica

Atualmente, as empresas operadoras do sistema elétrico são remuneradas pelo tempo que conseguem manter as suas áreas de concessões funcionando sem que ocorram paradas no fornecimento de energia elétrica. Falhas que causem paradas no sistema elétrico nas áreas de concessões ocasionam a aplicação de multas pelo Operador Nacional do Sistema (ONS), na forma de descontos na remuneração paga às empresas operadoras do sistema elétrico.

Interessa então, para as empresas, terem sistemas que apontem as causas das paradas graves e inesperadas, de forma a corrigir os problemas o mais rápido possível e a diminuir o impacto das multas a serem aplicadas em suas remunerações mensais.

Neste trabalho, é estudado e implementado um método de analisar sequências de eventos geradas pelo sistema de proteção de um sistema elétrico. Este método vai auxiliar os operadores do sistema elétrico a descobrirem as sequências críticas que causaram um desligamento, de maneira que eles possam recompor o sistema o mais rapidamente possível.

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

EVENT PROCESSING IN AN ELECTRICAL SYSTEM MONITORING

Rafael Jorge Csura Szendrodi

September/2013

Advisors: Jorge Lopes de Souza Leão

Department: Electrical Engineering

Currently, the operating companies of the electrical system are paid for the time they keep their concession areas working without outages in electricity supply. Failures that cause outages in the electrical system in the areas of concessions cause fines by the National System Operator (*Operador Nacional do Sistema* - ONS), in the form of discounts on the remuneration paid to these companies.

It is thus interesting for the companies to have systems which point the causes of severe and unexpected outages in order to fix the problems as soon as possible and reduce the impact of the penalties to be applied in their monthly remunerations.

In this dissertation, a method for the analysis of event sequences, generated by the protection system of an electrical system, is studied and implemented. This method will aid the operators of the electrical system to find the critical sequences that caused the outage, so they can work on system restoration.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AI	*Artificial Intelligence*, p. 48
ANEEL	Agência Nacional de Energia Elétrica, p. 19
ASP	*Answer Set Programming*, p. 42-44, 49, 54, 57-68, 75, 76
CAE	Central de Alarmes e Eventos, p. 3, 4, 19, 20, 21, 57, 73, 74
CEPEL	Centro de Pesquisas de Energia Elétrica da ELETROBRAS, p. 7, 8
CLP	Controlador Lógico Programável, p. 14
CNOS	Centro Nacional de Operação do Sistema, p. 8
COS	Centro de Operação do Sistema, p. 8, 9, 11
COSR	Centro de Operação de Sistema Regional, p. 8
DNP3	*Distributed Network Protocol version 3*, p. 7
DROLLS	*Business Logic Integration Platform*, p. 49
ELETROBRAS	Centrais Elétricas Brasileiras S/A, p. 7
EC	*Event Calculus*, p. 2, 28-32, 36
EMS	*Electric Management Suite*, p. 8
GUI	*Graphical User Interface*, p. 8, 9
GPS	*Global Position System*, p. 11
HTTP	*HyperText Transfer Protocol*, p. 19
IEC	*International Electrotechnical Commission*, p. 7, 14
IEEE	*Institute of Electrical and Electronics Engineers*, p. 7
IHM	Interface Homem-Máquina, p. 14
JESS	*Java Expert System Shell*, p. 49
LISP	*LISt Processing*, p. 49

LSD (ver SDL), p. 43

LT Linhas de Transmissão, p. 6

OEC *Original Event Calculus*, p. 33, 36

ONS Operador Nacional do Sistema, p. 1, 10

OPS5 *Official Production System version 5*, p. 49

NFF Negação por Falha Finita, p. 43

P&D Pesquisa e Desenvolvimento, p. 19

POTASSCO *Potsdam Answer Set Solving Collection*, p. 44

PROLOG *Programmation en Logique*, p. 42,43, 49, 73

SAGE Sistema Aberto de Gerenciamento de Energia, p. 7-12, 14, 19, 20, 23, 51-54, 57, 69, 70, 73, 74, 76

SBC Sistemas Baseados no Conhecimento, p. 48-51, 53

SBM Sistemas Baseados em Modelos, p. 51-53

SCADA *Supervisory Control and Data Acquisition System*, p. 7-12, 14, 19, 20, 23, 51-54, 57, 69, 70, 73, 74, 76

SDL *Selective Linear Definite clause resolution*, p. 43

SDLNF *Selective Linear Definite clause resolution with Negation as Failure*, p. 43

SE Sistemas Especialistas, p. 48-50

SEC *Simplified Event Calculus*, p. 2, 36, 54, 57-68, 73, 75

SEE Sistemas de Energia Elétrica, p. 3-8, 13, 51, 52, 54

SEL *Scheweitzer Engineering Laboratories*, p. 6

TDD Transferência de Disparo Direto, p. 16, 17

TPAX Transformador de Potência Auxilias, p. 18

TRIP *Transfer-Trip* (Transferência de Disparo), p. 16, 17

WEB *World Wide Web*, p. 19

1. Introdução

1.1. Motivação

Atualmente, as empresas concessionárias do sistema elétrico são remuneradas pelo tempo que conseguem manter as suas áreas de concessões funcionando sem a ocorrência de paradas no fornecimento de energia elétrica. Falhas que causem paradas nos sistemas elétricos nas áreas de concessões ocasionam a aplicação de multas pelo Operador Nacional do Sistema (ONS), na forma de descontos na remuneração pagas às empresas concessionárias do sistema elétrico. Tais multas, denominadas de "*parcelas variáveis*", variam conforme o tempo de demora até a restituição do funcionamento do sistema elétrico na área de concessão e também conforme o tipo de evento que ocasionou a parada, sendo que o cálculo da mesma pode ser visto com mais detalhes em (OPERADOR NACIONAL DO SISTEMA - ONS, 2010). Para que se tenha uma idéia exata do impacto da *"parcela variável"* sobre a remuneração de uma empresa concessionária, (AQUINO, 2009) cita o exemplo de um único desligamento não programado de apenas 2 minutos de duração ocorrido na LT Itutinga – Juiz de Fora 345kV, que para um faturamento mensal de R$1.106.584,84, este desligamento não programado representou a incidência de uma *"parcela variável"* de R$7.684,62. A empresa concessionária deve, por obrigação da ONS, restaurar o funcionamento do sistema elétrico da área de sua concessão no tempo limite de até 1 minuto após ocorrência da falha, após isto começa a aplicar-se a penalização ao concessionário, ver (OPERADOR NACIONAL DO SISTEMA - ONS, 2007). Em até 30 minutos após a ocorrência da falha, o concessionário deverá apresentar um relatório preliminar sobre o incidente e, posteriormente, enviar um relatório final sobre o mesmo para o ONS, que dai poderá julgar se aplica ou não novas penalidades ao concessionário.

Desta forma, interessa às empresas terem sistemas que apontem as causas das paradas graves e inesperadas, o mais rápido possível, de forma a corrigirem os problemas que causaram estes tipos de paradas e diminuírem assim o impacto das multas a serem aplicadas em suas remunerações mensais.

1.2. Objetivos

Neste trabalho, é estudado e implementado um método de analisar sequências de eventos geradas pelo sistema de proteção de um sistema elétrico. Esse trabalho será usando, posteriormente, no desenvolvimento de um sistema de auxílio aos operadores de um sistema de energia elétrica, de forma a permitir que estes descubram as prováveis causas de problemas nas áreas de concessão, chegando mais rapidamente às suas soluções.

Porém, a implementação de sistemas inteligentes o bastante para fazerem deduções lógicas sobre eventos ocorridos e daí apontar possíveis causas para tais eventos não é uma tarefa trivial. Por exemplo, quais as características que se deseja para o sistema? O mesmo agirá apenas como um sistema reativo aos acontecimentos ou o mesmo será um sistema racional com a capacidade de prever acontecimentos futuros e planejar as ações necessárias antecipadamente? Qual a melhor abordagem lógica para a implementação destes sistemas?

Em relação à abordagem lógica, dentre os diversos tipos existentes, podemos destacar três: Redes Neurais, Lógica Bayesiana e Lógica do *Cálculo de Eventos*. As Redes Neurais (HAYKIN, 1999) são uma abordagem moderna bastante popular onde se utilizam algoritmos de forma a se criar sistemas computacionais estruturados em ligações semelhantes aos neurônios do cérebro humano. A Lógica Bayesiana (PEARL, 1988) é uma abordagem mais antiga que teve um grande desenvolvimento nas últimas décadas, baseada em probabilidades especificadas a priori e em tabelas de probabilidade condicional. A Lógica do *Cálculo de Eventos* (KOWALSKI, 1986) é uma abordagem mais moderna que vem sendo bastante pesquisada nas últimas três décadas, mas ainda pouco utilizada em algoritmos para sistemas de detecção de avalanches de alarmes (o escopo deste trabalho). A mesma baseia-se no uso do *Cálculo de Eventos*, a partir de cláusulas estruturadas num formalismo lógico, para representar eventos, ações e seus efeitos, de forma a se chegar a resultados lógicos satisfatórios usando vários tipos de inferências.

Dentre as várias abordagens possíveis na Lógica do *Cálculo de Eventos*, a abordagem conhecida por *Event Calculus* (EC) (SHANAHAN, 1997a; SHANAHAN, 2000) e sua variante *Simplified Event Calculus* (SEC) (SHANAHAN, 1997b) se mostra

mais vantajosa no desenvolvimento de algoritmos de detecção de avalanche de alarmes em sistemas elétricos do que as abordagens por Redes Neurais e por Lógica Bayesiana. Enquanto em *Event Calculus* conseguem-se resultados satisfatórios, baseados em formalismo lógico, na abordagem por Redes Neurais é necessário ter-se a amostragem de um grande número de eventos prévios para que o sistema adquira "*inteligência*" suficiente para começar a processar corretamente os sinais de alarmes e apontar possíveis causas para os problemas das avalanches de alarmes. Já na Lógica Bayesiana, apesar da inegável qualidade da mesma para se chegar a resultados satisfatórios precisos, tem-se o problema de se ter, para determinadas aplicações, que construir redes bayesianas gigantescas, com um número muito grande de nós raízes de probabilidades *à priori* e com tabelas de probabilidade condicional de dimensões grandes, de forma que a obtenção destas pode torna-se um impeditivo para a implementação do sistema de alarmes em si. Ainda sim, devemos concordar que para muitas situações especificas (mas diferentes dos objetivos do escopo deste trabalho) as abordagens ou por Redes Neurais ou por Lógica Bayesiana possam ser mais vantajosas computacionalmente do que pela Lógica do *Cálculo de Eventos*.

Outra abordagem que será utilizada neste trabalho será o uso de *Reconhecimento de Crônicas* (GHALLAB, 1996) para tarefa do reconhecimento dos modelos, que serão gerados com o uso de Cálculo de Eventos. Isto tornará mais eficiente e rápida a tarefa de se chegar às causas raízes das paradas que venham a ocorrer.

Posteriormente, esses algoritmos serão usados no desenvolvimento de um módulo para a Central de Alarmes e Eventos (CAE), para auxílio ao diagnóstico das causas raízes quando da ocorrência de avalanches de alarmes em sistemas de energia elétrica.

1.3. Organização deste Trabalho

No capítulo 2 discorre-se brevemente sobre como funciona a monitoração de sistemas de energia elétrica baseada em alarmes.

No capítulo 3 é mostrado um exemplo simplificado de um fragmento de um Sistema de Energia Elétrica (SEE) que será usado no capítulo 6, na demonstração da implementação dos algoritmos deste trabalho.

No capítulo 4 descreve-se brevemente o funcionamento da Central de Alarmes e Eventos (CAE), um *software* em desenvolvimento, fora do escopo desta dissertação, ao qual será aplicado futuramente o algoritmo desenvolvido neste trabalho.

No capítulo 5 começa-se com uma revisão dos aspectos teóricos utilizados para representação do conhecimento. Discorre-se sobre o funcionamento do *Processamento de Eventos*, que pode ser aplicado tanto a sistemas baseados em conhecimento de especialistas como os baseados em modelos.

No capítulo 6 é descrita a implementação do Processamento de Eventos na lógica de proteção de uma subestação, utilizando-se o exemplo de SEE simplificado apresentado no capítulo 3. Mostra-se também como os modelos das crônicas podem ser obtidos usando-se *Cálculo de Eventos*. São também discutidos os resultados desta implementação no esquema de proteção do sistema elétrico simplificado e o desempenho do algoritmo para uso futuro no desenvolvimento do módulo de *Processamento de Eventos* na CAE.

No capítulo 7, finalmente, chega-se às conclusões finais sobre os resultados alcançados, onde discorre-se sobre o que foi desenvolvido e dão-se sugestões para trabalhos futuros.

1.4. Contribuições deste Trabalho

Ao termino desta dissertação, se terá conseguido chegar à modelagem e à análise, usando *Cálculo de Eventos* e *Reconhecimento de Crônicas*, de sequencias críticas de eventos presentes em uma avalanche de alarmes em um esquema de proteção de um sistema elétrico. A partir dessa modelagem, se chegará também a algoritmos para uso em sistemas de proteção elétrica. Esses algoritmos irão ser utilizados, posteriormente, no *software* de Central de Alarmes e Eventos (CAE), na forma de um módulo de *Processamento de Eventos*, para a análise de avalanches de alarmes nas áreas sob a concessão de uma das empresas concessionárias do Sistema Elétrico Brasileiro.

2. Monitoração de SEE baseada em Eventos

Pode-se dizer de uma maneira simplificada que os Sistemas de Energia Elétrica (SEE) são compostos por unidades geradoras, linhas de transmissão, subestações de energia elétrica, redes de distribuição, etc., cada qual com vários componentes de monitoração e proteção. No escopo deste trabalho, nós limitaremos a conceituação de SEE, em *lato* e em *stricto sensu*, como mostrado na Figura 1.

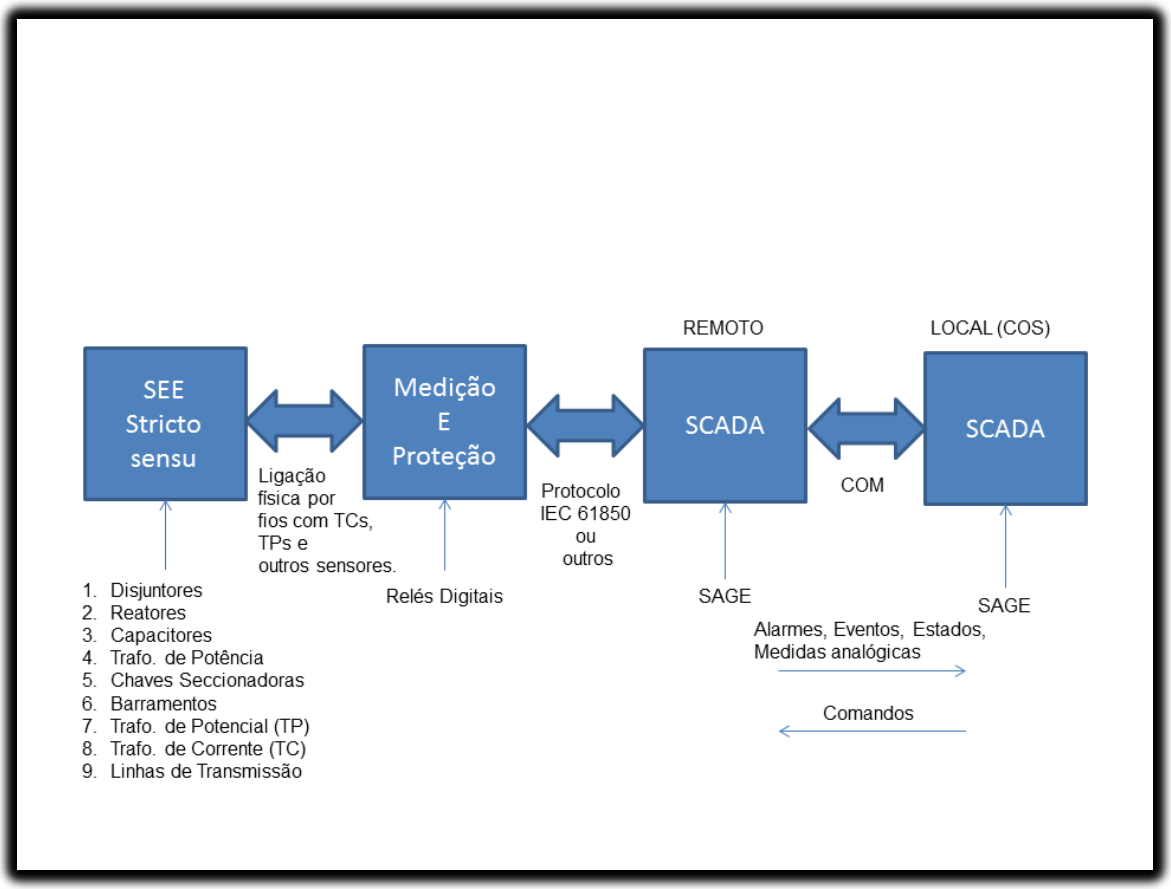


Figura 1 – Sistema de Energia Elétrica (SEE) *Lato Sensu*

Não é possível a um operador humano realizar a monitoração em tempo real de todos os componentes que compõem um SEE. Em vez disto, utilizam-se sistemas de monitoração computadorizados que mostram, em tempo real, a ocorrência dos eventos de falhas dos componentes, a atuação de proteções, as manobras feitas pelos operadores e vários tipos de mensagens. Em alguns casos, tais eventos de falhas podem

desencadear aquilo que se convencionou chamar de **avalanche de eventos de alarmes** ou, simplesmente, **avalanche de alarmes**.

Para uma melhor compreensão do funcionamento da monitoração de um SEE baseada em eventos, explicaremos a funcionalidade de cada um dos quatro blocos da imagem anterior.

2.1. SEE (em *stricto sensu*)

Neste trabalho, está-se interessado em SEEs que são compostos por subestações interligadas por linhas de transmissão (LT). As subestações possuem como componentes principais os seguintes dispositivos:

1. Disjuntores.
2. Reatores.
3. Capacitores.
4. Transformadores de Potência.
5. Chaves Seccionadoras.
6. Barramentos.
7. Transformadores de Potencial.
8. Transformadores de Corrente.

Estes dispositivos podem ser controlados por operadores na própria subestação, no chamado controle manual, ou através de controle automático ou remoto. Nos dias de hoje, o controle manual se faz apenas em casos muito excepcionais, procurando-se utilizar preferencialmente o controle automático/remoto, o que é feito pela interligação física destes dispositivos a relés digitais, que executam a função de medição e proteção dos SEE, como visto a seguir.

2.2. Medição e Proteção

Para as tarefas de medição e proteção (e eventualmente, controle) dos dispositivos de um SEE, faz-se uso de relés digitais. O relé digital SEL-421 (SCHEWEITZER ENGINEERING LABORATORIES, INC., 2013; SCHEWEITZER ENGINEERING

LABORATORIES, INC., 2011) ilustra bem a versalidade que possuem os relés digitais, atualmente disponíveis no mercado, para essas tarefas.

Os relés digitais são computadores com interfaces que fazem aquisição de dados do SEE (*strictu sensu*), i.e. tensões, correntes, energia reativa, etc, e implementam algoritmos padronizados de proteção (IEEE STANDARD ELECTRIC POWER SYSTEM, 1998).

Além disto, os relés digitais implementam, através de *software*, uma lógica de proteção definida pelos projetistas da instalação da subestação e que faz a coordenação das proteções existentes (AREVA-PROEL, 2009).

Finalmente, o relé digital possui uma interface de conexão para sistemas SAGE/SCADA (ou similar) através, por exemplo, dos protocolos IEC 61850 (ALMEIDA, 2011) ou DNP3 (TRIANGLE MICROWORKS, INC, 2002).

2.3. O Sistema SAGE

O sistema SAGE (ELETROBRAS-CEPEL, 2011) é uma plataforma adotada por vários concessionários do sistema elétrico brasileiro para o monitoramento dos sistemas elétricos das áreas de concessão. A sigla SAGE significa *Sistema Aberto de Gerenciamento de Energia*. Este sistema foi desenvolvido na época em que a ELETROBRAS – Centrais Elétricas Brasileiras S/A era a responsável por todas as etapas do fornecimento de energia elétrica no Brasil, à exceção da distribuição residencial, e o mesmo foi e continua sendo desenvolvido pelo CEPEL – *Centro de Pesquisas de Energia Elétrica da ELETROBRAS*. Porém, cabe lembrar que há outros sistemas que desempenham a mesma função do SAGE e que são também adotadas por outras concessionárias no setor elétrico brasileiro, havendo até casos de utilização simultânea de sistemas diferentes por uma mesma concessionária.

Segundo (MOREALE, 2007), o SAGE foi desenvolvido buscando-se minimizar problemas comuns aos SEE atualmente implantados em diversas empresas de concessão, devido à heterogeneidade de equipamentos e fabricantes, de forma a facilitar a incorporação de avanços tecnológicos e minimizar os custos de manutenção e expansão dos SEE.

Ainda segundo (MOREALE, 2007), o SAGE possui dois modos de funcionamento, a saber:

- O modo SAGE/SCADA (modo básico) (SCADA - *Supervisory Control and Data Acquisition System*) composto por um sistema que faz tratamento de informações (definição e manutenção *offline* da base de dados), suporte computacional (gerencia da base de dados e tarefa de controle dos processos em tempo real), aquisição e controle de dados (aquisição, tratamento e distribuição de dados e comunicação entre níveis hierárquicos diferentes) e interface gráfica GUI (*Graphical User Interface*) para a intercomunicação homem-máquina. Este modo de funcionamento é o que é utilizado pelas empresas concessionárias dos SEE tanto nas subestações de energia elétrica (SAGE/SCADA – REMOTO) como nos COS locais de hierarquia inferior de supervisão e controle (SAGE/SCADA – LOCAL) e este modo de funcionamento está relacionado diretamente ao conteúdo deste trabalho.
- O modo SAGE/EMS (modo avançado) que se constitui das funcionalidades do modo SAGE/SCADA somados às funcionalidades de análise de redes e aplicativos que possam atender a necessidades eventuais de alguns clientes. Como este modo é geralmente é utilizado apenas em níveis hierárquicos superiores de supervisão e controle (COSR e CNOS), o mesmo não é alvo deste trabalho.

Informa ainda (MOREALE, 2007) que o SAGE foi desenvolvido sobre a plataforma UNIX, compatível com o padrão X/Open, utilizando ambiente gráfico X-Windows e Motif versão 1.2, utilizando ferramentas de desenvolvimento baseadas em C, C++, ANSI Fortran 97 e, mais recentemente, também Java.

Em complementação às informações acima, atualmente o SAGE suporta como gerenciador de bancos de dados relacional os *softwares* de bancos de dados da *Oracle*, *Postgresql*, *Informix* e, mais recentemente, o *PI* da *OSIsoft* (CENTRO DE PESQUISAS EM ENERGIA ELÉTRICA - CEPEL, 2009). Em *Workstations* de usuários pode-se utilizar o *softwares* emuladores X-Windows, como o X-MING (*software Freeware*), o EXCEED (*software* proprietário da *OpenText Conectivity Solutions*), e outros similares, para rodar a interface gráfica (GUI) do SAGE.

2.3.1. SAGE/SCADA - REMOTO

Conforme dito anteriormente, os relés digitais possuem uma interface de conexão para que o sistema SAGE/SCADA.

O sistema SAGE/SCADA – REMOTO é instalado nas subestações, distantes do COS, dai a denominação *REMOTO*. O mesmo realiza a coleta dos dados disponibilizados pelos relés digitais através de *polling*. O sistema SAGE/SCADA ainda utiliza-se desta mesma interface de conexão para enviar comandos aos relés digitais.

O plano de arquitetura das subestações (AREVA-PROEL, 2009) prevê o uso de três computadores, um para a engenharia de oscilografia interna à subestação e dois rodando o *software* do sistema SAGE/SCADA, prevendo-se assim uma redundância, caso um dos computadores falhe, o outro computador assume suas funções.

O sistema SAGE/SCADA – REMOTO comunica-se com o SAGE/SCADA – LOCAL do COS através de um sistema de comunicação, que pode utilizar diferentes meios de comunicação como fibras ópticas, micro-ondas, cabos coaxiais, etc. Os dados fornecidos pelo sistema SAGE/SCADA – REMOTO ao sistema SAGE/SCADA – LOCAL do COS são alarmes, eventos, estados e medidas analógicas. Por sua vez, o mesmo pode receber de volta do SAGE/SCADA – LOCAL do COS comandos a serem transmitidos para os relés digitais.

2.3.2. SAGE/SCADA - LOCAL

O *software* do SAGE/SCADA – LOCAL é instalado no COS, distante das subestações, dai a denominação *LOCAL*. O mesmo monitora os sistemas SAGE/SCADA – REMOTO das subestações e retorna constantemente aos operadores do sistema, através de uma interface de visualização padrão (GUI), eventos, estados e dados analógicos na tela dos computadores de monitoramento e controle (o que pode ser visto na Figura 2). Como geralmente a atuação de uma proteção provoca um efeito cascata de atuação de outras proteções, pode ocorrer na tela dos computadores uma avalanche de mensagens de alarmes. Com isto, a recuperação do sistema fica

dependendo da experiência e da velocidade com a qual o operador descobre as causas do problema, as corrige e faz a restauração do estado normal do funcionamento do sistema elétrico da área de concessão.

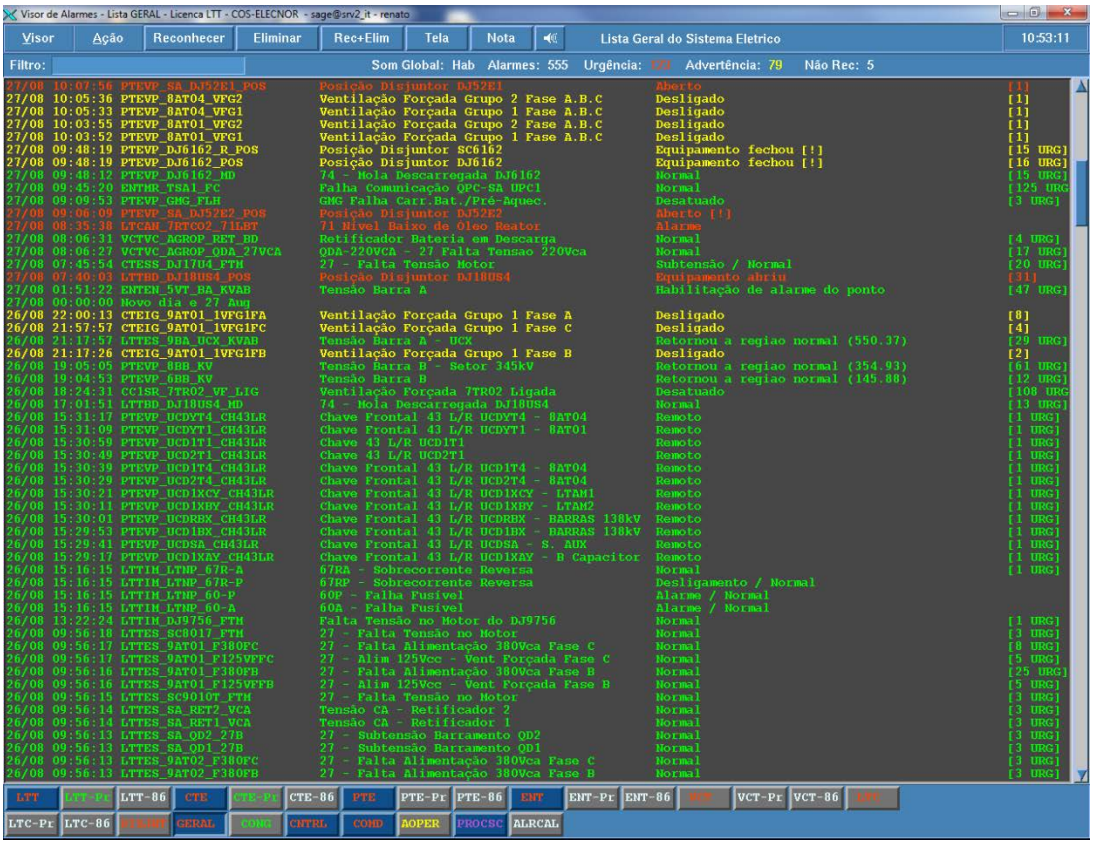


Figura 2 – Tela nativa do visualizador de eventos do SAGE

Como já mencionado no capítulo 1, quando uma falha ocorre no sistema elétrico de uma área de concessão de uma empresa operadora, a mesma deve, por norma da ONS, corrigir a falha e restaurar o funcionamento do sistema elétrico da área de concessão afetada em até 1 minuto, findo os quais a penalização da *parcela variável* começará a ser imposta à concessionária. Mesmo com a resolução do problema, a concessionária deve apresentar um relatório preliminar à ONS em até 30 minutos após a ocorrência da falha, sendo que posteriormente deverá enviar à ONS o relatório final sobre a ocorrência da falha, quando a mesma decidirá se outras penalizações poderão ou não ser aplicadas à concessionária.

Desta forma, como já dito no início deste trabalho, é de interesse das empresas concessionárias terem um *software* que faça a análise dos dados dos *históricos* do SAGE/SCADA em tempo real, ajudando aos operadores encontrarem as origens dos

problemas de forma mais rápida, para que os mesmos tomem as providências o quanto antes, reduzindo o impacto da *parcela variável*.

2.4. Monitoração Baseada em Eventos

Conforme visto nas seções anteriores deste capítulo, as subestações possuem relés digitais responsáveis pela tarefa de medição e proteção. No acontecimento de uma ou mais ocorrências, que podem ser uma falha de disjuntor, uma abertura de seccionadora, falha de aterramento, sobrecarga, etc., é gerado um evento pelo relé digital.

Esse evento é transmitido ao SAGE/SCADA – REMOTO que o repassará ao SAGE/SCADA – LOCAL no COS, sendo que este último o receberá e o acrescentará ao *histórico*.

Aqui cabe salientar que alguns eventos podem ser reconhecidos pelo SAGE/SCADA como “eventos de alarmes” ou simplesmente “alarmes”. No caso da ocorrência de um alarme o tratamento do mesmo será diferente do tratamento de um evento comum.

1. O SAGE/SCADA produzirá uma mensagem piscante na tela do operador;
2. O SAGE/SCADA acionará um sinal sonoro associado ao tipo do alarme;
3. O SAGE/SCADA requererá que o operador reconheça o alarme ocorrido, o mesmo poderá fazer isto através do botão “Reconhecer”, que pode ser vizualizada no quadrante superior esquerdo da tela do SAGE/SCADA em seu console de monitoramento.

Cabe salientar que na ocorrência de em evento, o relé digital possui uma codificação própria para o mesmo que é diferente da codificação do SAGE/SCADA. Ao ser transmitido ao SAGA/SCADA, este irá consultar uma tabela de conversão, que relaciona a codificação utilizada pelo relé digital com a codificação utilizada pelo próprio SAGE/SCADA.

Cabe ainda colocar que, como cada relé digital está conectado a um aparelho de GPS, os *timestamps* dos eventos gerados pelos relés digitais, e recebidos pelo SAGE/SCADA de cada subestação, estão perfeitamente sincronizados no tempo.

Como os eventos no *histórico* do SAGE/SCADA são padronizados e possuem um *timestamp* preciso é confiável, é possível fazer-se com segurança uma monitoração dos mesmos de forma a utilizá-los para a procura de padrões de ocorrências de eventos que possam sinalizar ações de dispositivos de proteção. Isto é a base para a utilização de técnicas de *Processamento de Eventos*, que serão vistas mais à frente no capítulo 5.

3. Exemplo de SEE

Neste capítulo, apresenta-se um exemplo de uma seção de uma subestação usada em uma linha de transmissão de um trecho de um SEE.

Este exemplo é uma simplificação da subestação de Barra dos Coqueiros 230kV (AREVA-PROEL, 2009) e será usado para aplicar os métodos de processamento de eventos que serão apresentados nos próximos capítulos.

Na Figura 3, pode-se visualizar o diagrama unifilar deste exemplo. Cabe salientar que esta seção de uma subestação simplificada utiliza apenas 1 relé digital (UCD1). Internamente, o circuito lógico deste relé, além de receber medidas e controlar o funcionamento do disjuntor e das secionadoras, recebe também alguns sinais dos demais relés digitais da mesma subestação, assim como fornece sinais para outros relés digitais.

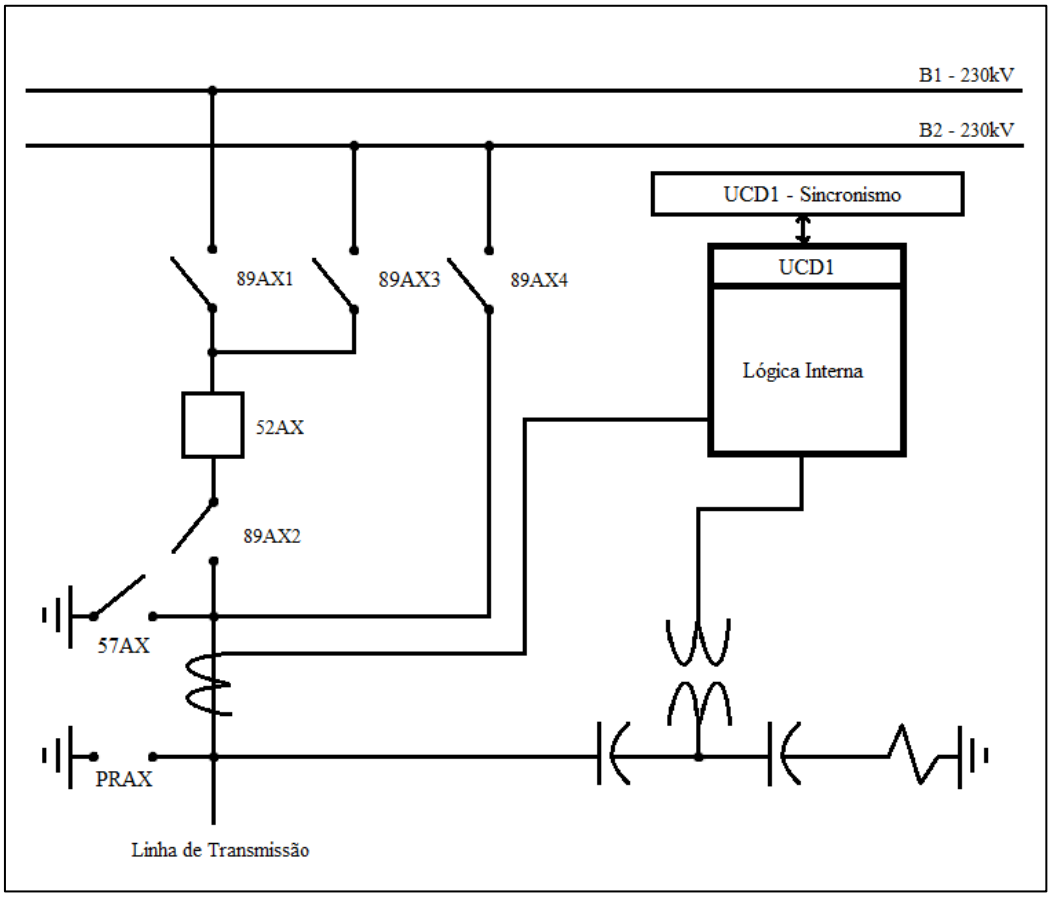


Figura 3 – Unifilar de uma seção da subestação do exemplo

O exemplo de *Processamento de Eventos* que será utilizado com esta seção da subestação refere-se a ocorrência da falha do fechamento do disjuntor 52AX numa manobra de troca de barramentos.

O cenário que leva à ocorrência do problema é o seguinte: As secionadoras 89AX1 e 89AX2 encontram-se inicialmente fechadas, assim como o disjuntor 52AX. As secionadoras 89AX3, 89AX4 e 57AX estão abertas. O barramento B1 está energizado e o barramento B2 está desenergizado. Ocorre então uma manobra programada onde tem-se a intenção de fazer uma troca de barramentos, desenergizando-se B1 e energizando-se B2.

Para que esta manobra ocorra corretamente, deve-se primeiramente abrir o disjuntor 52AX, de forma a desenergizar o barramento B1 e, por conseguinte, a secionadora 89AX1. Depois disto, a secionadora 89AX1 poderá ser aberta e a secionadora 89AX3 poderá ser fechada, o que habilitará o fechamento do disjuntor 52AX. No cenário do problema, é justamente neste momento que ocorrerá a falha do disjuntor 52AX, que impedirá o seu fechamento e a falta da energização do barramento B2, o que ocasiona a falha da operação de manobra entre barramentos. Como medida de segurança, o sistema de proteção então abrirá as secionadoras 89AX3 e 89AX2 para isolar o disjuntor 52AX.

Subentende-se que existem relés digitais que fazem a aquisição de dados da subestação, tais como o estado, e a mudanção de estado, das chaves secionadoras, do disjuntor, dos comandos de seleção, abertura, fechamento, permissão de manobra, etc.

A lógica mostrada nas páginas do Diagrama Lógico da Proteção é implementada por *software* nos relés digitais, de uma forma semelhante aos CLP’s industriais. As entradas desta lógica são captadas pelos relés digitais nos equipamentos da subestação ou da interface homem-máquina (IHM) dos próprios relés digitais, ou das proteções padronizadas (IEC), implementadas pelos relés digitais.

As saídas desta lógica vão gerar eventos, que poderão ser transformados em alarmes no SAGE/SCADA, ou comandar a abertura ou fechamento do disjuntor, das chaves secionadoras ou serem realimentadas na própria lógica.

Para cada página do Diagrama Lógico de Proteção, que representa uma parte do circuito da lógica de proteção, se procurará fazer uma descrição resumida da sua função na lógica de proteção.

A folha 015 do Diagrama Lógico de Proteção mostra a lógica utilizada, a partir das entradas de sinais digitas provenientes dos relés, para as secionadoras 57AX e 89AX2. A lógica retorna as variáveis internas dos estados das secionadoras (abertas, fechadas ou indefinido), variáveis estas que serão utilizadas em outras seções da lógica de proteção. Os temporizadores pickups têm a função de evitar a ocorrência do estado indefinido durante as transições de abertura e fechamento das secionadoras.

A folha 016 do Diagrama Lógico de Proteção mostra a lógica descrita para a secionadora 89AX4 e seus estados (aberta, fechada, indeterminada e intermediária). A lógica retorna as variáveis internas dos estados das secionadoras, variáveis estas que serão utilizadas em outras seções da lógica de proteção. Os temporizadores pickups têm a função de evitar a ocorrência do estado indefinido durante as transições de abertura e fechamento das secionadoras.

A folha 017 do Diagrama Lógico de Proteção mostra a lógica descrita para as secionadoras 89AX3 e 89AX1 e seus estados (aberta, fechada, indeterminada e intermediária). A lógica retorna as variáveis internas dos estados das secionadoras, variáveis estas que serão utilizadas em outras seções da lógica de proteção. Os temporizadores pickups têm a função de evitar a ocorrência do estado indefinido durante as transições de abertura e fechamento das secionadoras.

A folha 018 do Diagrama Lógico de Proteção mostra a lógica descrita para o disjuntor 52AX e seus estados (aberto, fechado, indefinido). A lógica retorna as variáveis internas dos estados das secionadoras, variáveis estas que serão utilizadas em outras seções da lógica de proteção. Os temporizadores pickups têm a função de evitar a ocorrência do estado indefinido durante as transições de abertura e fechamento das secionadoras.

A folha 080 do Diagrama Lógico de Proteção mostra a lógica para a operação de comandos, remotos ou locais através de cartões, para a secionadora 89AX1. É a partir das entradas desta lógica que são transmitidos os comandos de abertura e fechamento para a secionadora 89AX1 por meio de variáveis internas.

A folha 081 do Diagrama Lógico de Proteção mostra a lógica para a operação de comandos, remotos ou locais através de cartões, para a secionadora 89AX3. É a partir das entradas desta lógica que são transmitidos os comandos de abertura e fechamento para a secionadora 89AX3 por meio de variáveis internas.

A folha 082 do Diagrama Lógico de Proteção mostra a lógica para a operação de comandos, remotos ou locais através de cartões, para a secionadora 89AX2. É a partir das entradas desta lógica que são transmitidos os comandos de abertura e fechamento para a secionadora 89AX2 por meio de variáveis internas.

A folha 083 do Diagrama Lógico de Proteção mostra a lógica para a operação de comandos, remotos ou locais através de cartões, para a secionadora 57AX. É a partir das entradas desta lógica que são transmitidos os comandos de abertura e fechamento para a secionadora 57AX por meio de variáveis internas.

A folha 084 do Diagrama Lógico de Proteção mostra a lógica para a operação de comandos, remotos ou locais através de cartões, para a secionadora 89AX4. É a partir das entradas desta lógica que são transmitidos os comandos de abertura e fechamento para a secionadora 89AX4 por meio de variáveis internas.

A folha 085 do Diagrama Lógico de Proteção mostra a lógica para a operação de comandos, remotos ou locais através de cartões, para o disjuntor 52AX. É a partir das entradas desta lógica que são transmitidos os comandos de abertura e fechamento para o disjuntor 52AX por meio de variáveis internas.

A folha 086 do Diagrama Lógico de Proteção mostra a lógica para a operação de comandos, remotos ou locais, para sinalizar o bloqueio/desbloqueio do disjuntor 52AX. É a partir das entradas desta lógica que são transmitidos os comandos de bloqueio e desbloqueio para o disjuntor 52AX por meio de variáveis internas.

A folha 087 do Diagrama Lógico de Proteção mostra a lógica para a seleção de transferência de proteção (Normal, Em Transferência e Transferida), a transferência pode ser realizada à nível local ou remoto, tais quais os circuitos anteriores, a saída desta lógica se dá por meio de variáveis internas.

A folha 088 do Diagrama Lógico de Proteção mostra a lógica para a seleção do tipo de religamento que será ativado para o disjuntor 52AX (Monopolar, Tripolar ou Desligado), tais quais os circuitos anteriores, a saída desta lógica se dá por meio de variáveis internas.

A folha 053 do Diagrama Lógico de Proteção mostra a lógica para o circuito de *TDD Mantido* vindo do relé UDPX1, com as lógicas para a entrada de TRIP e que serão usadas no circuito de liberação/bloqueio de fechamento do disjuntor 52AX na seção

deste circuito vista na folha 025. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 063 do Diagrama Lógico de Proteção mostra a lógica para o circuito de *TDD Mantido* vindo do relé UDPX2, com as lógicas para a entrada de TRIP e que serão usadas no circuito de liberação/bloqueio de fechamento do disjuntor 52AX na seção deste circuito vista na folha 025 (a mesma já referida na explicação da folha 053). A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 019 do Diagrama Lógico de Proteção mostra a lógica que dá a permissão para a execução da manobra da secionadora 89AX1, após serem satisfeiras as condições para a abertura ou o fechamento da mesma. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 020 do Diagrama Lógico de Proteção mostra a lógica que dá a permissão a execução da manobra da secionadora 89AX3, após serem satisfeiras as condições para a abertura ou o fechamento da mesma. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 021 do Diagrama Lógico de Proteção mostra a lógica que dá a permissão a execução da manobra da secionadora 89AX2, após serem satisfeiras as condições para a abertura ou o fechamento da mesma. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 022 do Diagrama Lógico de Proteção mostra a lógica que dá a permissão a execução da manobra da secionadora 57AX, após serem satisfeiras as condições para a abertura ou o fechamento da mesma. A saída desta lógica se dá por conexões físicas aos equipamentos de proteção.

A folha 023 do Diagrama Lógico de Proteção mostra a lógica que dá a permissão a execução da manobra da secionadora 89AX4, após serem satisfeiras as condições para a abertura ou o fechamento da mesma. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 024 do Diagrama Lógico de Proteção mostra a lógica que dá a permissão a execução da manobra do disjuntor 52AX, após serem satisfeiras as condições para a

abertura ou o fechamento da mesma. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 025 do Diagrama Lógico de Proteção mostra a logica de liberação das secionadoras isoladoras, verificando se houve ou não falha do disjuntor 52AX e bloqueia ou desbloqueio o fechamento do mesmo em caso de falha. A saída desta lógica se dá por variáveis internas.

A folha 026 do Diagrama Lógico de Proteção mostra a lógica para a seleção de transferência de proteção (Normal, Em Transferência e Transferida) e sinaliza as atuações/desatuações sobre os relés 43T e 43N. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 027 do Diagrama Lógico de Proteção mostra a lógica da transferência de proteção (Normal, Em Transferência, Transferida). Este circuito alimenta várias entradas do circuito de seleção de transferência de proteção da figura 27. A saída desta lógica se dá por variáveis internas.

A folha 028 do Diagrama Lógico de Proteção mostra a lógica de seleção de tensão para sincronismo. Este circuito verifica o estado do enrolamento do TPAX, faz a seleção da tensão de barra (1 ou 2) e sinaliza a permissão de manobra do disjuntor 52AX para os circuitos lógicos de controle de fechamento/abertura das secionadoras 89AX1, 89AX2, 89AX3, 89AX4, 57AX e do disjuntor 52AX, vistos nas figuras anteriores. A saída desta lógica se dá por variáveis internas e por conexões físicas aos equipamentos de proteção.

A folha 029 do Diagrama Lógico de Proteção mostra a lógica de seleção de religamento para o disjuntor 52AX, onde será selecionado se o disjuntor fará religamento monopolar ou tripolar e se o religamento do mesmo se encontra ligado ou desligado. A saída desta lógica se dá por variáveis internas.

4. A Central de Alarmes e Eventos

A Central de Alarmes e Eventos (CAE) é um sistema de programas que está sendo desenvolvido no âmbito de um projeto de P&D da ANEEL (projeto nº 001/11) que pode ser visto como uma extensão do sistema SAGE/SCADA. A CAE é um sistema baseado na WEB e, como tal, possui clientes baseados em navegadores e um servidor HTTP.

O servidor da CAE está sendo desenvolvido em JAVA, sobre o hospedeiro de *servlets* APACHE TOMCAT. Os clientes utilizam intensivamente a linguagem JavaScript e foram testados, principalmente, no navegador Google Chrome versão 29.

A CAE está sendo desenvolvida também no âmbito de uma tese de mestrando (COLLAVIZZA, 2013). O objetivo da CAE é, quando da ocorrência de uma avalanche de alarmes, enviar as informações destes eventos para um processo que rodará o algoritmo desenvolvido neste trabalho. Este processo executará um algoritmo de *Reconhecimento de Crônicas*, e retornará à CAE os resultados obtidos, de forma a auxiliar ao operador do sistema na busca pelas causas raízes de acarretaram a avalanche de alarmes.

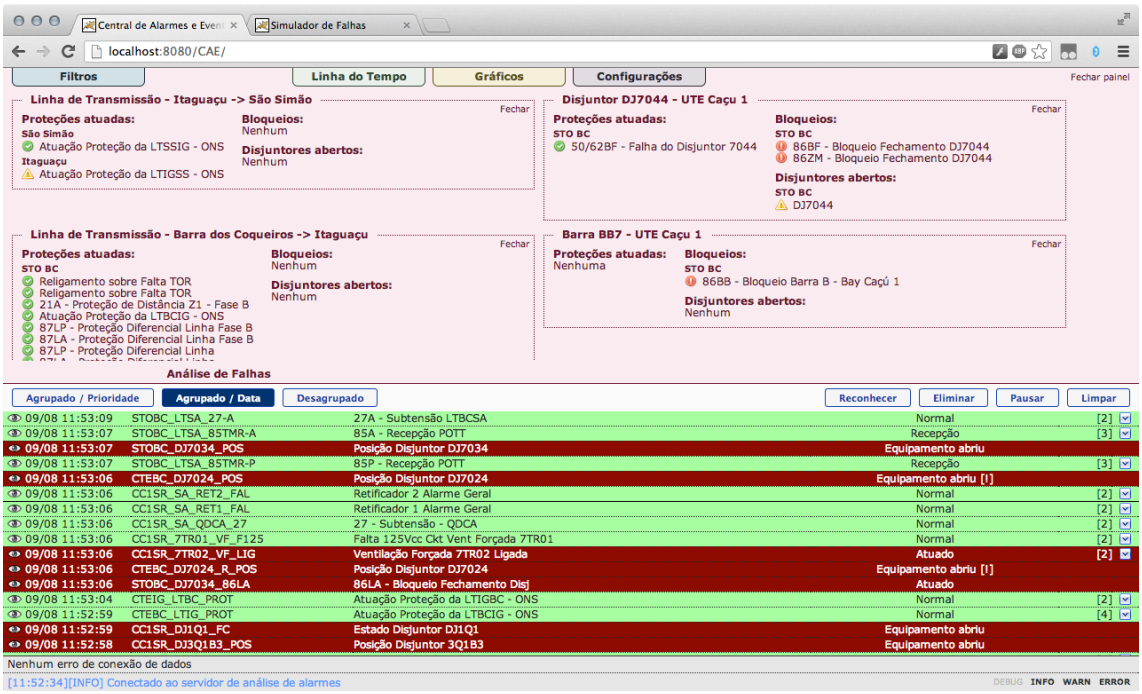


Figura 4 – Tela da Central de Alarmes e Eventos

4.1. Interface mais simples e amigável

A interface padrão que vem no sistema SAGE/SCADA, mostrada na Figura 2, é basicamente uma visualização em tempo real dos acontecimentos relatados no *histórico* do SAGE/SCADA. O operador dispõe de poucas opções de configuração de tela e o maior problema se nota quando ocorre de uma avalanche de alarmes. Neste momento centenas de mensagens aparecem na tela em um intervalo de poucos segundos, o que torna difícil a compreensão do fato ocorrido, obrigando o operador a agir rapidamente e a retroceder rapidamente nas mensagens, pois ele dispõe de apenas 1 minuto para restabelecer o funcionamento do sistema elétrico afetado antes da incidência da penalidade da *parcela variável* sobre a concessionária.

Na Figura 5 e nas Figura 6 e Figura 7, podemos perceber que a interface da CAE apresenta opções novas, que permitem uma melhor visualização das mensagens de evento, com possibilidade de agrupá-las pela data de ocorrência ou pelo seu grau de prioridade.

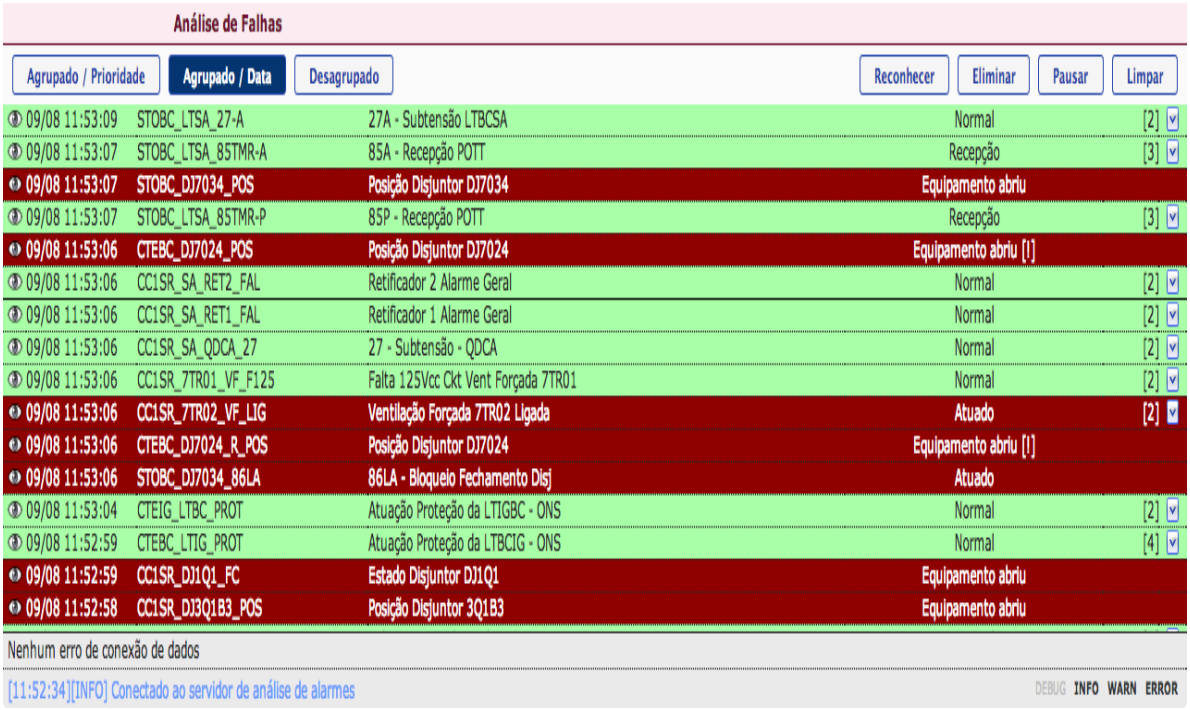


Figura 5 – Detalhe da tela do CAE mostrando os novos recursos de *log*

Quando em modo agrupado, a CAE permite que o operador abra a linha da mensagem na tela, visualizando desta forma as ocorrências anteriores relacionadas a esta mesma mensagem.

4.2. Resumo das ocorrências

Outra funcionalidade disponibilizada pela CAE é a caixa de resumo de ocorrências, onde o operador poderá visualizar um resumo dos eventos ocorridos em cada equipamento, quais disjuntores abriram, quais proteções atuaram e os bloqueios, conforme mostrado nas Figura 6 e Figura 7.

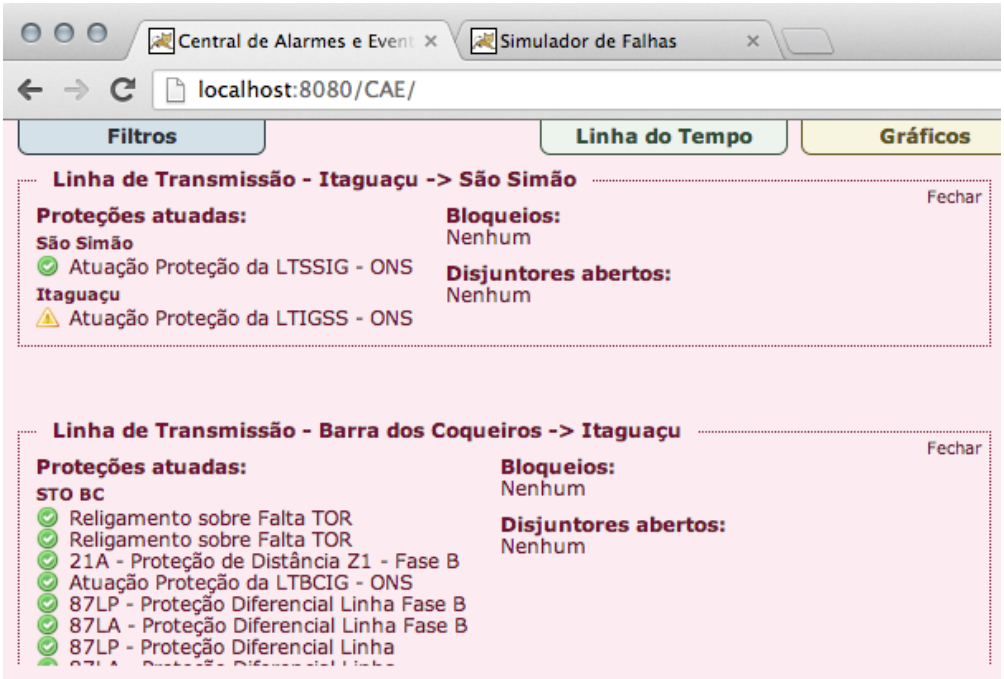


Figura 6 – Tela de resumo de ocorrências - I (detalhe)

Com isto, fica mais fácil para o operador ter uma noção dos eventos do *histórico* que aparecem na tela de análise de falhas, logo abaixo na CAE.

Essa tela possui a funcionalidade de rolagem (*scroll*) o que permite que vários dispositivos afetados possam ser visualizados pelo operador, bastando que o mesmo clique na tela com o *mouse* e role a mesma para cima.

Há ainda a possibilidade de o operador fechar as caixas de resumo de ocorrência que não sejam úteis para a solução do problema, permitindo que ele deixe a tela de resumo de ocorrências menos carregada de informações.

Figura 7 – Tela de resumo de ocorrências - II (detalhe)

Finalmente, conforme as ocorrências aconteçem, a tela de resumo de ocorrências é atualizada em tempo real para o operador, facilitando que o mesmo identifique os pontos mais críticos que devem ser analisados primeiro.

5. Processamento de Eventos

5.1. Introdução

Como visto nos capítulos anteriores, o *Processamento de Eventos* em sistemas de monitoração, como o SAGE/SCADA, não é uma tarefa trivial devido à quantidade e à variedade de eventos envolvidos. Pode-se classificar o processamento de eventos em *Processamento de Eventos Baseado em Conhecimento* e o *Processamento de Eventos Baseado em Modelo*. No final deste capítulo, ficará claro para o leitor porque o uso do *Processamento de Eventos Baseado em Modelo* na lógica de proteção de uma subestação.

Será feita inicialmente uma revisão dos aspectos teóricos que serão utilizados para, posteriormente, explicar o *Processamento de Eventos Baseado em Conhecimento* e o *Processamento de Eventos Baseado em Modelo*.

Trechos deste capítulo foram baseados no livro *Solving the Frame Problem – A Mathematical Investigation of the Common Sense Law of Inertia* de autoria de Murray Shanahan (SHANAHAN, 1997b).

5.1.1. Representação do tempo na lógica

A representação do tempo na lógica apresenta certa complexidade. Geralmente não está clara a noção do que é a representação do tempo em relação ao que é a representação da ocorrência de eventos, porque a ocorrência de um evento pode parecer dissociada de uma cronologia de tempo. Durante muito tempo isto foi objeto de estudos e a revisão dos aspectos teóricos, a seguir, irá dar mais claridade sobre a evolução da ideia da representação do tempo na lógica, além de apresentar ao leitor as noções de monotonicidade e não-monotonicidade, circunscrição, abdução, etc.

5.1.1.1. *Frame Problem* (Problema do Contorno)

O ***Frame Problem*** (**Problema do Contorno**, do Entorno ou do Enquadramento) foi descrito inicialmente por McCarthy e Heyes (McCARTHY, et al., 1969). Segundo

(SHANAHAN, 1997b): "*O mesmo surgiu da necessidade de se descrever os efeitos de ações ou eventos usando-se uma abordagem lógica. Em essência, no* ***Frame Problem*** *procura-se descrever o que é alterado quando uma determinada ação ou um determinado evento ocorre. Porém, também se deve evitar ter que descrever tudo aquilo que não é alterado por essa ação ou evento.*"

Murray Shanahan (SHANAHAN, 1997b) usa um exemplo bem didático para ilustrar o *Frame Problem*:

- Pintar a parede do meu escritório muda a cor da mesma.

Porém:

- Pintar a parede do meu escritório não altera o seu formato;
- Pintar a parede do meu escritório não altera o penteado do meu cabelo;
- Pintar a parede do meu escritório não precipitam eleições gerais;
- Pintar a parede do meu escritório não faz o sol nascer.
- E assim por diante.

Dessa forma, segundo (SHANAHAN, 1997b): "*Verifica-se, usando os recursos da lógica clássica de primeira-ordem, que a quantidade de coisas que não são alteradas é infinita e é potencialmente impossível descrevê-las todas. Assim, devemos nos concentrar naquilo que muda com o efeito da ação ou evento, e devemos ser capazes de tomar aquilo que não mudou como algo garantido e certo. Logo, o* ***Frame Problem*** *se resume a construir-se um quadro formal que nos permita fazer exatamente isto.*"

Existem dois formalismos principais de raciocínio que podem ser usados na solução do ***Frame Problem***, o *Cálculo das Situações* e o *Cálculo dos Eventos*. A seguir, são apresentados esses dois formalismos e três formas canônicas de inferência que podem ser utilizados junto com os mesmos, o *Raciocínio Indutivo*, o *Raciocínio Dedutivo* e o *Raciocínio Abdutivo*. Apresentar-se-ão também os conceitos de *Monotonicidade* e *Não-Monotonicidade*, *Lei da Inércia* e *Circunscrição*.

5.1.1.2. Cálculo das Situações

O **Cálculo das Situações** foi primeiramente descrito por John McCarthy em 1963 em um artigo não publicado (McCARTHY, 1963) e que posteriormente foi incorporado ao trabalho de McCarthy e Heyes (McCARTHY, et al., 1969).

A ontologia do **Cálculo das Situações** se baseia em:

- Situações: As situações representam os instantâneos dos acontecimentos do "mundo" (ou o problema considerado).
- Fluentes: Os fluentes têm a característica de assumirem diferentes valores, dependendo da situação, e podem ser considerados com propriedades que variam no tempo.
- Ações: As ações são responsáveis pela alteração dos valores dos fluentes.

Em adição a essa ontologia, a linguagem do **Cálculo das Situações** inclui ainda o símbolo de predicado *Holds* e o símbolo de função *Result*.

Escrevemos ***Holds(f,s)*** para denotar que o fluente ***f*** é verdadeiro na situação ***s*** e escrevemos ***Result(a,s)*** para denotar a nova situação que se obtém ao realizar a ação ***a*** na situação prévia ***s***. Com isto, pode-se então escrever fórmulas que descrevam os efeitos das ações.

Para uma exemplificação do funcionamento do **Cálculo das Situações**, mostraremos uma variante simplificada do exemplo clássico do *Mundo dos Blocos* descrito por Shanahan (SHANAHAN, 1997b), onde nossa intenção é mover um bloco, passando da situação **S0** para a situação **S1**. Suponhamos que temos uma situação inicial **S0** deste exemplo que pode ser vista na Figura 8. Também necessitamos introduzir dois fluentes ***On(x,w)*** para denotar que o bloco ***x*** está sobre **w** (onde ***w*** pode ser a mesa **M** ou um dos blocos **A**, **B** ou **C**) e ***Clear(w)*** para denotar que o topo de ***w*** possui um espaço livre para se colocar um dos outros blocos.

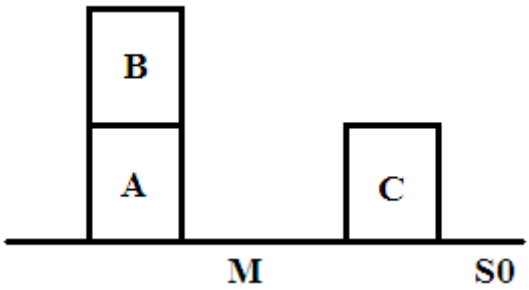


Figura 8 – Exemplo de Cálculo das Situações - Situação S0

O conjunto de fórmulas abaixo (que chamaremos de Σ) representa parcialmente a situação S0:

Holds(On(A,M),S0) (1)

Holds(On(B,A),S0) (2)

Holds(Clear(B),S0) (3)

Holds(On(C,M),S0) (4)

Holds(Clear(C),S0) (5)

Holds(Clear(M),S0) (6)

Quando se diz *"parcialmente"*, quer-se dizer que somente se pode representar informação positiva a respeito de **S0**, logo Σ não representa uma situação completa de S0. Estas fórmulas são interpretadas como fórmulas clássicas da lógica e não permitem que se tirem conclusões da forma ¬Holds(On(w,z),S0) ou ¬Holds(Clear(w),S0).

Suponhamos que se queira agora passar para a situação **S1**, mostrada na Figura 9.

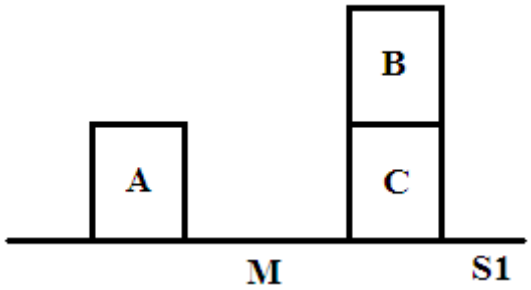


Figura 9 – Exemplo de Cálculo das Situações – Situação S1

Para isto, agora deve-se introduzir um tipo de ação denominada ***Move(x,w)***, onde x representa um bloco (**A**, **B** ou **C**) e ***w*** representa a mesa **M** ou um dos outros dois blocos restantes. Assim, a nova situação **S1** é denotada pela fórmula:

$$\text{Result}(\text{Move}(B,C),S0) \qquad (7)$$

Porém, isso apenas não é suficiente para descrever o processo de se passar da situação **S0** para a situação **S1**, pois deve-se escrever fórmulas, chamadas de *axiomas de efeito*, que capturarão o efeito da ação **Move** sobre os fluentes **On** e **Clear**. Os *axiomas de efeito* são usados para predizer o resultado de uma sequência de ações e levam em conta as precondições necessárias para que a ação ocorra com sucesso (por exemplo, neste caso é necessário que o topo de C esteja livre).

O conjunto de fórmulas abaixo, que será chamado de Δ, descreve o *axioma de efeito* deste exemplo:

$$\text{Holds}(\text{On}(x,y),\text{Result}(\text{Move}(x,y),s)) \leftarrow \text{Holds}(\text{Clear}(x),s) \wedge \text{Holds}(\text{Clear}(y),s) \wedge x \neq y \wedge x \neq M \qquad (8)$$

$$\text{Holds}(\text{Clear}(z),\text{Result}(\text{Move}(x,y),s)) \leftarrow \text{Holds}(\text{Clear}(x),s) \wedge \text{Holds}(\text{Clear}(y),s) \wedge \text{Holds}(\text{On}(s,z),s) \wedge x \neq z \wedge x \neq y \qquad (9)$$

Novamente aqui, vemos que as fórmulas do *axioma de efeito* descritas capturam somente a informação positiva do problema, porém esta é toda a informação necessária para a solução do *Frame Problem* desta situação.

Assim, tomando-se a conjunção de todas as conclusões de Δ e Σ, teremos:

$$\Delta \wedge \Sigma \models \text{Holds}(\text{On}(B,C),\text{Result}(\text{Move}(B,C),S0)) \qquad (10)$$

$$\Delta \wedge \Sigma \models \text{Holds}(\text{Clear}(A),\text{Result}(\text{Move}(B,C),S0)) \qquad (11)$$

Apesar de o exemplo anterior parecer simples, deve-se lembrar que o mesmo é uma simplificação e que vários aspectos de um problema mais abrangente foram relevados. Por exemplo, se a cada movimento de blocos, o mesmo fosse pintado de uma cor diferente da anterior, seria necessário descrever um conjunto de fórmulas e axiomas para denotar essa ação na nova situação.

Em verdade, se este universo de blocos fosse aumentado, poder-se-ia chegar a um número gigantesco de fórmulas. Isto novamente representa bem o que é o *Frame*

Problem, a busca por uma representação lógica formal sem que se necessite ter que escrever um número enorme de fórmulas axiomáticas.

5.1.1.3. Cálculo dos Eventos

O **Cálculo dos Eventos** foi primeiramente descrito por Robert A. Kowalski e Marek Sergot em 1986 (KOWALSKI, 1986) e é uma evolução do *Cálculo das Situações* de McCarthy e Heyes (McCARTHY, et al., 1969). Segundo Kowalski e Sergot, em seu trabalho original, a principal diferença do **Cálculo de Eventos** em relação ao *Cálculo das Situações* é que enquanto este lida com estados globais para a resolução do *Frame Problem*, aquele lida com eventos locais e períodos de tempo para resolver o problema, podendo ainda lidar com eventos simultâneos e parcialmente ordenados. Após a publicação do trabalho de Kowalski e Sergot, seguiram-se diversos trabalhos a respeito do **Cálculo de Eventos** e vários dialetos do mesmo foram criados. Neste trabalho focou-se na variante descrita por Rob Miller e Murray Shanahan (SHANAHAN, 1997a; SHANAHAN, 2000; MILLER, 1999; MILLER, 2002), chamada simplesmente de *Event Calculus* (EC).

Outra motivação para o desenvolvimento do **Cálculo dos Eventos** foi o seu uso no tratamento de eventos temporais, o que não se mostrava viável com o uso do *Cálculo das Situações*. Ainda que já se tenha provado, atualmente, que o tratamento de eventos temporais não seria impedimento para o uso do *Cálculo das Situações*, Gelfond, Lifschitz e Rabinov (GELFOND, 1991) *apud* Shanahan (SHANAHAN, 1997b), seriam necessárias extensões ao *Cálculo das Situações* para que o mesmo pudesse fazer um tratamento temporal eficiente, o que não é necessário usando o **Cálculo dos Eventos**, e ainda se teriam os problemas das mudanças de estado a nível global e ao tratamento de eventos simultâneos e parcialmente ordenados, como já dito antes.

A ontologia do **Cálculo dos Eventos** se baseia em:

- Eventos ou Ações (*Tipos*): Tal qual no *Cálculo das Situações*, são responsáveis pela alteração dos valores dos fluentes.
- Fluentes: Os fluentes têm a característica de assumirem diferentes valores ao longo do tempo. Podem ser tanto uma quantificação numérica (i.e. a temperatura em um aposento) como uma proposição (i.e. está chovendo).

- Pontos de Tempo: Descrevem o momento em que um evento modificou um fluente. Por exemplo: “Rita foi professora (fluente) daquela escola até 1983 (ponto de tempo), quando pediu exoneração (evento) do cargo”.

A Figura 10 mostra uma exemplificação simplificada do funcionamento do **Cálculo de Eventos** descrito por Kowalski e Sergot:

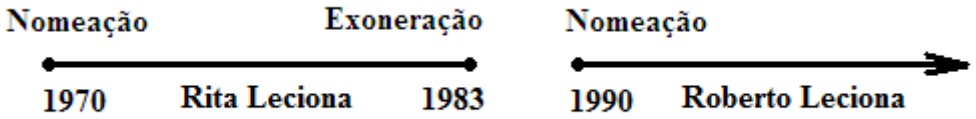


Figura 10 – Exemplo simplificado do Cálculo de Eventos

Além desta ontologia, o **Cálculo de Eventos** possui ainda uma linguagem com predicados associados. Na tabela a seguir são descritos os predicados principais da linguagem do **Cálculo de Eventos** (variante **EC**):

Tabela 1 – Predicados da linguagem do Cálculo de Eventos/EC

Predicado (Formula)	Descrição
Initiates(α,β,τ)	O fluente β se mantém (holds) após a ação α ocorrer no tempo τ
Terminates(α,β,τ)	O fluente β não se mantém (not holds) após a ação α ocorrer no tempo τ
Releases(α,β,τ)	O fluente β não está sujeito à lei do senso comum da inércia após a ação α ocorrer no tempo τ
Initially$_P$(β)	O fluente β se mantém (holds) a partir do tempo 0
Initially$_N$(β)	O fluente β não se mantém (not holds) a partir do tempo 0

Happens(α,τ1,τ2)	A ação α se inicia no tempo τ1 e termina no tempo τ2
HoldsAt(β,τ)	O fluente β se mantém (holds) no tempo τ
Clipped(τ1,β,τ2)	O fluente β é encerrado entre os tempos τ1 e τ2
Declipped(τ1,β,τ2)	O fluente β é iniciado entre os tempos τ1 e τ2

Usando-se os predicados da Tabela 1, Shanahan e Miller chegaram aos sete axiomas que em conjunto formam o seu dialeto do Cálculo dos Eventos (EC), visto na Tabela 2:

Tabela 2 – Axiomas do Cálculo de Eventos/EC de Shanahan e Miller

Axioma	Índice
$\text{HoldsAt}(f,t) \leftarrow \text{Initially}_{\mathbf{P}}(f) \wedge \neg \text{Clipped}(1,f,t)$	EC1
$\text{HoldsAt}(f,t3) \leftarrow \text{Happens}(a,t1,t2) \wedge \text{Initiates}(a,f,t1) \wedge t2 < t3 \wedge \neg \text{Clipped}(t1,f,t3)$	EC2
$\text{Clipped}(t1,f,t4) \leftrightarrow \exists\, a,t2,t3\ [\text{Happens}(a,t2,t3) \wedge t1 < t3 \wedge t2 < t4 \wedge [\text{Terminates}(a,f,t2) \vee \text{Releases}(a,f,t2)]]$	EC3
$\neg\, \text{HoldsAt}(f,t) \leftarrow \text{Initially}_{\mathbf{N}}(f) \wedge \neg \text{Declipped}(0,f,t)$	EC4
$\neg\, \text{HoldsAt}(f,t3) \leftarrow \text{Happens}(a,t1,t2) \wedge \text{Terminates}(a,f,t1) \wedge t2 < t3 \wedge \neg \text{Declipped}(t1,f,t3)$	EC5
$\text{Declipped}(t1,f,t4) \leftrightarrow \exists\, a,t2,t3[\text{Happens}(a,t2,t3) \wedge t1 < t3 \wedge t2 < t4 \wedge [\text{Initiates}(a,f,t2) \vee \text{Releases}(a,f,t2)]]$	EC6

$Happens(a,t1,t2) \rightarrow t1 \leq t2$	EC7

O **Cálculo de Eventos/EC** pode ser estendido com a adição de três novos predicados, conforme mostrado na Tabela 3:

Tabela 3 – Predicados adicionais do Cálculo de Eventos de Shanahan e Miller

Cancels($\alpha1$, $\alpha2$, β)	A ocorrência do evento $\alpha1$ cancela o efeito da ocorrência simultânea do evento $\alpha2$ sobre o fluente β
Cancelled(α, β, $\tau1$, $\tau2$)	Algum evento ocorre entre os instantes $\tau1$ e $\tau2$ que cancela o efeito do evento α sobre o fluente β
trajectory($\beta1$, τ, $\beta2$,δ)	Se o fluente $\beta1$ inicia-se no instante τ, então o fluente $\beta2$ torna-se verdadeiro no instante $\tau+ \delta$

Com a adição destes três novos predicados, o **Cálculo de Eventos/EC** da tabela 2 é estendido com novos axiomas, todos os predicados do **Cálculo de Eventos/EC Estendido** podem ser vistos na Tabela 4:

Tabela 4 – Axiomas do Cálculo de Eventos/EC Estendido de Shanahan e Miller

Axioma	Índice
$HoldsAt(f,t) \leftarrow Initially_{\mathbf{P}}(f) \wedge \neg Clipped(1,f,t)$	EC1
$HoldsAt(f,t3) \leftarrow Happens(a,t1,t2) \wedge Initiates(a,f,t1) \wedge t2 < t3 \wedge \neg Clipped(t1,f,t3)$	EC2

$Clipped(t1,f,t4) \leftrightarrow \exists\ a,t2,t3\ [Happens(a,t2,t3) \wedge t1 < t3 \wedge t2 < t4 \wedge [Terminates(a,f,t2) \vee Releases(a,f,t2)]] \wedge \neg canceled(a,f,t2,t3)$	EC3
$\neg\ HoldsAt(f,t) \leftarrow Initially_N(f) \wedge \neg\ Declipped(0,f,t)$	EC4
$\neg\ HoldsAt(f,t3) \leftarrow Happens(a,t1,t2) \wedge Terminates(a,f,t1) \wedge \neg\ cancelled(a,f,t1,t2) \wedge t2 < t3 \wedge \neg\ Declipped(t1,f,t3)$	EC5
$Declipped(t1,f,t4) \leftrightarrow \exists\ a,t2,t3[Happens(a,t2,t3) \wedge t1 < t3 \wedge t2 < t4 \wedge [Initiates(a,f,t2) \vee Releases(a,f,t2)]] \wedge cancelled(a,f,t2,t3)$	EC6
$Happens(a,t1,t2) \rightarrow t1 \leq t2$	EC7
$cancelled(a1,f,t1,t2) \leftrightarrow happens(a2,t1,t2) \wedge cancels(a2,a1,f)$	EC8
$holdsAt(f2,t3) \leftarrow happens(a,t1,t2) \wedge initiates(a,f1,t1) \wedge \neg cancelled(a,f,t1,t2) \wedge (t2<t3) \wedge (t3=t2+D) \wedge trajectory(f1,t1,f2,D) \wedge \neg\ clipped(t1,f1,t3)$	EC9

O **Cálculo de Eventos** surgiu por uma necessidade de um melhor tratamento dos problemas que envolvem acontecimentos atrelados ao tempo (o que alguns julgavam uma das deficiências do *Cálculo das Situações*), eventos simultâneos e parcialmente ordenados e aos estados locais. A versão do **Cálculo dos Eventos/EC** de Shanahan e Miller ainda leva em conta o uso de mais três ferramentas da Lógica Matemática, que são o *Raciocínio Dedutivo*, a *Não-Monotonicidade* (o **Cálculo de Eventos** original de Kowalski e Sergot já fazia uso do mesmo) e a *Circunscrição*. Todos estes serão vistos mais adiante.

Para deixar mais clara a diferença entre a resolução por **Cálculo de Eventos** e a resolução por *Cálculo das Situações*, usaremos a mesma variante simplificada do exemplo do *Mundo dos Blocos* de Shanahan (SHANAHAN, 1997b), visto anteriormente no Cálculo das Situações, aplicando-se, porém, o **Cálculo de Eventos**

Original (*Original Event Calculus* - OEC) de Kowalski e Sergot (Figura 11). Note que, para dar mais clareza e simplicidade à explanação deste exemplo, não estão apresentados aqui os axiomas genéricos do *Cálculo de Eventos Original* de Kowalski e Sergot, mas os mesmos fazem parte da resolução deste problema, para consultá-los ver (KOWALSKI, 1986).

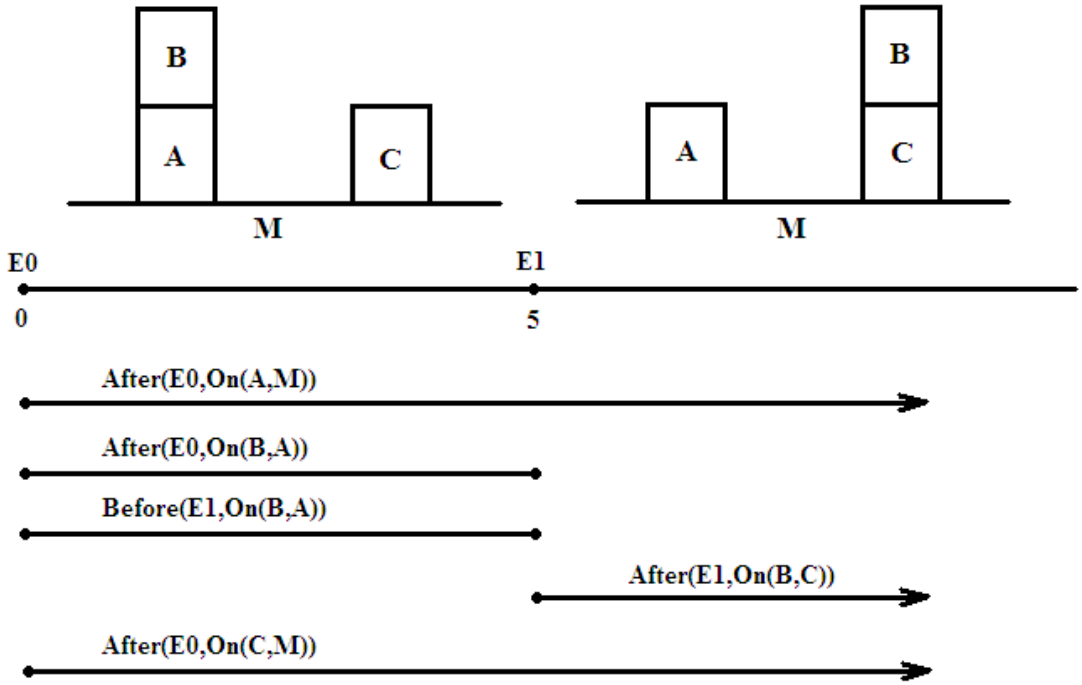


Figura 11 – Narrativa do *Mundo dos Blocos*

A diferença pode ser notada na forma como os efeitos dos eventos são descritos. Ao invés do estilo dos axiomas de efeitos do *Cálculo das Situações*, o **Cálculo dos Eventos Original** possui as cláusulas ***Initiates***, ***Terminates*** e ***Incompatible***. Segundo (SHANAHAN, 1997b), o trabalho original de Kowalski e Sergot possui uma técnica de representação baseada em redes semânticas (KOWALSKI, 1979), na qual as descrições dos eventos são decompostas em relações binárias. A fórmula abaixo mostra como esta técnica se aplica ao *Mundo dos Blocos*.

$$\text{Initiates}(e,\text{On}(x,y)) \leftarrow \text{Act}(e,\text{Move}) \wedge \text{Object}(e,x) \wedge \text{Destination}(e,y) \wedge$$
$$\text{Time}(e,t) \wedge \text{HoldsAt}(\text{Clear}(x),t) \wedge \text{HoldsAt}(\text{Clear}(y),t) \wedge \quad (12)$$
$$\text{Diff}(x,y) \wedge \text{Diff}(x,M)$$

A formula (12) é equivalente à fórmula (8) do *Cálculo das Situações*, sendo que o predicado ***Diff(x,y)*** representa **x ≠ y** (**x** é diferente de **y**).

As precondições aparecem como fórmulas atômicas do predicado ***HoldsAt*** no corpo das clausulas, as ocorrências de eventos são descritas em termos dos predicados ***Act***, ***Object*** e ***Destination***. Cada evento ou ocorrência possui uma única denominação (*unique name*). A descrição da narrativa do bloco B sendo movido de cima do bloco A para cima do bloco C no tempo E1 pode ser representada como:

$$\text{Act}(E1,\text{Move}) \tag{13}$$

$$\text{Object}(E1,B) \tag{14}$$

$$\text{Destination}(E1,C) \tag{15}$$

$$\text{Time}(E1,5) \tag{16}$$

Porém, a técnica acima não é muito útil porque não mostra muita diferença entre o *Cálculo das Situações* e o **Cálculo dos Eventos Original**. Uma técnica melhor para descrever essa narrativa do *Mundo dos Blocos* usa uma parametrização do predicado ***Move***, sem o uso dos predicados ***Object*** e ***Destination***:

$$\begin{aligned}\text{Initiates}(e,\text{On}(x,y)) \leftarrow\ & \text{Act}(e,\text{Move}(x,y)) \wedge \text{Time}(e,t) \wedge \\ & \text{HoldsAt}(\text{Clear}(x),t) \wedge \text{HoldsAt}(\text{Clear}(y),t) \wedge \\ & \text{Diff}(x,y) \wedge \text{Diff}(x,M)\end{aligned} \tag{17}$$

$$\begin{aligned}\text{Terminates}(e,\text{On}(x,z)) \leftarrow\ & \text{Act}(e,\text{Move}(x,y)) \wedge \text{Time}(e,t) \wedge \\ & \text{HoldsAt}(\text{Clear}(x),t) \wedge \text{HoldsAt}(\text{Clear}(y),t) \wedge \text{Diff}(x,y) \wedge \\ & \text{HoldsAt}(\text{On}(x,z),t) \wedge \text{Diff}(y,z)\end{aligned} \tag{18}$$

$$\text{Incompatible}(\text{On}(x,y),\text{On}(x,z)) \leftarrow \text{Diff}(y,z) \tag{19}$$

$$\text{Initiates}(e,\text{Clear}(z)) \leftarrow \text{Act}(e,\text{Move}(x,y)) \wedge \text{Time}(e,t) \wedge$$

$$\text{HoldsAt}(\text{Clear}(x),t) \wedge \text{HoldsAt}(\text{Clear}(y),t) \wedge \text{Diff}(x,y) \wedge \text{HoldsAt}(\text{On}(x,y),t) \wedge \text{Diff}(y,z) \quad (20)$$

$$\text{Terminates}(e,\text{Clear}(y)) \leftarrow \text{Act}(e,\text{Move}(x,y)) \wedge \text{Time}(e,t) \wedge \text{HoldsAt}(\text{Clear}(x),t) \wedge \text{HoldsAt}(\text{Clear}(y),t) \wedge \text{Diff}(x,y) \wedge \text{Diff}(x,M) \quad (21)$$

$$\text{Incompatible}(\text{Clear}(y),\text{On}(x,y)) \quad (22)$$

Agora a descrição da narrativa torna-se bem mais simples, sem os predicados *Object* e *Destination*:

$$\text{Act}(E1,\text{Move}(B,C)) \quad (23)$$

$$\text{Time}(E1,5) \quad (24)$$

A situação inicial E0 do *Mundo dos Blocos* deste exemplo pode ser descrita da forma abaixo:

$$\text{Initiates}(E0,\text{On}(A,M)) \quad (25)$$

$$\text{Initiates}(E0,\text{On}(C,M)) \quad (26)$$

$$\text{Initiates}(E0,\text{On}(B,A)) \quad (27)$$

$$\text{Initiates}(E0,\text{Clear}(B)) \quad (28)$$

$$\text{Initiates}(E0,\text{Clear}(C)) \quad (29)$$

$$\text{Initiates}(E0,\text{Clear}(M)) \quad (30)$$

$$\text{Time}(E0,0) \quad (31)$$

Através do exemplo do mundo dos blocos, notamos que a forma de construção da resolução de um problema usando o **Cálculo de Eventos** pode ser dividida em três partes: A declaração dos *Axiomas Genéricos* (as variantes do **Cálculo de Eventos**, i.e. EC, SEC, OEC, etc.), a declaração dos *Axiomas do Problema* e a *Declaração da Narrativa dos Eventos* que ocorrem durante o desenrolar do problema.

O **Cálculo de Eventos Original** (*Original Event Calculus* - OEC) de Kowalski e Sergot foi utilizado neste exemplo do *Mundo dos Blocos* apenas para fins didáticos. Na prática o mesmo possui deficiências em sua concepção que inviabilizam o seu uso na solução de problemas mais complexos, por isto neste trabalho foi utilizada a variante **Cálculo de Eventos/SEC**, também conhecida como *Simplified Event Calculus* (SEC), proposta inicialmente por Kowalski (KOWALSKI, 1986) e desenvolvida posteriormente por Sadri (SADRI, 1987), Eshghi (ESHGHI, 1988) e Shanahan (SHANAHAN, 1997b), e que se resume, no final, a uma simplificação da variante mais completa *Event Calculus* (EC), já estudada neste capítulo, ou seja, neste trabalho utilizamos os axiomas EC1, EC2 e EC3, comuns tanto ao EC como ao SEC.

O conhecimento visto aqui sobre **Cálculo de Eventos** será suficiente para o leitor compreender, após complementarmos esse conhecimento com noções de *raciocínio indutivo*, *dedutivo*, *abdutivo*, *monotonicidade* e *não-monotonicidade*, *lei da inércia* e *circunscrição*, o desenvolvimento dos algoritmos utilizados neste trabalho.

5.1.1.4. Raciocínio Dedutivo

O **raciocínio dedutivo** (MARCOS, et al., 2005) é uma das três formas canônicas de inferência adotadas para o estabelecimento de hipóteses de ordem científica, as outras duas são o *raciocínio abdutivo* e o *raciocínio indutivo*.

Esta forma canônica de inferência permite inferir as consequências conhecendo-se previamente as causas.

Esquematicamente, escreve-se:

$$\frac{A, A \rightarrow B}{B}$$

Por exemplo, poder-se-ia ter:

A $\equiv_{def}$ Zone1TripA, ocorrência da proteção padronizada 21, a proteção de distância da zona 1, fase A.

B $\equiv_{def}$ AbrirDisjuntor52AX, abertura do disjuntor da linha protegida.

5.1.1.5. Raciocínio Abdutivo

O **raciocínio abdutivo** (MARCOS, et al., 2005) é uma forma canônica de inferência que permite inferir, a partir das consequências, as prováveis causas sem o conhecimento prévio das mesmas. Ou seja, a partir de um acontecimento se formulam hipóteses (a serem provadas) que levem a uma evidência (a causa) para aquele acontecimento.

Esquematicamente, escreve-se:

$$\frac{B, A \rightarrow B}{A}$$

5.1.1.6. Raciocínio Indutivo

O **raciocínio indutivo** (MARCOS, et al., 2005) é uma forma canônica de inferência que permite inferir uma regra universal a partir de premissas particulares. Porém, não se pretende que essas premissas forneçam provas cabais da veracidade da conclusão.

Esquematicamente, escreve-se:

$$\frac{A, B}{A \rightarrow B}$$

A Figura 12 ilustra a diferença entre estas três formas de raciocínio, quando aplicada ao *Cálculo de Eventos*:

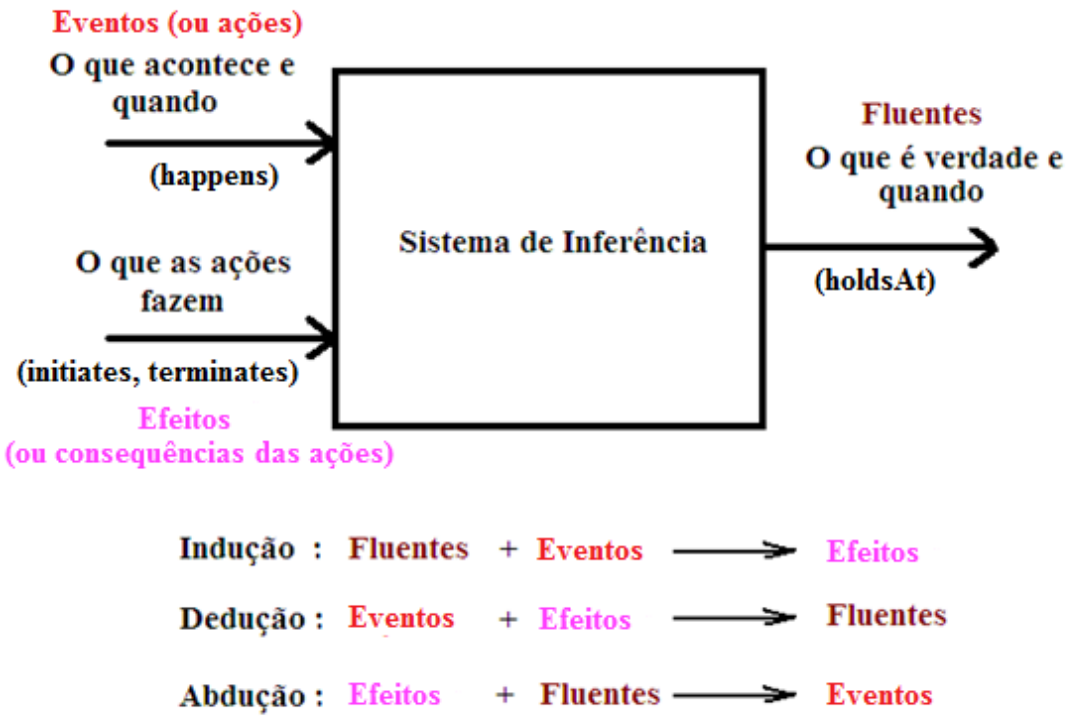


Figura 12 – As diferenças entre os raciocínios Indutivo, Dedutivo e Abdutivo

5.1.1.7. Monotonicidade e Não-Monotonicidade

Monotonicidade (SILVESTRE, 2008) é uma propriedade lógica que diz que adicionando-se uma premissa a um conjunto de outras premissas, não alteram-se as conclusões já existentes sobre estas premissas. Para ficar mais claro, digamos que temos uma linguagem **L**, desta linguagem nos também temos um conjunto de enunciados (ou premissas) **Γ** ou (**$\Gamma \subseteq L$**). Digamos também que temos um enunciado **α** que pertence a **L** (**$\alpha \in L$**). Se tomarmos os elementos de **Γ** como premissas e se tomarmos **α** como sendo a conclusão dessas premissas, a consequência lógica (|=) nos dirá se o argumento $< \Gamma , \alpha >$ é válido ou não. Caso o seja, diremos que $\Gamma \models \alpha$ (**α é consequência lógica de Γ** ou **α** pode ser deduzido a partir de **Γ**), caso contrário diremos que $\Gamma \not\models \alpha$.

Se tivermos uma premissa $\beta \in L$ e a mesma for adicionada a **Γ**, se a consequência lógica entre **Γ** e **α** não for alterada, i.e $\Gamma \cup \beta \models \alpha$, chamaremos esta propriedade de

monotonicidade. A lógica clássica e dita monotônica porque o conjunto de premissas de uma teoria é sempre um subconjunto de qualquer extensão das premissas dessa teoria.

Porém, muitas vezes necessitamos estender o raciocínio lógico de primeira ordem, admitindo que inferências sejam realizadas na ausência de informações em contrário, podendo estas inferências serem invalidadas por novas informações. Esta propriedade é chamada de **não-monotonicidade**.

Um exemplo clássico de não-monotonicidade é o exemplo do pássaro Tweety (SHANAHAN, 1997a; PIGARI, et al., 2012), onde pela regra da monotonicidade, devemos concluir que Tweety voa (pois não há regras em contrário que digam que um pássaro não voa). Ou seja:

$\mathbf{\Gamma}$ = Premissas do que são os pássaros e do porque eles voam.

β = Premissa que Tweety é um pássaro.

α = Conclusão que Tweety voa.

Assim: $\mathbf{\Gamma} \cup \beta \models \alpha$

Porém, ao descobrirmos que Tweety é um pinguim, e sabendo que pinguins não voam, teríamos que, $\mathbf{\Gamma} \cup \beta \not\models \alpha$, o que é um contrassenso, pois se pinguins são pássaros, e se pássaros voam, então pinguins deveriam voar.

Para contornar este problema deveremos adicionar uma nova regra (premissa) às premissas $\mathbf{\Gamma}$, formando a nova premissa $\mathbf{\Gamma'}$, que diz “pinguins são pássaros, porém pinguins não voam”, assim com a nova premissa, termos $\mathbf{\Gamma'} \cup \beta \models \neg\alpha$ ou $\mathbf{\Gamma'} \cup \beta \not\models \alpha$, ou seja, Tweety é um pássaro que não voa (um pinguim).

Isto ilustra bem o conceito de **não-monotonicidade**, que é utilizado tanto no **Cálculo de Eventos** como também na conceituação de **Circunscrição**.

5.1.1.8. Lei da Inércia do Senso Comum

Uma das grandes dificuldades do *Frame Problem* pode ser verificada quando nos deparamos com problemas que envolvam, por exemplo, um universo muito grande de situações, eventos, fluentes e ações. No momento de se fazer o tratamento das

consequências de uma ação, vem a pergunta: " – O que foi realmente afetado e o que não foi afetado? Quais os fluentes que sofreram a ação? ".

A resposta para isso está no estudo do *Default Reasoning* (Raciocínio Padrão), que pode ser entendido como a forma pela qual será conduzido o raciocínio lógico para se chegar à solução mais eficiente para o *Frame Problem*.

Numa tentativa de direcionar uma solução para este problema, McCarthy, em um trabalho posterior (Circumscription – A Form of Non-Monotonic Reasoning, 1980) *apud* (SHANAHAN, 1997b) em que apresenta o conceito de *Circunscrição*, formulou o conceito da **Lei da Inércia do Senso Comum**, que pode ser resumida desta forma:

"Dada uma ação e um fluente, quase sempre a realização da ação não afetará o fluente"

O significado desta lei é que a MUDANÇA é uma excepcionalidade, logo a INÉRCIA (não mudança) é o que devemos normalmente esperar (pelo senso comum do raciocínio padrão) como o resultado de uma ação. Devido a isto, essa lei é conhecida como a **Lei da Inércia do Senso Comum** ou simplesmente **Lei da Inércia**.

A **Lei da Inércia do Senso Comum** é útil para um direcionamento da solução do *Frame Problem*, ao eliminar um universo grande de fluentes da análise de um problema. Porém, a mesma, por si só, não é suficiente, pois muitas vezes não basta apenas verificarmos as consequências das ações sobre aquilo que foi afetado, porque os fluentes afetados podem não ter significância para a solução do problema e então devemos restringir o universo de fluentes afetados apenas a uma vizinhança de interesse ao problema. A esta restrição da vizinhança do problema nos chama-se de *Circunscrição* e a conceituação sobre a mesma será vista na seção seguinte.

5.1.1.9. Circunscrição

A **Circunscrição** é uma forma de *Default Reasoning* (Raciocínio Default) que foi informalmente descrita por McCarthy em 1977 num artigo para o Proceedings IJCA 77 (McCARTHY, 1977) *apud* (SHANAHAN, 1997b) e, mais tarde, em 1980,

detalhadamente num artigo para a revista *Artificial Intelligence* (Circumscription – A Form of Non-Monotonic Reasoning, 1980) *apud* (SHANAHAN, 1997b).

A **Circunscrição** consiste numa limitação do conjunto de objetos para os quais um predicado é verdadeiro. Este processo é também conhecido como *minimização da extensão do predicado* e utiliza o conceito de **não-monotonicidade**, visto anteriormente, pois se está minimizando um predicado ao não saber o seu valor verdade para determinados argumentos.

Para ilustrar como funciona a **Circunscrição**, suponhamos que Σ seja uma conjunção de fórmulas tal que tenhamos $\Sigma \equiv_{def} pred1(A) \wedge pred1(B) \wedge pred2(C)$.

Podemos deduzir, por exemplo, que $\Sigma \models pred1(A)$, mas não podemos dizer nada sobre $\Sigma \models pred1(C)$, porquê só conhecemos pred2(C), e nem sobre $\Sigma \models \neg pred1(C)$. Para resolver este problema, necessitamos *minimizar* o predicado pred1 de forma a fazer o mesmo ser verdadeiro ***apenas*** para aqueles objetos que Σ forçar serem verdadeiros e nos outros casos fazer pred1 ser falso. Usamos a notação CIRC[Σ ; pred1] para denotar essa *minimização* de pred1 pela circunscrição de Σ.

Assim, teremos $CIRC[\Sigma ; pred1] \models \neg pred1(C)$ assumindo-se que $C \neq A$ e $C \neq B$. De modo mais geral:

$$CIRC[\Sigma ; pred1] \models \forall x [pred1(x) \leftrightarrow [x = A \vee x = B]] \qquad (32)$$

Uma definição mais formal para definição é dada por Shanahan (SHANAHAN, 1997b) e é vista a seguir:

Primeiro, para questões de notação, sejam ρ1 e ρ2 predicados com *aridade* n e seja $\boldsymbol{x}$ a tupla de n variáveis distintas, assim:

- $\rho1 = \rho2$ significa $\forall \boldsymbol{x} [\rho1(\boldsymbol{x}) \leftrightarrow \rho2(\boldsymbol{x})]$
- $\rho1 \leq \rho2$ significa $\forall \boldsymbol{x} [\rho1(\boldsymbol{x}) \rightarrow \rho2(\boldsymbol{x})]$
- $\rho1 < \rho2$ significa $[\rho1 \leq \rho2] \wedge \neg [\rho1 = \rho2]$

Agora, nesta definição de **Circunscrição** dada por Shanahan. Seja ϕ uma formula que menciona o símbolo predicativo ρ. A **Circunscrição** de ϕ que minimiza ρ, escrevendo-se CIRC[ϕ ; ρ], é a fórmula de segunda ordem:

$$\phi \wedge \neg \exists\, q\, [\, \phi(q) \wedge q < \rho\,] \qquad (33)$$

Aonde $\phi(q)$ é a fórmula obtida substituindo-se todas as ocorrências de ρ em ϕ por q.

Essa fórmula também pode ser escrita na forma equivalente:

$$\phi \wedge \forall\, q\, [[\, \phi(q) \wedge q \leq \rho\,] \rightarrow \rho \leq q\,] \qquad (34)$$

Esta forma equivalente é muitas vezes mais fácil de se usar nas demonstrações sobre **Circunscrição**.

Uma definição mais sucinta para **Circunscrição** pode ser resumida desta forma:

"Dada uma sentença F e um predicado p, a circunscrição de F que minimiza p, denotado por CIRC[F,p], é uma fórmula cujos modelos são modelos de F que têm a mínima extensão de p."

A grande vantagem do uso da **Circunscrição** no *Frame Problem* é que com a mesma consegue-se limitar mais facilmente o universo dos fluentes afetados na vizinhança de interesse para a solução do problema.

5.1.2. Answer Set Programming (ASP)

PROLOG (PEREIRA, 2009) é uma linguagem de desenvolvimento para aplicações em lógica que já vem sendo utilizada há algumas décadas. O *Cálculo de Eventos* foi desenvolvido originalmente em linguagem PROLOG. Porém, durante o

desenvolvimento deste trabalho, ao tentarmos executar muitos dos exemplos didáticos da literatura de *Cálculo de Eventos* em interpretadores PROLOG, nos deparamos com severas limitações nos mesmos, oriundas das fragilidades da concepção desta linguagem.

No *Cálculo de Eventos*, é bastante comum a ocorrência de chamadas recursivas entre os axiomas que compõem os *axiomas genéricos* e também os *axiomas do problema*. Porém, o uso de recursão em PROLOG provoca muitas vezes a ocorrência de *loops* infinitos e estouros de pilhas durante a execução das soluções dos problemas.

Modificar a ordem dos axiomas e/ou dos predicados dos mesmos pode ajudar a resolver o problema dos *loops* em algumas situações ou para algumas variáveis, porém este método pode se tornar impraticável quando os problemas envolvem um número muito grande de variáveis e de axiomas no problema, o que pode tornar o uso de PROLOG inviável para a busca de soluções usando-se o *Cálculo dos Eventos*.

Uma alternativa é o uso de uma nova linguagem para aplicações em lógica, originada da programação em lógica, chamada de **Answer Set Programming** (ASP) (MAREK, et al., 1999; NIEMELÄ, 1999).

As diferenças principais entre ASP e PROLOG são:

- PROLOG tem como característica a descrição do problema e a sua consulta através de uma cláusula objetivo, explicitadas (implementadas) em axiomas como *Cláusulas de Horn* (CASANOVA, 2008; CASANOVA, 2006), utilizando *refutação LSD* (*Método da Resolução Linear com Função de Seleção para Cláusulas Definidas* (*Resolução-LSD*)) (CASANOVA, 2008), i.e. *SLD Resolution - Selective Linear Definite clause resolution*, a partir da mesma no caso de cláusulas definidas positivas (modelo mínimo de Herbrand). No caso da introdução da negação em PROLOG, utiliza-se a *Negação por Falha Finita* (NFF), i.e. *SLDNF – Selective Linear Definite clause resolution with Negation as Failure* (CASANOVA, 2006).
- ASP, por sua vez, faz o tratamento da negação através de *Circunscrição* e não através de *Negação por Falha Finita* (NFF).
- ASP é também mais flexível do que PROLOG, por poder representar cláusulas de 1ª ordem com muito mais liberdade.

- Os avaliadores (*ASP Solvers*) não buscam se há uma resposta a uma consulta, mas buscam calcular um modelo mínimo, o que evita respostas espúrias.
- Para programas mais gerais, como os admitidos pela linguagem de entrada de ASP, havendo negações no programa, pode não existir um modelo único, mas sim múltiplos modelos.

O funcionamento das ferramentas da linguagem ASP pode ser dividido em dois níveis de arquitetura (THOMAS, 2009):

1. ***Grounding Step* (Passo de Instanciação Básica)**: Dado um programa P com variáveis, o programa subconjunto (*subset*) P' de sua fundamentação (*grounding*) é gerado e possui os mesmos conjuntos de respostas (*answer sets*) que P.
2. ***Model Search* (Pesquisa do Modelo)**: Os conjuntos de respostas (*answer sets*) do programa subconjunto (*subset*) P' proposicional (fundamentado / *grounded*) são computados.

Geralmente, para o Passo de **Instanciação Básica** (*Grounding Step*), são utilizadas ferramentas chamadas **Instanciadores Básicos** (*Grounding Tools*). Podemos citar como exemplos o LPARSE, o GRINGO, entre outros.

Já para a **Pesquisa do Modelo** (*Model Search*), são utilizadas ferramentas chamadas *SAT Solvers*. Podemos citar como exemplos o SMODELS, o CMODELS, o CLASPD, entre outros.

Das implementações de ASP disponíveis, optou-se neste trabalho pelo conjunto do projeto POTASSCO (*the Potsdam Answer Set Solving Collection*) (POSTDAM, 2012), após algumas avaliações de desempenho e devido ao mesmo possuir o conjunto de ferramentas ASP completas já embutidas no pacote e bem documentadas. Dentre estas ferramentas citamos o CLINGO, que é uma ferramenta que já faz o papel de *Grounding Tool e SAT Solver* ao mesmo tempo, pois já inclui internamente as ferramentas GRINGO e CLASPD num mesmo binário, o que torna mais rápida a computação das respostas dos modelos fornecidos.

5.2. Reconhecimento de Crônicas

O **Reconhecimento de Crônicas** é uma forma de representação do conhecimento, inicialmente proposto por (MACKWORTH, 1985) e posteriormente desenvolvido em trabalhos posteriores que chegam até os dias de hoje. Notadamente podemos citar os trabalhos de (DECHTER R., 1991), de (DOUSSON C., 1993) e de (GHALLAB, 1996).

Neste tipo de representação, definimos **crônica** (GHALLAB, 1996) como uma representação temporal da ocorrência de uma quantidade finita de eventos, sujeita a restrições temporais, e com os intervalos de ocorrências de cada evento da crônica determinados por limites mínimos e máximos. A Figura 13 tenta dar uma ideia aproximada de uma crônica e suas instanciações.

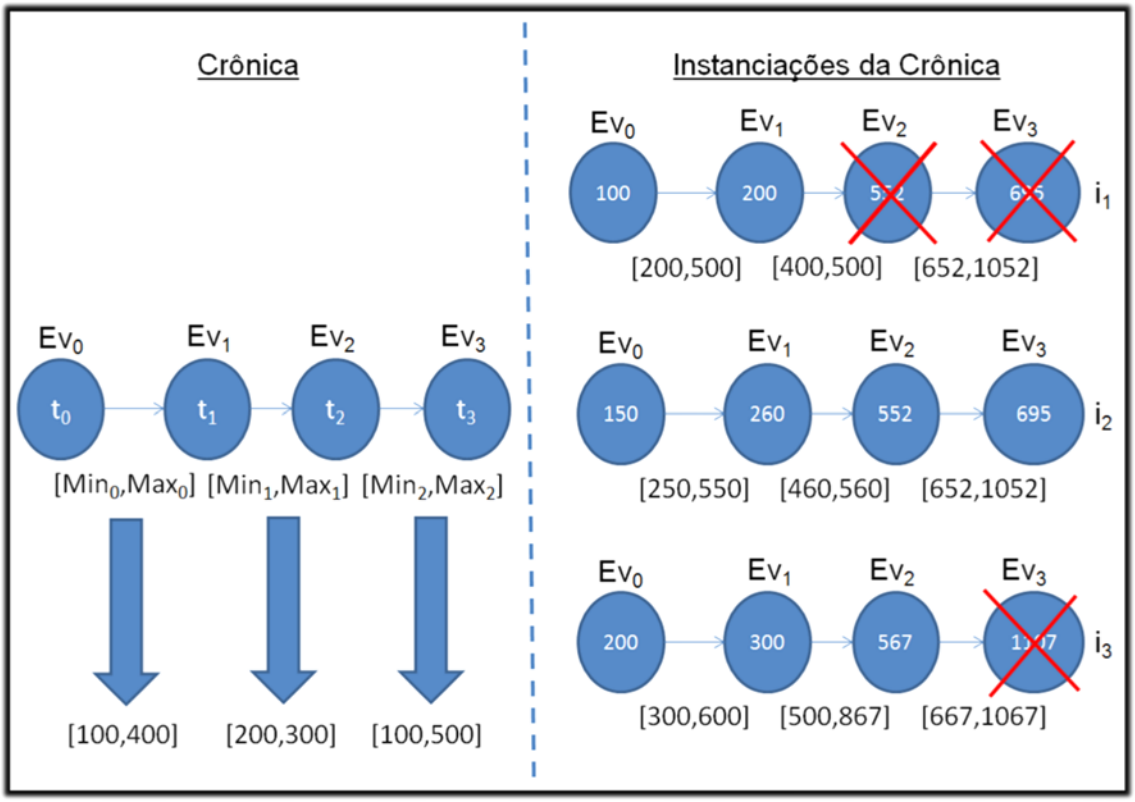


Figura 13 – Exemplo de uma representação de uma crônica e suas instâncias

A Figura 13 mostra três candidatas a instanciações (i_1, i_2 e i_3) de uma crônica. Cada candidata possui a mesma sequência Ev_0, Ev_1, Ev_2 e Ev_3. Os intervalos entre colchetes especificam os limites temporais mínimos e máximos, i.e. $[Mim_x,Max_x]$. A informação que cada evento, i.e. t_0, t_1, etc., carrega é um *timestamp* que indica ano, mês, dia, hora, minuto, segundo e milissegundo da ocorrência do evento, que por simplificação será

representado pelo número de milissegundos passados desde 1º de Janeiro de 1970. A verificação do tempo do evento atual, em milissegundos, para os limites mínimos e máximos especificados se dá pela adição dos mesmos ao *timestamp* do evento anterior, a partir do primeiro evento da crônica, i.e. Ev_0, ou seja, os limites para Ev_1 serão os valores de Min_0 e Max_0 somados ao tempo t_0 daquela instância de Ev_1. Note que os limites mínimos e máximos de ocorrência dos eventos seguintes são fornecidos sempre pelos eventos anteriores, sendo que o último evento da crônica não possui, obviamente, informação de limites para o próximo evento. Note também que todas as instanciações de uma mesma crônica possuem os mesmos limites mínimos e máximos para cada evento seguinte, sendo que o que diferencia uma instanciação de outra é o valor de seus *timestamps*, o que não significa que não possa haver instanciações diferentes com um ou outro evento com o mesmo *timestamp*.

O funcionamento do **Reconhecimento de Crônicas** se dá da seguinte forma, os eventos são monitorados de um *log/stream (histórico)* de entrada. Caso um evento Ev_0 ocorra, o *timestamp* do mesmo é guardado na crônica, criando uma primeira candidata a instância da mesma (i_1). Caso não ocorram novos eventos Ev_0, nenhuma nova instância é criada e a crônica se resume à primeira instância, caso o contrário será criada uma nova instância para cada ocorrência de Ev_0, i.e. i_2, i_3, etc.

Caso ocorra um evento Ev_1, e como no caso de Ev_0 pode ocorrer de 0 a N eventos Ev_1 e o mesmo para os demais eventos da crônica, será verificado se o mesmo ocorreu dentro dos intervalos dos limites mínimos (Min_0) e máximos (Max_0) de alguma instanciação com Ev_1 disponível, respeitada a ordem de criação das instanciações da crônica. Se sim, o *timestamp* do mesmo é guardado na respectiva instanciação. Caso o *timestamp* de Ev_1 ocorra abaixo do limite mínimo de uma dada instanciação, a mesma não é descartada ainda porque podem ocorrer futuramente outros eventos Ev_1 dentro dos limites mínimos e máximos. Caso o *timestamp* de Ev_1 ocorra acima do limite máximo de uma dada candidata a instancia, a mesma é descartada, mesmo que os eventos seguintes ocorram dentro dos intervalos esperados naquela instancia. Mesmo sem a ocorrência de novos eventos no *histórico*, é feita uma verificação contínua para verificar se os limites máximos das instanciações já não foram ultrapassados. Desta forma, as instancias irão sendo descartadas atendam mais as restrições da crônica.

No caso de situações mais complexas e que envolvam mais de uma crônica, o mesmo raciocínio é aplicado, sendo que além da verificação das candidatas a instancias, irá se verificar se os eventos analisados também acarretam a criação de novas candidatas a instancias nestas outras crônicas.

No exemplo da figura anterior, na crônica ocorreram três instanciações que perduraram juntas até a ocorrência de um evento Ev_2 em 552ms que invalidou a instância i_1, mas que estando dentro dos limites para Ev_2 na instância i_2, teve o *timestamp* guardando na mesma. A ocorrência de um novo evento Ev_2, agora em 567ms, foi guardado no Ev_2 da instância i_3, assim na crônica continuaram válidas as instâncias i_2 e i_3. Em 695ms ocorreu um evento Ev_3, que apesar de estar dentro dos limites para a instância i_1, a mesma já fora invalidada anteriormente pelo descarte da primeira ocorrência de Ev_2, logo o evento Ev_3 foi guardado em Ev_3 da instância i_2 por estar dentro dos limites do mesmo. E, finalmente, aos 1107ms ocorreu novamente um evento Ev_3. Só que, por estar fora dos limites de Ev3 da instância i_3, a ocorrência deste evento provocou a invalidação da mesma.

Claro que poderíamos ter também a hipótese deste último Ev_3 ocorrer num *timestamp* mais baixo, por exemplo, aos 703ms, então ai teríamos as crônicas i_2 e i_3 válidas. Qual delas nós poderíamos escolher como o resultado mais preciso da crônica para a solução do problema que a mesma procura resolver? Poderíamos, neste caso, considerar como resposta válida a primeira instanciação a terminar? Seria isto correto?

Isto vai depender do problema que esteja sendo analisado e não é um motivo de preocupação agora, pois se está apenas fazendo o **Reconhecimento de Crônicas** neste momento. Posteriormente, numa segunda etapa, se teria que escolher algum critério para achar a solução ao problema, que poderia ou não levar a uma resposta correta ou errônea, mas que, como dito antes, não é a nossa preocupação neste momento.

Este é um exemplo básico e simplificado, com apenas uma crônica, do funcionamento do **Reconhecimento de Crônicas**. Na prática as análises serão feitas em modelos mais complexos que envolvam mais de uma crônica. A Figura 14 dá uma ideia bem mais realista da complexidade que pode atingir um problema de **Reconhecimento de Crônicas**.

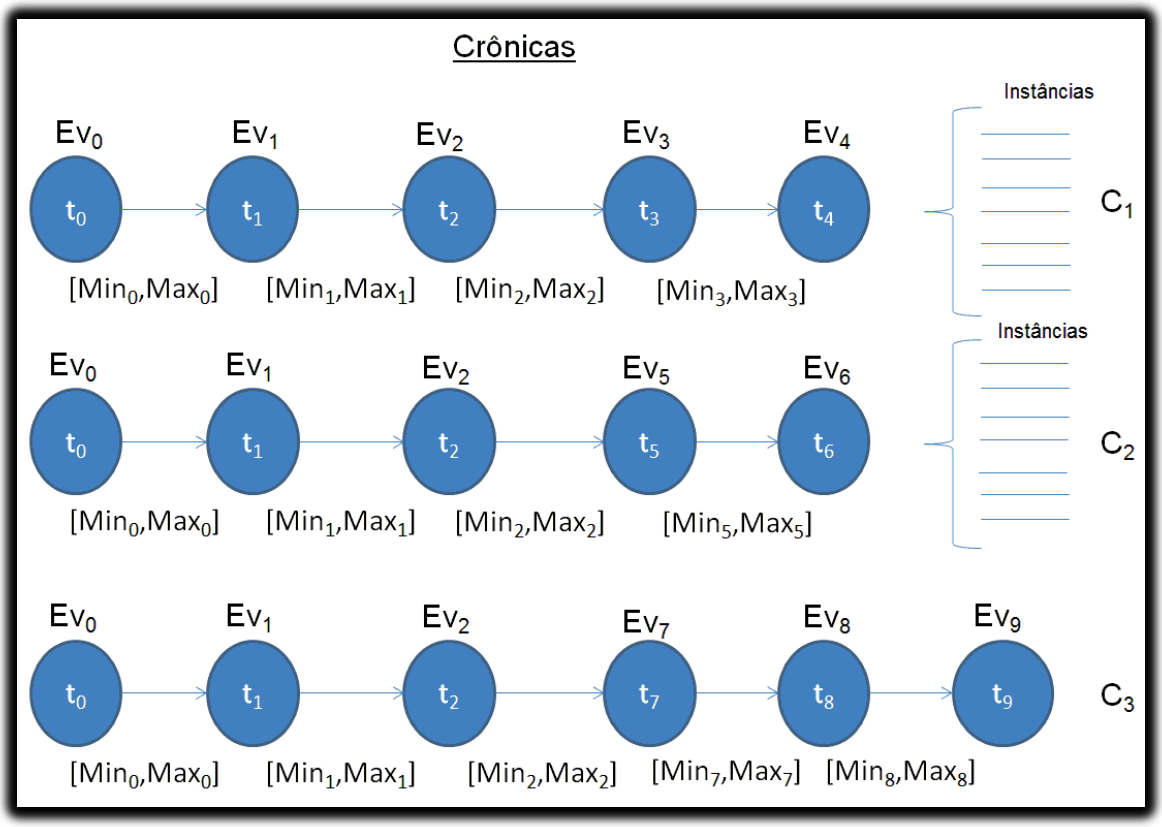


Figura 14 – Exemplo de Reconhecimento de Crônicas mais complexo

A aplicação do **Reconhecimento de Crônicas** na solução do problema-objeto deste trabalho será vista com mais detalhes à frente.

5.3. Processamento de Eventos Baseado em Conhecimento

Na área de Inteligência Artificial (*Artificial Intelligence* - AI), boa parte das aplicações faz uso de modelos computacionais que se baseiam nos conhecimentos empregados por um especialista humano, aplicações essas conhecidas como **Sistemas Especialistas (SE)** (WATERMAN, 1986), (DURKIN, 1994) e (NEGNEVITSKY, 2002). Sistemas especialistas também são chamados de **Sistemas Baseados em Conhecimento (SBC)** que têm como característica resolver problemas ordinariamente resolvidos por um especialista humano (REZENDE, 2003).

Considerando que o *Processamento de Eventos* pode ser feito por um sistema **baseado em conhecimento**, resta entender o funcionamento de um **SBC/SE** para que possamos aplicá-lo à *Processamento de Eventos*.

A descrição a seguir do funcionamento de um **SBC/SE** foi baseado no trabalho de (ZUBEN, 2011), sendo que este se baseou em notas de aulas vinculadas ao livro de (NEGNEVITSKY, 2002).

O **conhecimento**, que é o entendimento prático e teórico de um assunto ou domínio, geralmente necessita de um **especialista**, que é a pessoa que detém a experiência prática de determinado **domínio** e que exerce algum poder, decisório ou preditivo sobre o mesmo.

Esse poder decisório pode ser expresso através de **regras** que representam esse **conhecimento** e que tem uma estrutura da forma *IF* <FACT> *THEN* <CONSEQUENCE> e que relaciona **fatos** (informações na forma de *antecedentes* ou *premissas*) com alguma **ação/resultado** (*consequentes* ou *conclusões*), de forma que sejam de fácil criação e interpretação.

Nessa estrutura, os **antecedentes** e **consequentes** podem ser compostos por conjunções/disjunções de múltiplas proposições que são dadas por objetos linguísticos e seus valores, sendo ligados por operadores que podem ser atribuições tipo SIM e NÃO ou relações como MAIOR e MENOR.

Ao conjunto de uma ou mais **regras** na forma acima definida, denominamos de **base do conhecimento** e estão sempre associadas ao **domínio** do problema.

Geralmente, o escopo de aplicação de **SBC/SE** abrange desde situações em que a **base de conhecimento** é bem caracterizada por um especialista, mas de difícil aprendizado por outros seres humanos, como também indicados para casos em que são tomadas ações repetitivas ou quando os erros humanos devem ser minimizados ou até mesmo evitados. Exemplos de uso vão desde tarefas de diagnóstico e controle de processos até gerenciamento de recursos/atividades produtivas.

Como exemplos de linguagens usadas para o desenvolvimento de **SBC/SE**, podemos citar LISP, OPS5, JESS, DROOLS, PROLOG e ASP.

Para que as **regras** da nossa **base de conhecimento** possam ser processadas, necessitamos implementar uma **máquina de inferência** para a realização do raciocínio

lógico, a função desta é inferir conclusões (**ações/resultados**) a partir dos **fatos** (dados de entrada) e da nossa **base de conhecimento**.

Existem dois tipos de **modelos de inferência** que a **máquina de inferência** pode realizar, que são o **encadeamento direto** (*forwarding chaining*) e o **encadeamento reverso** (*backward chaining*).

No **encadeamento direto**, o raciocínio da **máquina de inferência** é guiado pelos **fatos**, onde se partindo dos **dados de entrada** executa-se a regra mais ativa, causado a adição de um fato novo à **base de dados**, sendo que cada regra só pode ser executada uma única vez, terminando o ciclo de inferência quando não houver mais regras que possam ser ativadas. Normalmente este tipo de encadeamento é indicado quando se parte de **fatos** e o objetivo e inferir **fatos novos**. Este tipo de encadeamento não é eficiente para objetivos previamente estabelecidos (**fato específico**), pois muitas regras podem ser executadas sem que estejam vinculadas de alguma maneira com o fato específico.

No **encadeamento reverso**, o raciocínio da **máquina de inferência** é guiado pelos objetivos, onde se recebendo um **fato**, tem-se como objetivo demonstrar que o mesmo é verdadeiro, logo a **máquina de inferência** busca a veracidade do **fato**, localizando regras que têm o mesmo **fato** como *consequente* e tentando validar o *consequente* de, ao menos, uma dessas regras localizadas. Caso isso não seja possível, são criados *objetivos **(fatos)** intermediários* a serem provados associados ao *antecedente* da regra. Este processo recursivo se encerra quando não existirem mais regras na **base de conhecimento** que tenham como *consequente* o ***objetivo intermediário*** **pretendido** (impossibilidade de comprovar o fato inicial) ou quando todos os ***objetivos intermediários*** e o ***objetivo inicial*** forem atendidos (comprovação do fato inicial).

Dentre as vantagens do uso de **SBC/SE**, podemos citar que emulam a tomada de decisões por especialistas humanos, tendem a ser precisos em suas conclusões e as produzem em curto espaço de tempo, alta capacidade de explanação do processo de inferência e estruturação uniforme com cada regra sendo independente do conhecimento disponível.

5.4. Processamento de Eventos Baseado em Modelo

Em contraposição à abordagem por *Sistemas Baseados em Conhecimento*, vista na seção anterior, existem os **Sistemas Baseados em Modelo (SBM)** (CORDIER, 1998). O termo **modelo** pode ser subentendido com um sinônimo do termo **equação**, tomado num sentido genérico, por exemplo, a equação da lógica de um circuito de proteção.

A abordagem **SBC**, vista na seção anterior, que usa a **base de conhecimento** de um ou mais especialistas, possui vários problemas:

- O custo se colher a **base de conhecimento**, de um ou mais especialistas, é geralmente alto, pois é uma tarefa trabalhosa e que demanda tempo.
- Não há como provar a correção das regras obtidas dos especialistas, o que pode levar ao problema de *conhecimento incompleto*.
- A evolução constante na tecnologia pode tornar difícil a manutenção da **base de conhecimento**.

Logo, o uso apenas da abordagem por *Sistemas Baseados em Conhecimento* não garante o *Processamento de Eventos* correto que desejamos, por isto necessitamos de uma complementação ao mesmo.

O *Cálculo de Eventos* e o *Reconhecimento de Crônicas* são abordagens que podemos utilizar num **SBM**.

O *Cálculo de Eventos* é usado para, a partir das descrições das lógicas de proteção dos relés digitais, gerar um conjunto de crônicas que representem mais aproximadamente o conjunto total de crônicas da lógica de proteção usadas nas SEE.

O *Reconhecimento de Crônicas* é utilizado para, a partir dos *históricos* do SAGE/SCADA, fazer a comparação com as crônicas obtidas e, através do reconhecimento das mesmas ou de suas instâncias, fazer a detecção dos eventos nas SEE.

A utilização desta abordagem trás as seguintes vantagens:

- As crônicas se tornam mais genéricas e passam a representar um conjunto de possíveis eventos.
- As crônicas estão sempre associadas a um subconjunto de configurações dos equipamentos.
- Opcionalmente, numa crônica pode-se expressar a não-ocorrência de um evento.
- Eventos caracterizados como opcionais não necessitam ser reconhecidos para que a crônica seja reconhecida.
- Os eventos detectados não representam os estados dos dispositivos, mas sim a ocorrência temporal e discreta da mudança dos estados dos dispositivos.

Por outro lado, esta abordagem também possui algumas desvantagens:

- Uma crônica necessita da ocorrência de um evento inicial (*trigger event*) para que o reconhecimento da mesma (ou de suas instâncias) seja iniciado, se esse evento ocorrer e não for detectado, a crônica não será reconhecida.
- Após iniciada a crônica ou instância, a falha na detecção de um (ou mais) evento ou o atraso na detecção do mesmo(s) pode causar também o não reconhecimento da mesma, por exemplo, o *histórico* do SAGE/SCADA deixa de informar a ocorrência de um ou mais eventos.
- Ambas as desvantagens anteriores também podem ser classificadas como problema de *conhecimento incompleto*.

Aqui se abre um parêntese para explicar que **conhecimento incompleto** refere-se ao fato de que todo conhecimento que não é explicitamente sabido como verdade deve ser tomado como falso. Porém, o uso de *conhecimento incompleto* pode trazer sérios problemas na busca das soluções, o que pode levar à ocorrência de respostas errôneas.

Neste trabalho, o *conhecimento incompleto* ocorre quando há uma perda de informação no *histórico* do SAGE/SCADE e um ou mais eventos são perdidos e deixam de constar do *histórico*, o que é um acontecimento não raro de ocorrer.

Aplicado o uso de **SBM** ao *Processamento de Eventos*, podemos descrever, usando **ASP/EC**, a lógica de proteção utilizada pelos relés digitais das proteções das SEE para a

obtenção das **crônicas** (**modelos**). A nossa **base de dados**, continua sendo os **eventos** do *histórico* do SAGE/SCADA, que continuam sendo utilizados para o reconhecimento das crônicas.

5.5. Reconhecimento de Crônicas e o Processamento de Eventos Baseado em Modelo

Como visto anteriormente, o sistema SAGE/SCADA gera *históricos* dos eventos de alarmes nos SEE em tempo real. Dentre as mensagens de eventos de alarmes, aquelas que mais interessam aos operadores se referem aos eventos de abertura/falha de disjuntores, de proteções que atuam e de bloqueios que ocorrem. Assim, como podemos ver, não serão todos os eventos de alarmes que necessitam ser analisados, mas, sobretudo, esses três conjuntos de eventos de alarmes.

O problema então se reduz a como proceder à análise desses três conjuntos de eventos de forma a se chegar a uma solução para as causas raízes das falhas nos SEE de forma rápida e segura. Como nas seções anteriores vemos que ambas as abordagens, tanto por **SBC** como por **SBM** possuem vantagens que se complementam e basicamente um problema em comum, o *conhecimento incompleto,* sendo que ambas as abordagens podem ser usadas combinadas para minimizar este último.

Inicialmente, ter-se-á que gerar na fase de processamento *off-line* os modelos previamente para que os mesmos possam ser comparados na fase de processamento *on-line,* evento a evento, em suas instâncias às informações de eventos fornecidas em tempo real pelos *históricos* do SAGE/SCADA. Uma forma de se gerar na fase de processamento *off-line* esses modelos seria pela análise das situações mais comuns de eventos apontados para um determinado SEE, de forma a se extrair daí os modelos necessários à tarefa de reconhecimento, o que é um método puramente **baseado no conhecimento** do especialista (o operador do sistema). Porém, como vimos, esse método poderia (e provavelmente pode) não abranger todos os casos de falhas, de forma que falhas que não tenham ocorrido anteriormente poderão passar despercebidas no momento do reconhecimento na fase de processamento *on-line*. Provavelmente, esse

método geraria uma quantidade absurda de dificuldades que acabaria acarretando problemas de ***conhecimento incompleto***.

Outra forma de se gerar esses modelos é através do uso de *Cálculo de Eventos/SEC*, que previamente seria usado para gerar, na fase de processamento *off-line*, todos os modelos possíveis para as ocorrências de eventos em um determinado SEE. Em contrapartida, a dificuldade deste método é que seria necessária a implementação em ASP, usando SEC, da lógica de proteção dos relés digitais, o que demanda tempo para se implementar manualmente, se não se tiver um *dump* da lógica de proteção em linguagem XML, por exemplo, para se gerar essa implementação através de um programa montador de scripts em ASP que faça o uso de macros para facilitar a descrição de componentes complexos descritos em SEC.

Outra dificuldade inerente ao processamento para o reconhecimento dos modelos é a ocorrência de falhas nos *históricos* do SAGE/SCADA causadas, por exemplo, por falhas na transmissão de dados de um ou mais relés digitais, o que é uma ocorrência até comum de acontecer, de forma que os eventos dessas falhas não sejam registrados ou então sejam registrados num período de tempo muito posterior ao esperado como limite máximo para a ocorrência do evento. Neste caso, pela falta de um evento, por exemplo, o modelo deixaria de ser reconhecido pelo sistema de análise de falhas.

Há alguns métodos que poderia contornar este problema de *conhecimento incompleto*, uma sugestão de um método simples:

- Primeiro, pode-se criar *submodelos* (*subcrônicas*) derivados dos modelos gerados na fase de processamento *off-line* por *Cálculo de Eventos/SEC*, isto simularia casos em que haja falhas na comunicação de alguns dos relês digitais e, por conseguinte, falhas nos *históricos* do SAGE/SCADA.
- Segundo, se utilizaria esses *submodelos* como modelos para o reconhecimento das crônicas, feito na fase de processamento *on-line*, ou suas instanciações, de forma que para cada falha ou conjunto de falhas haja a possibilidade de um reconhecimento do modelo e uma solução para a(s) causa(s) raiz(raízes) do problema.
- Uso da *expertise* dos operadores do sistema, que com sua "*base de conhecimento*" poderiam ajudar na geração destes *submodelos*, tornaria mais

preciso esse método, ainda que não elimine totalmente o problema da perda de eventos.

Este método, apesar de simplificado, poderia ajudar a chegar-se a uma resposta mais próxima do possível da ocorrência da falha real, mas ainda demanda um estudo mais aprofundado. Neste trabalho, utilizamos o método acima para o problema do *conhecimento incompleto*, quando no capitulo seguinte discorreremos da implementação do circuito simplificado para exemplificar o uso do algoritmo na implementação do *Processamento de Eventos* na lógica da proteção de um subestação.

6. Implementação do Processamento de Eventos na Lógica de Proteção de uma Subestação

Este capítulo mostra o desenvolvimento de um algoritmo para o Processamento de Eventos da lógica de proteção da seção de uma subestação. O esquema de proteção elétrico utilizado neste trabalho, mostrado no capítulo 3, foi uma simplificação feita com base na descrição lógica do sistema de proteção da SUBESTAÇÃO BARRA DOS COQUEITOS 230kV (AREVA-PROEL, 2009).

Para se ter uma visão geral das etapas do processamento de eventos, a Figura 15 mostra um diagrama de blocos com as respectivas etapas de processamento *on-line* e *off-line*.

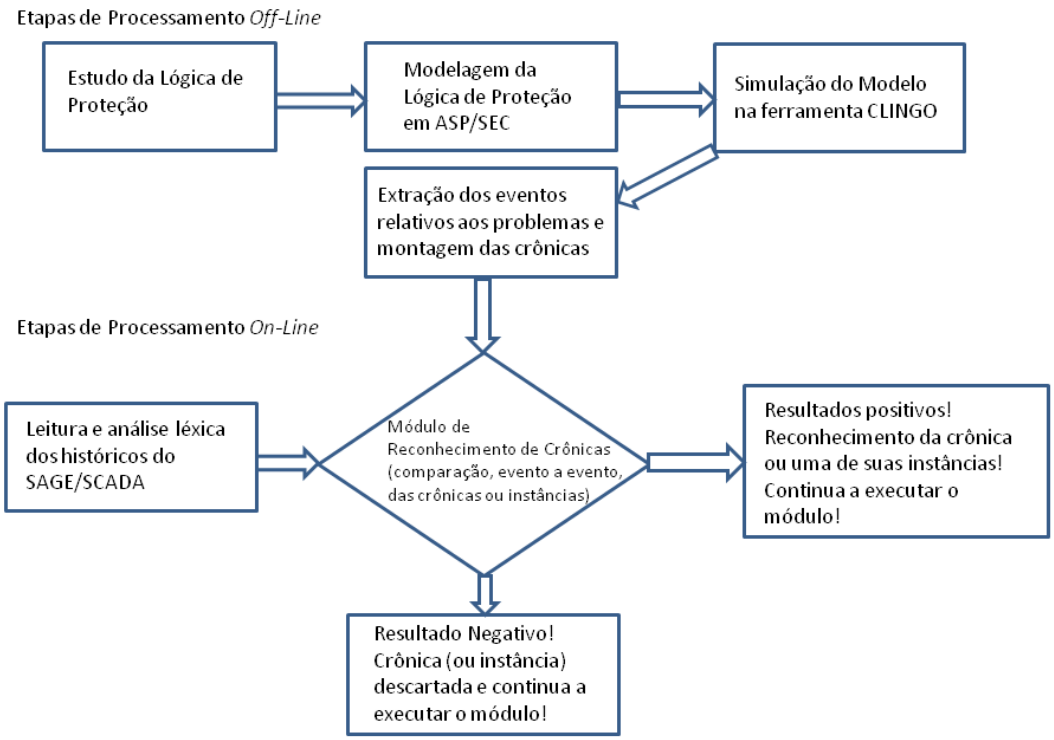


Figura 15 – Etapas de Processamento dos Eventos

As etapas *off-line* correspondem a modelagem da lógica de proteção em Cálculo de Eventos/SEC utilizando linguagem ASP, seguido da simulação e da extração dos eventos de falhas e, finalmente, a montagem das crônicas que serão utilizadas no reconhecimento.

As etapas *on-line* correspondem à execução do módulo de Reconhecimento de Crônicas, que será futuramente desenvolvido para a CAE, que faz a leitura e a análise léxica dos históricos do SAGE/SCADA, em tempo real, e faz a comparação evento-a-evento das ocorrências nos históricos com os eventos das crônicas ou suas instanciações, procurando verificar a ocorrência de uma crônica completa e que represente a falha de proteção que foi simulada na fase *off-line* e que resultou naquela crônica específica.

A seguir, é mostrado como se chegou ao algoritmo que será utilizado futuramente no desenvolvimento do módulo da CAE tomando-se por base o esquema de proteção elétrico já mencionado.

6.1. Implementação dos Diagramas Lógicos em ASP usando Cálculo de Eventos/SEC

Para as duas primeiras etapas do processamento *off-line*, vistas na Figura 16, fez-se a transcrição dos diagramas lógicos de proteção e a descrição em ASP dos mesmos utilizando Cálculo de Eventos/SEC.

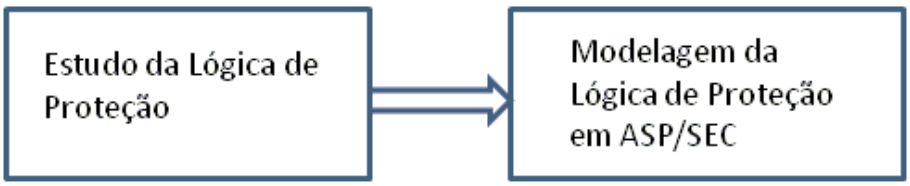


Figura 16 – Etapas E_1 e E_2 do Processamento *Off-line*

As portas lógicas básicas, temporizadores e flip-flops utilizados, e suas descrições em ASP por SEC, seguem nas figuras e descrições a seguir, lembrando que as portas lógicas com mais de 2 entradas foram descritas pela composição de duas ou mais portas lógicas básicas. Por exemplo, portas OR de três entradas foram descritas pela composição de duas portas OR de duas entradas e portas AND de quatro entradas foram

descritas pela composição de três portas AND de duas entradas. Nem todas as portas descritas a seguir foram utilizadas no exemplo do esquema de proteção simplificado utilizado nesta implementação, mas aqui foram colocadas para o caso de aparecerem, futuramente, em algum rele digital de algum fabricante.

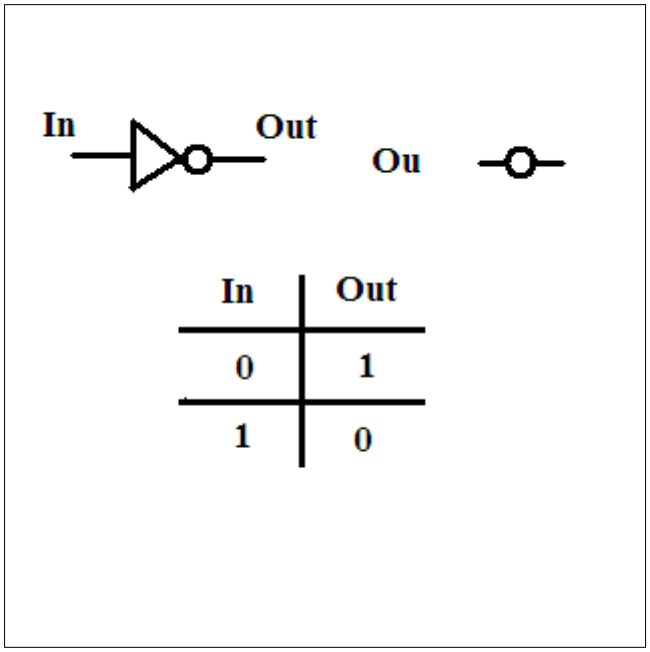
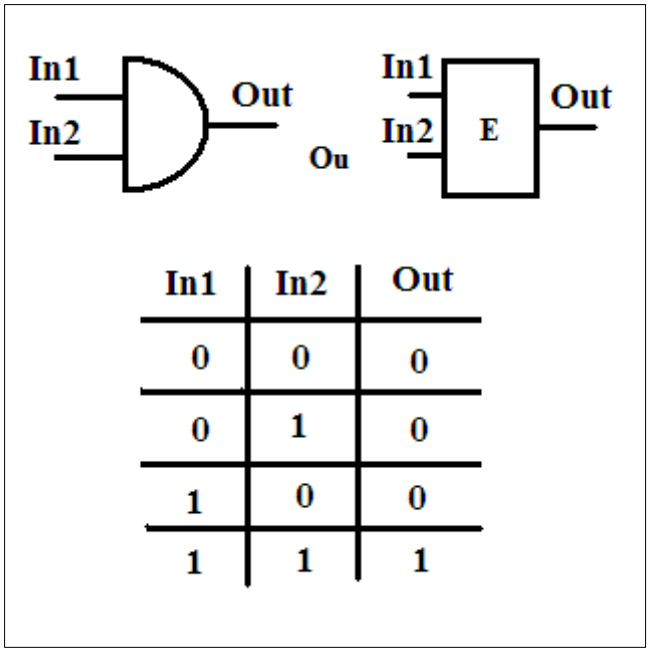


Figura 17 – Esquema e tabela verdade da porta lógica NOT

A descrição em **ASP/SEC** da porta lógica NOT pode ser vista no fragmento de código a seguir:

```
% Definição: Porta NOT (OBS.: Em ASP, not é palavra reservada!)
nott(0,1).
nott(1,0).
```

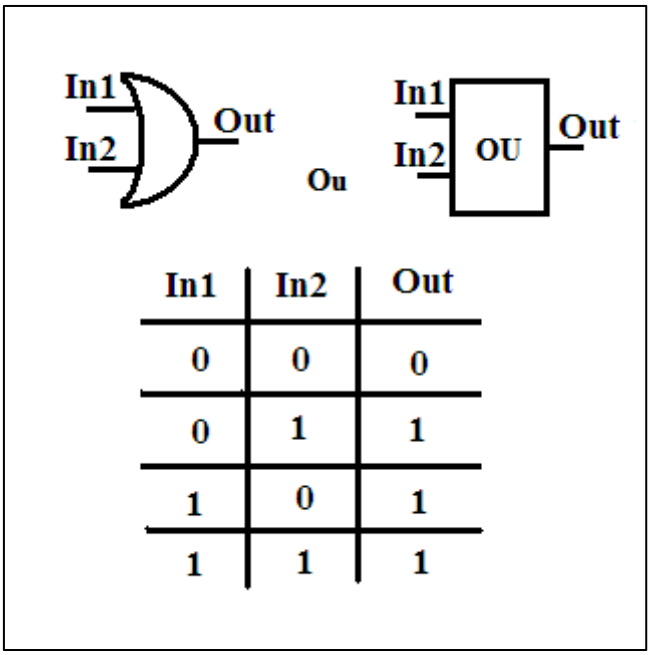


In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Figura 18 – Esquema e tabela verdade da porta lógica AND

A descrição em **ASP/SEC** da porta lógica AND pode ser vista no fragmento de código a seguir:

```
% Definição: Porta AND
and(0,0,0).
and(0,1,0).
and(1,0,0).
and(1,1,1).
```

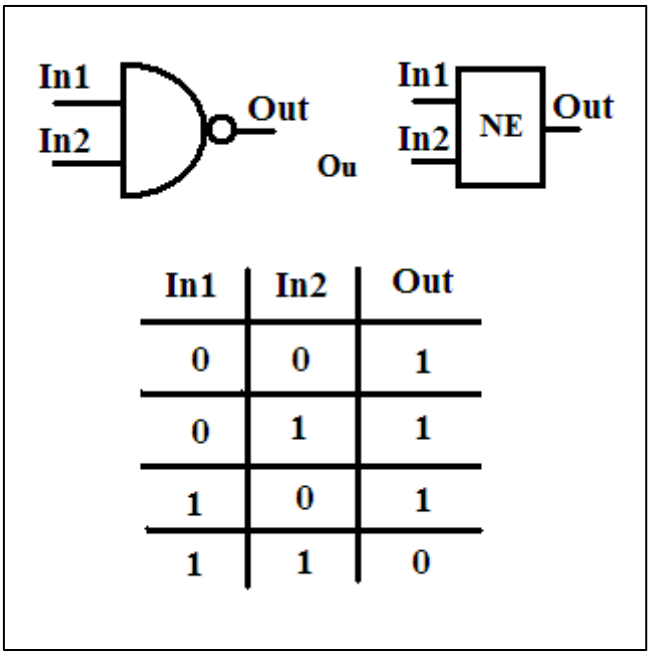


In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	1

Figura 19 – Esquema e tabela verdade da porta lógica OR

A descrição em **ASP/SEC** da porta lógica OR pode ser vista no fragmento de código a seguir:

```
% Definição: Porta OR
or(0,0,0).
or(0,1,1).
or(1,0,1).
or(1,1,1).
```

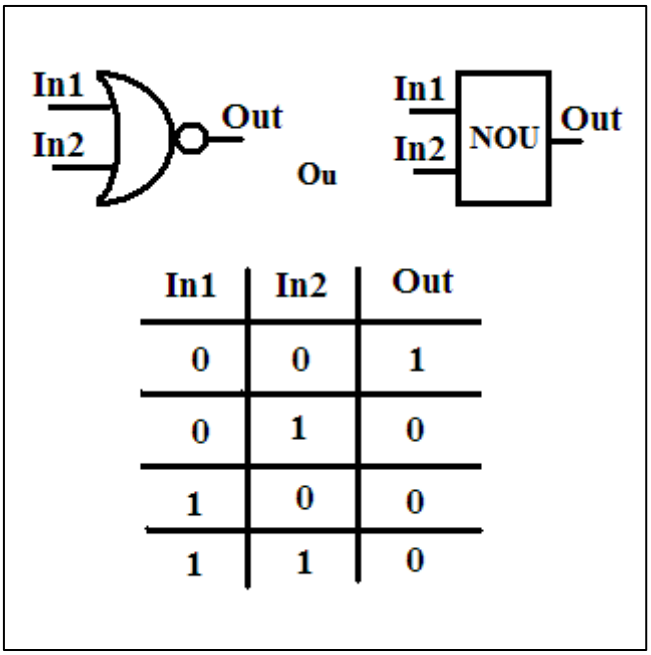


In1	In2	Out
0	0	1
0	1	1
1	0	1
1	1	0

Figura 20 – Esquema e tabela verdade da porta lógica NAND

A descrição em **ASP/SEC** da porta lógica NAND pode ser vista no fragmento de código a seguir:

```
% Definição: Porta básica NAND
nand(0,0,1).
nand(0,1,1).
nand(1,0,1).
nand(1,1,0).
```

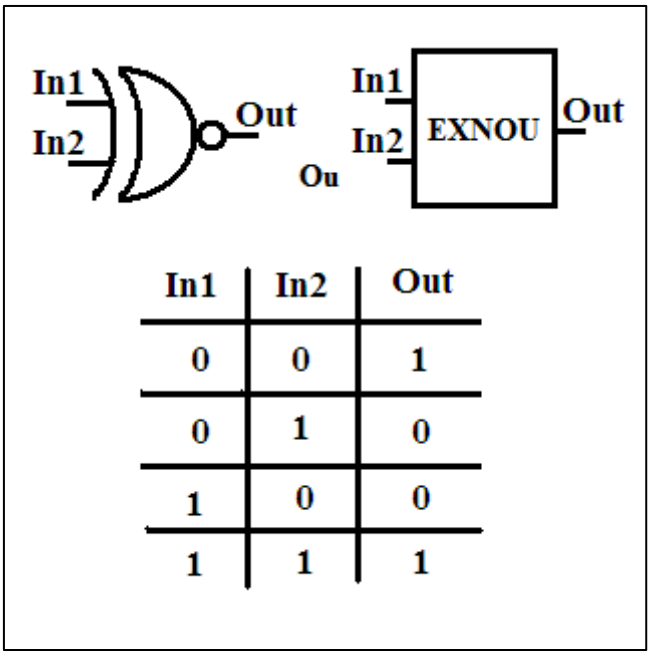


In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	0

Figura 21 – Esquema e tabela verdade da porta lógica NOR

A descrição em **ASP/SEC** da porta lógica NOR pode ser vista no fragmento de código a seguir:

```
% Definição: Porta NOR
nor(0,0,1).
nor(0,1,0).
nor(1,0,0).
nor(1,1,0).
```

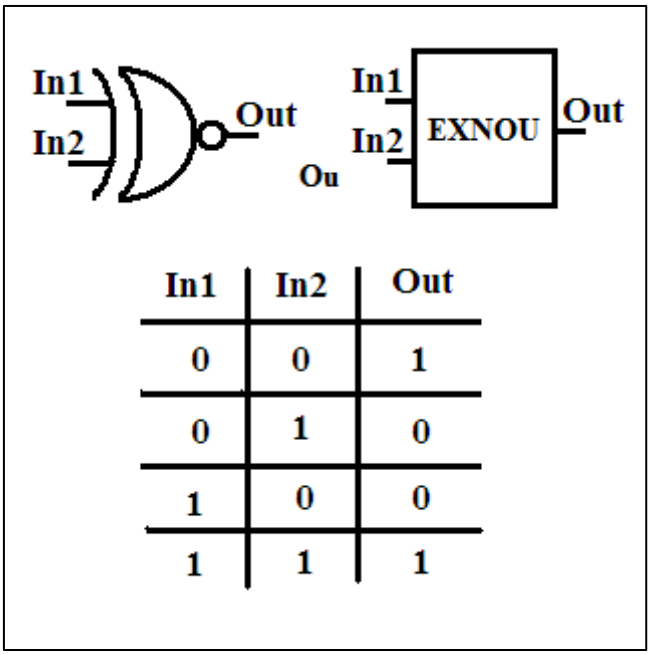


In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	1

Figura 22 – Esquema e tabela verdade da porta lógica XOR

A descrição em **ASP/SEC** da porta lógica XOR pode ser vista no fragmento de código a seguir:

```
% Definição: Porta XOR
xor(0,0,0).
xor(0,1,1).
xor(1,0,1).
xor(1,1,0).
```

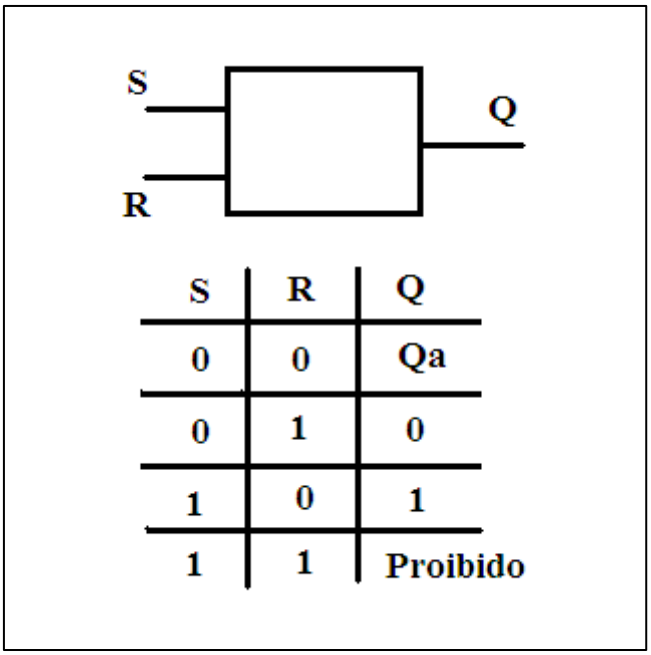


In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	1

Figura 23 – Esquema e tabela verdade da porta lógica XNOR

A descrição em **ASP/SEC** da porta lógica XNOR pode ser vista no fragmento de código a seguir:

```
% definição: Porta XNOR
xnor(0,0,1).
xnor(0,1,0).
xnor(1,0,0).
xnor(1,1,1).
```



S	R	Q
0	0	Qa
0	1	0
1	0	1
1	1	Proibido

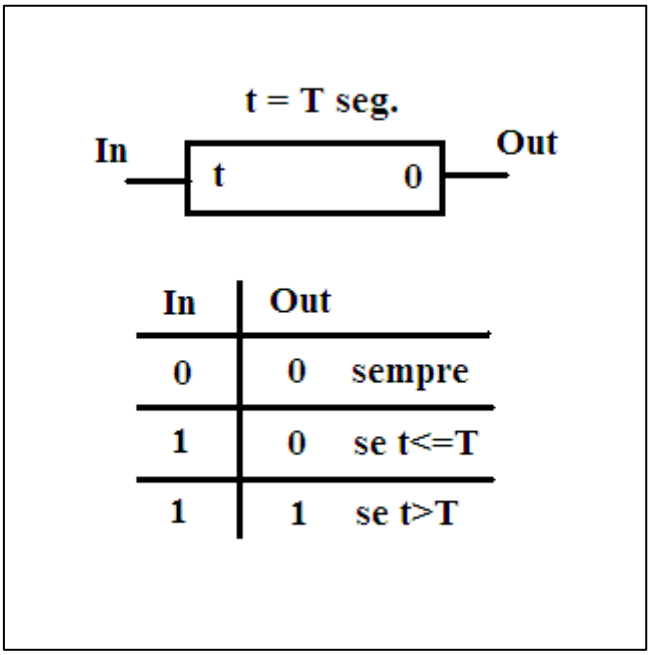
Figura 24 – Esquema e tabela verdade do Flip-Flop SR (Set-Reset)

A descrição genérica em **ASP/SEC** de um Flip-Flop SR pode ser vista no fragmento de código a seguir:

NOTA: Verificou-se que na implementação do Flip-Flop SR dos relés digitais da AREVA, como a implementação não é feita por *hardware* mas por *software*, o estado S=1 e R=1 é tratado como um *reset* e o valor de Q=0 neste caso! Logo o código abaixo tem um adendo prevendo isto!

```
%=== gerador dos sinais de entrada

initially(sin(0)).
initially(rin(0)).

terminates(i1_sup,sin(0),T):-timepoint(T).
 initiates(i1_sup,sin(1),T):-timepoint(T).
terminates(i1_sdw,sin(1),T):-timepoint(T).
 initiates(i1_sdw,sin(0),T):-timepoint(T).

terminates(i1_rup,rin(0),T):-timepoint(T).
 initiates(i1_rup,rin(1),T):-timepoint(T).
terminates(i1_rdw,rin(1),T):-timepoint(T).
 initiates(i1_rdw,rin(0),T):-timepoint(T).

%=== Flip-flop SR
initially(qout(0)).
happens(i2_sup,T1):-holdsAt(sin(0),T1), holdsAt(sin(1),T2),
T2=T1+1,timepoint(T1), timepoint(T2).
happens(i2_sdw,T1):-holdsAt(sin(1),T1), holdsAt(sin(0),T2), T2=T1+1,
timepoint(T1), timepoint(T2).
happens(i2_rup,T1):-holdsAt(rin(0),T1), holdsAt(rin(1),T2), T2=T1+1,
timepoint(T1), timepoint(T2).
```

```
happens(i2_rdw,T1):-holdsAt(rin(1),T1), holdsAt(rin(0),T2), T2=T1+1,
timepoint(T1), timepoint(T2).

terminates(i2_sup,qout(0),T) :- holdsAt(rin(0),T), holdsAt(qout(0),T),
timepoint(T).
 initiates(i2_sup,qout(1),T) :- holdsAt(rin(0),T), holdsAt(qout(0),T),
timepoint(T).
terminates(i2_rup,qout(1),T) :- holdsAt(sin(0),T), holdsAt(qout(1),T),
timepoint(T).
 initiates(i2_rup,qout(0),T) :- holdsAt(sin(0),T), holdsAt(qout(1),T),
timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema do rele
digital da AREVA
terminates(i2_rup,qout(1),T) :- holdsAt(sin(1),T), holdsAt(qout(1),T),
timepoint(T).
 initiates(i2_rup,qout(0),T) :- holdsAt(sin(1),T), holdsAt(qout(1),T),
timepoint(T).
```



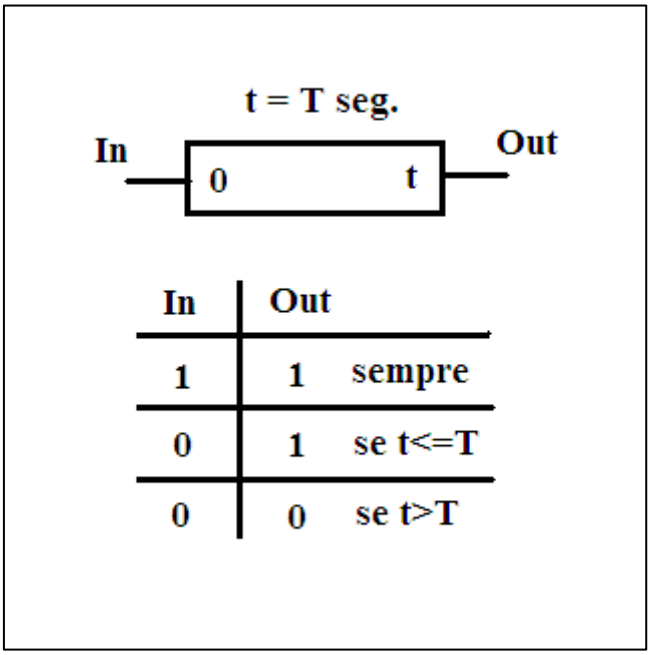
In	Out	
0	0	sempre
1	0	se t<=T
1	1	se t>T

Figura 25 – Esquema e tabela verdade do temporizador PickUp

A descrição genérica em **ASP/SEC** do temporizador PickUp (de 2 segundos) pode ser vista no fragmento de código a seguir:

```
%============ Input signal ======================
initially(inputIs(0)).
terminates(evInputTo1,inputIs(0),T) :- timepoint(T).
 initiates(evInputTo1,inputIs(1),T) :- timepoint(T).
terminates(evInputTo0,inputIs(1),T) :- timepoint(T).
 initiates(evInputTo0,inputIs(0),T) :- timepoint(T).

%============ Pick-up timer de 6 segundos ========
initially(monoOutput(0)).
```

```
happens(upTriger,T3) :- holdsAt(inputIs(0),T1),
holdsAt(inputIs(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger,monoOutput(0),T) :- happens(upTriger,T),
holdsAt(inputIs(1),T), timepoint(T).
initiates(upTriger,monoOutput(1),T) :- happens(upTriger,T),
holdsAt(inputIs(1),T), timepoint(T).

happens(downTriger,0) :- initially(inputIs(0)).
happens(downTriger,T1) :- holdsAt(inputIs(1),T1),
holdsAt(inputIs(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger,monoOutput(1),T) :- happens(downTriger,T),
holdsAt(monoOutput(1),T), timepoint(T).
initiates(downTriger,monoOutput(0),T) :- happens(downTriger,T),
holdsAt(monoOutput(1),T), timepoint(T).
```



In	Out	
1	1	sempre
0	1	se t<=T
0	0	se t>T

Figura 26 – Esquema e tabela verdade do temporizador DropOut

A descrição genérica em **ASP/SEC** do temporizador DropOut (de 30 segundos) pode ser vista no fragmento de código a seguir:

```
%============ Input signal ======================
initially(inputIs(0)).
terminates(evInputTo1,inputIs(0),T) :- timepoint(T).
 initiates(evInputTo1,inputIs(1),T) :- timepoint(T).
terminates(evInputTo0,inputIs(1),T) :- timepoint(T).
 initiates(evInputTo0,inputIs(0),T) :- timepoint(T).

%============ Drop-out timer 6 segundos ===================
initially(monoOutput(0)).
happens(upTriger,0) :- initially(inputIs(1)).
happens(upTriger,T2) :- holdsAt(inputIs(0),T1),
holdsAt(inputIs(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(upTriger,monoOutput(0),T) :- happens(upTriger,T),
holdsAt(inputIs(1),T), timepoint(T).
```

```
initiates(upTriger,monoOutput(1),T) :- happens(upTriger,T),
holdsAt(inputIs(1),T), timepoint(T).

happens(downTriger,T3) :- holdsAt(inputIs(1),T1),
holdsAt(inputIs(0),T2), T2=T1+1, T3=T1+30, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(downTriger,monoOutput(1),T) :- happens(downTriger,T),
holdsAt(monoOutput(1),T), timepoint(T).
initiates(downTriger,monoOutput(0),T) :- happens(downTriger,T),
holdsAt(monoOutput(1),T), timepoint(T).
```

A implementação completa em ASP, usando-se SEC, dos diagramas lógicos interligados, formando assim a lógica de proteção completa do exemplo simplificado da proteção da subestação de 230kV encontra-se no *ANEXO A* deste trabalho.

A utilidade do uso da implementação em SEC nestes circuitos de proteção está na facilidade de se efetuarem diferentes simulações com as mesmas descrições destes circuitos em ASP, modificando-se apenas as ocorrências dos eventos, através dos predicados *happens()*, para se extrair-se corretamente todos os eventos necessários à composição dos diferentes modelos (crônicas) para as situações de diferentes tipos de falhas nos circuitos de proteção das subestações de energia elétrica.

6.2. Extração do Modelo (Crônica)

A extração do modelo (crônica), feito nas duas etapas finais do processamento *off-line*, vistas na Figura 27.

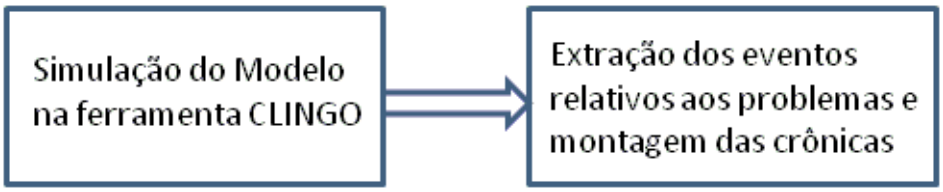


Figura 27 – Etapas E_3 e E_4 do Processamento *Off-line*

Na tabela do *ANEXO B* deste trabalho pode ser visto o *timeline* com os eventos extraídos pela captura dos predicados *happens()* mostrados na saída padrão (*output*) do utilitário CLINGO, utilizado para a execução de implementações em ASP. Este *timeline* também pode ser obtido pela análise do código mostrado no *ANEXO A*.

Ao extrair-se os eventos dos predicados *happens()*, e utilizando a codificação de mensagens do SAGE/SCADA utilizada pela empresa concessionária, teríamos como resultado um *timeline* do qual já se poderia extrair uma aproximação de um modelo (crônica), como pode ser visto a seguir:

06s - SECOQ_CMD_AD52AX - CMD/PERM Abrir DISJUNTOR 52AX

08s - SECOQ_INFOR_AD52AX - Abertura DISJUNTOR 52AX

09s - SECOQ_CMD_SEC89AX1 - Abrir SECIONADORA 89AX1

12s - SECOQ_INFOR_ASEC89AX1 - Abertura SECIONADORA 89AX1

13s - SECOQ_CMD_FSEC89AX3 - Fechar SECIONADORA 89AX3

15s - SECOQ_INFOR_FSEC89AX3 - Fechamento SECIONADORA 89AX3

16s - SECOQ_CMD_FD52AX - CMD/PERM Fechar DISJUNTOR 52AX

17s - SECOQ_EMER_FD52AX - Falha DISJUNTOR 52AX

19s - SECOQ_CMD_SEC89AX2 - Abrir SECIONADORA 89AX2

21s - SECOQ_INFOR_ASEC89AX2 - Abertura SECIONADORA 89AX2

22s - SECOQ_CMD_SEC89AX3 - Abrir SECIONADORA 89AX3

24s - SECOQ_INFOR_ASEC89AX3 - Abertura SECIONADORA 89AX3

Esta listagem ainda não é um modelo (crônica) porque os tempos mostrados estão em segundos (e não em milissegundos) e são absolutos e não limites mínimos e máximos para a ocorrência dos eventos. Finalmente, estimando os limites mínimos e máximos, por exemplo, utilizando-se o conhecimento dos operadores do sistema ou uma fonte bibliográfica (manual) da AREVA, podemos então chegar ao modelo para o caso da falha de abertura do disjuntor da referida subestação de exemplo, que pode ser visto na figura a seguir (obs.: os tempos dos limites mínimos e máximos estão expressos em milissegundos):

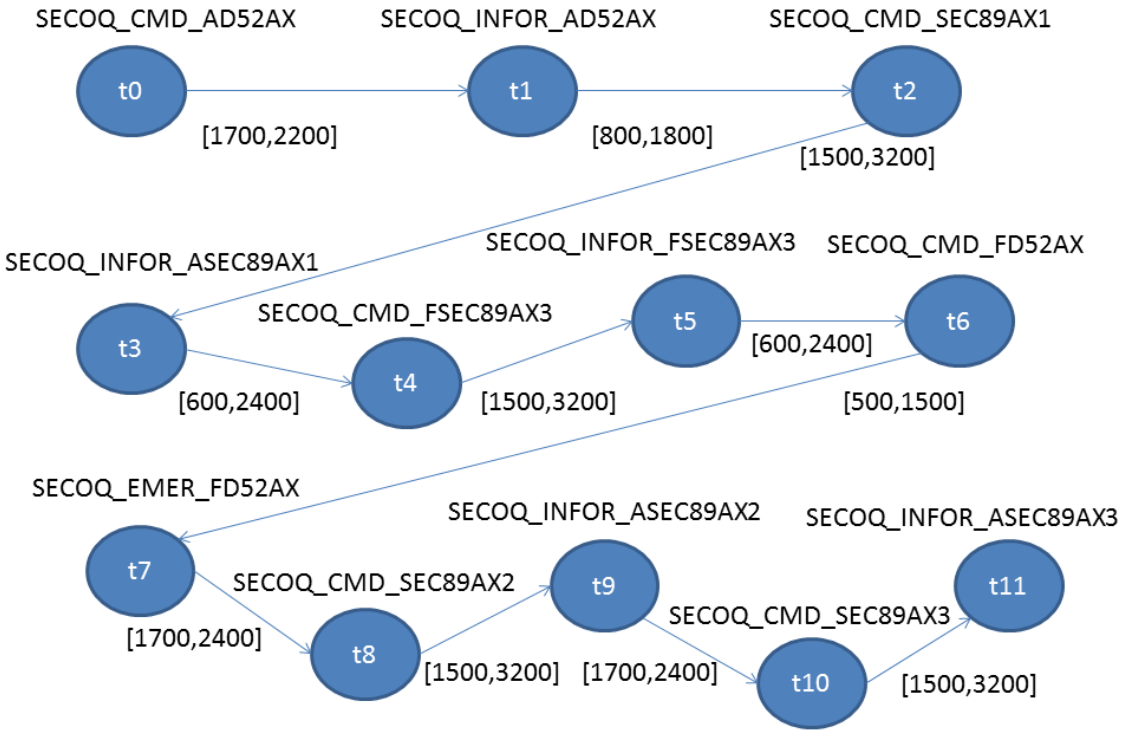


Figura 28 – Modelo (Crônica) para a falha de abertura do disjuntor 52AX da subestação

6.3. Análise dos *Históricos* do SAGE/SCADA usando-se o Modelo Extraído

A análise dos *históricos* do SAGE/SCADA, realizada de forma *on-line*, por *Reconhecimento de Crônicas*, conforme o diagrama de blocos na Figura 29, é feita utilizando-se o modelo das sequências de eventos que caracterizam uma falha de proteção, fazendo-se a checagem dos eventos dos *históricos*, evento a evento, conforme os mesmos vão ocorrendo. Supondo que o *histórico* tenha 48 eventos (1 evento por linha), como mostrado por exemplo no *ANEXO C*, um programa de *Reconhecimento de Crônicas* faz um *parsing* das mensagens do *histórico* do SAGE/SCADA, capturando os eventos e seus *timestamps*, procurando encaixá-los dentro de uma ou mais instâncias do modelo extraído anteriormente.

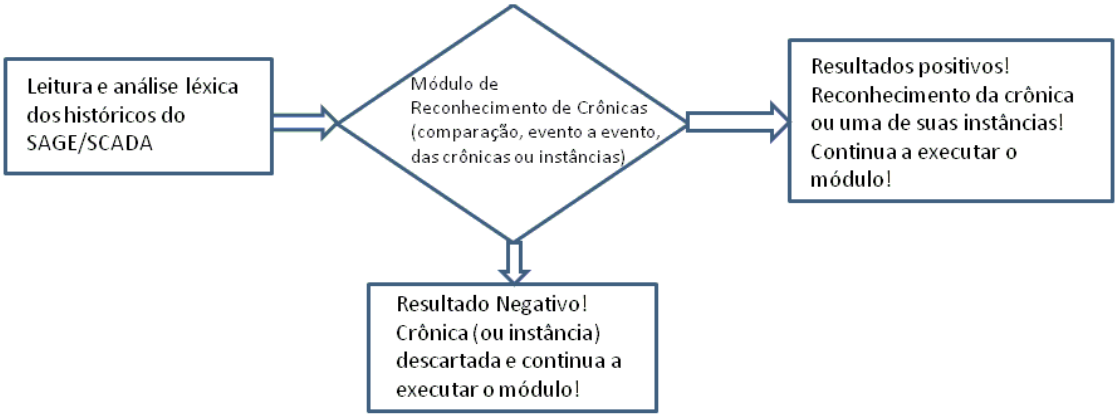


Figura 29 - Etapa E_5 do Processamento *On-line*

Como exemplo de programa para o *Reconhecimento de Crônicas*, um programa experimental foi construído em linguagem Java e pode ser visto nos *ANEXOS D e E*. O mesmo é constituído por uma classe chamada *Cronica*, que se encarrega de criar as instâncias de uma crônica, testar a ocorrência dos primeiros eventos das mesmas e verificar o atendimento das restrições temporais.

A classe principal, *MainFrame*, se encarrega de carregar os *históricos* (logs) dos eventos, carrega as crônicas que serão utilizadas, faz uma listagem das instancias destas crônicas (criadas pela classe *Cronica*) e retorna avisos na tela quanto uma instância de uma crônica é criada pelo reconhecimento do primeiro evento, quando uma instância é descartada pela extrapolação do limite máximo ou mínimo e, finalmente, quando uma instância de uma das crônicas é reconhecida.

Para testar o *Reconhecimento de Crônicas*, utilizamos o modelo extraído para montar duas "*crônicas de teste*", i.e. uma crônica completa (arquivo *Cronica01.cro*) e uma crônica incompleta (arquivo *Cronica02.cro*). Uma terceira "*crônica de teste*" mais simples (arquivo *Cronica03.cro*) foi gerada aleatoriamente para testar o caso do não-reconhecimento de uma crônica. Essas três crônicas utilizadas podem ser vistas no *ANEXO F* deste trabalho. O "*histórico de eventos de teste*" utilizado neste teste, i.e. arquivo *logDeEventos30.log*, que pode ser vista no ANEXO G este trabalho.

Neste exemplo já nos preocupamos com o problema do *conhecimento incompleto*, o que pode ser visto no reconhecimento da crônica incompleta (*Cronica02.cro*), pois queríamos testar também o funcionamento e o desempenho do *Reconhecimento de Crônicas* numa situação comum de perda de uma das mensagens, no caso da mensagem que informa a falha de fechamento do disjuntor 52AX, durante o reconhecimento.

Conforme vimos nas seções anteriores para o tratamento do *conhecimento incompleto*, utilizamos a abordagem vista na secção 5.4, onde do modelo extraído podemos obter *submodelos*, por exemplo, retirando-se o evento *SECOQ_EMER_FD52AX (Falha DISJUNTOR 52AX)*, obtemos um *submodelo* que pode representar o mesmo tipo de problema reportado pelo modelo completo, pois a simples indicação dos eventos *SECOQ_CMD_ASEC89AX2 (Abrir SECIONADORA 89AX2)* e *SECOQ_CMD_ASEC89AX3 (Abrir SECIONADORA 89AX3)* só faria sentido se houver a falha propriamente dita do disjuntor 52AX, pois a secionadora 89AX1 já se encontrava aberta no momento da falha e o sistema de proteção automática, para fazer o isolamento do disjuntor 52AX defeituoso (e já aberto), procedeu à abertura das secionadoras 89AX2 e 89AX3 para permitir a manutenção do disjuntor 52AX.

A saída do nosso programa experimental deu o seguinte resultado:

Criou instância de cronica: cronica01 Serial:3 nroLinha: 5
Criou instância de cronica: cronica02 Serial:4 nroLinha: 5
Criou instância de cronica: cronica03 Serial:5 nroLinha: 5
Descartou instância de cronica: cronica03 Serial:5 nroLinha: 35
Reconheceu instância de cronica: cronica01 Serial:3 nroLinha: 46
Reconheceu instância de cronica: cronica02 Serial:4 nroLinha: 46

Onde se verifica que as instâncias das três crônicas, por serem iguais no primeiro evento, foram criadas juntas logo no primeiro eventro (*SECOQ_CMD_AD52AX*) e continuaram válidas até o segundo evento (*SECOQ_INFOR_AD52AX*). A partir do 3º evento somente as *crônicas 01 e 02* possuem eventos comuns até o 7º evento (*SECOQ_CMD_FD52AX*), após o qual só na *crônica 02* há a falta do evento *SECOQ_EMER_FD52AX* que indica a falha do disjuntor 52AX. Porém, graças à base de conhecimento, compensou-se esta perda de informação "simulada" nesta crônica (simulação da ocorrência do *conhecimento incompleto*) através do aumento do limite de tempo máximo na *crônica 02* após o eventos *SECOQ_CMD_FD52AX*, de forma que mesmo sem o evento faltante (*SECOQ_EMER_FD52AX*) a mesma continuou válida e possibilitou a detecção da falha do disjuntor 52AX.

No caso da *crônica 03*, como este programa experimental ainda não está completo e não verifica se o limite máximo do último evento de uma instância foi extrapolado ou não, a instância desta crônica só foi descartada quando da ocorrência do 3º evento da mesma (*SECOQ_EMER_FD52AX*) num ponto mais distante do *histórico de eventos de*

teste, onde este evento ocorreu já muito depois do limite máximo esperado para o mesmo ter sido extrapolado na *crônica 03*.

6.4. Resultados obtidos com o Cálculo de Eventos/SEC

O *Cálculo de Eventos/SEC* mostrou-se uma ferramenta poderosa para a geração dos modelos (crônicas), no processamento *off-line*, necessários para a análise dos *históricos* do sistema SAGE/SCADA usando-se *Reconhecimento de Crônicas*.

No caso do exemplo deste trabalho, conseguiu-se obter satisfatoriamente uma crônica, ainda que o exemplo seja simplificado. Acredita-se que se obtendo os diagramas lógicos completos (e mais acurados), pode-se obter um número suficiente de crônicas que possam ser utilizados com eficiência num sistema de análise de crônicas em ambiente de produção.

6.5. Resultados obtidos com o Reconhecimento de Crônicas

O *Reconhecimento de Crônicas* mostrou-se também uma ferramenta poderosa para a análise dos *históricos* do SAGE/SCADA, no processamento feito *on-line*, sendo este último caso o objetivo final quando da integração do mesmo à CAE.

Experimentalmente, já se conseguiu criar *"históricos"* aleatórios com mais de 1000 eventos e o *parsing* dos mesmos para comparação com os modelos, gerando várias instancias para cada modelo, demora poucos segundos.

Já para os testes do algorítimo de *Reconhecimento de Crônicas*, visto no capítulo anterior, os testes práticos, próximos da realidade que será encontrada quando da implementação do módulo na CAE, mostraram um desempenho bastante satisfatório, ainda que se tenha que melhorar a implementação do algorítimo para torná-lo mais eficiente, sobretudo no que diz respeito à verificação dos limites máximos dos eventos

das instâncias, de forma que as mesmas possam ser descartadas sem que se tenha que esperar a ocorrência do evento fora do tempo máximo esperado para a ocorrência do mesmo. Os arquivos principais do programa Java desenvolvido para a realização destes testes práticos encontram-se no *ANEXO D* e no *ANEXO E* deste trabalho. As listagens das *"crônicas de teste"*, i.e. arquivos *Cronica01.cro*, *Cronica02.cro* e *Cronica03.cro*, utilizadas encontra-se no *ANEXO F* deste trabalho e o arquivo do *"histórico de eventos de teste"* utilizado encontra-se no *ANEXO G* deste trabalho, i.e. arquivo *logDeEventos30.log*.

Resta, portanto, fazer-se testes em tempo real, em um sistema SAGE/SCADA de produção, para comprovar a real eficácia dos resultados obtidos aqui experimentalmente. Também se deve aprimorar o método algorítmico proposto neste trabalho para o tratamento do *conhecimento incompleto* da informação, pois num sistema real pode-se necessitar de uma metodologia ainda mais robusta para se evitar a ocorrência de respostas dúbias do futuro módulo de *Processamento de Eventos* do CAE.

7. Conclusões

7.1. O que foi desenvolvido neste trabalho

Netse trabalho, desenvolveu-se uma abordagem composta de duas etapas: Uma etapa de criação de crônicas por simulação com **ASP** e **Cálculo de Eventos/SEC** e uma segunda etapa, de processamento que pode ser feito em tempo real, de *Reconhecimento das Crônicas*. As conclusões tiradas por estas duas abordagens foram:

- O *Cálculo de Eventos/SEC* se mostrou uma ferramenta poderosa e promissora para a extração dos modelos necessários dos diagramas lógicos dos relés digitais dos sistemas de proteção.
- *Answer Set Programming* (ASP) se mostrou uma ferramenta muito mais poderosa do que PROLOG para o processamento de programas em *Cálculo de Eventos/SEC*, por não sofrer dos problemas de recursividade observados neste último, mas principalmente por permitir a extração das sequências de eventos sem necessitar de abdução.
- Foi desenvolvido um trabalho de pesquisa das melhores ferramentas e dos melhores algoritmos de modelagem para os diagramas lógicos dos relés digitais. A variante *Simplified Event Calculus* (SEC) do *Cálculo de Eventos* se mostrou muito apropriada para a descrição dos circuitos e sua modelagem.
- O *Reconhecimento de Crônicas* mostrou-se a melhor abordagem para a análise dos modelos gerados por *Cálculo de Eventos/SEC*. A sua simplicidade permite implementações fáceis de aplicativos de *Reconhecimento de Crônicas*.

7.2. O que pode ser desenvolvido a partir deste trabalho

Existem algumas sugestões que podem ser dadas visando desenvolvimentos futuros a partir deste trabalho:

- Como a descrição dos diagramas lógicos dos relés digitais mostrou-se muito trabalhosa, fica a sugestão para o desenvolvimento de uma ferramenta que gere programas em ASP/SEC utilizando-se macros (módulos) de forma que em

diagramas onde haja a ocorrência repetida de *Flip-Flops*, *Pick-UPs* ou *Drop-OUTs*, a implementação possa ser simplificada e automatizada, sem o risco da ocorrência de erros sistemáticos quando da repetição da descrição destes elementos dos circuitos lógicos. Há vários trabalhos publicados discorrendo sobre métodos de uso de macros (módulos) em ASP, com destaque para (BARAL, 2006). No *ANEXO H* deste trabalho fica a sugestão de um exemplo de modelo para ser usado numa implementação futura de um gerador de *scripts* em ASP que suporte a utilização de modularização.

- A verificação do funcionamento do *Reconhecimento de Crônicas* a partir de modelos incompletos é uma abordagem interessante para contornar problemas práticos de falhas nos *históricos* do SAGE/SCADA (problema do *conhecimento incompleto*), que ocorre, por exemplo, quando um relé digital, por defeito, deixa de informar ao SAGE/SCADA LOCAL a ocorrência de uma falha em algum dispositivo da proteção elétrica da subestação. Fica, portanto, a sugestão para que futuros trabalhos explorem mais o problema do *conhecimento incompleto* na análise de *Processamento de Eventos*.
- Uma melhor integração entre o conhecimento dos operadores e os resultados obtidos pelos sistemas de monitoramento de eventos poderá ajudar em muito numa maior minimização do problema do *conhecimento incompleto*.

Referências Bibliográficas

ALMEIDA, E. M. de. 2011. Norma IEC 61850 - Novo Padrão em Automação de Subestações. Fortaleza : Universidade Federal do Ceará, 2011.

AQUINO, R. M. de, VALE, M. H. M. 2009. Impacto da Parcela Variável na Expansão, Operação e Manutenção do Sistema Interligado Nacional – Propostas para Atualização de Procedimentos. 2009.

AREVA-PROEL. 2009. *Elecnor - Coqueiros Transmissora de Energia Elétrica Ltda - SE Barra dos Coqueiros 230kV - Diagrama de Arquitetura.* 2009.

—. 2009. *Elecnor - Coqueiros Transmissora de Energia Elétrica Ltda - SE Barra dos Coqueiros 230kV - Diagrama Lógico.* 2009.

BARAL, C., DZIFCAK, J., TAKAHASHI, H. 2006. Macros, Macro Calls and Use of Ensembles in Modular Answer Set Programming. *Springer-Verlag Berlin Heidelberg.* 2006.

CASANOVA, M. A., BREITMAN, K. K. 2008. *Cláusulas de Horn e Resolução SLD.* 2008.

CASANOVA, M. A., GIORNO, F. A. C., FURTADO, A. L. 2006. *Programação em Lógica e a Linguagem Prolog.* 2006.

CENTRO DE PESQUISAS EM ENERGIA ELÉTRICA - CEPEL. 2009. SAGE - Sistema Aberto de Supervisão e Controle. *Boletim 93 - Notas de Atualização.* [Online] 2009. [Citado em: 05 de 12 de 2013.] http://www.sage.cepel.br/boletins/boletim093.html.

Circumscription – A Form of Non-Monotonic Reasoning. **McCARTHY, J. 1980.** 1980.

COLLAVIZZA, A. W. 2013. *Uma Central de Alarmes e Eventos (Título Provisório): Dissertação de Mestrado.* Rio de Janeiro, RJ, Brasil : COPPE/UFRJ, 2013.

CORDIER, M.-O., KRIVINE, J.-P. K., LABORIE P., THIEBAUX S. 1998. Alarm Processing and Reconfiguration in Power Distribution Systems. *Tasks and Methods in Applied Artificial Intelligence.* 1998.

DECHTER R., MEIRI I., PEARL J. 1991. Temporal Constraint Networks. *Artificial Intelligence.* 1991, Vol. 49.

DOUSSON C., GABORIT P., GHALLAB M. 1993. Situation Recognition: Representation and Algorithms. *IJCAI.* 1993.

DURKIN, J. 1994. *Expert Systems: Design and Development.* s.l. : Macmillan Publishing, 1994.

ELETROBRAS-CEPEL. 2011. *Manual de Configuração – SAGE-SCADA.* 2011.

ESHGHI, K. 1988. Abductive planning with event calculus. *Logic Programming: Proceedings of the Fifth International.* 1988.

GELFOND, M., LIFSCHITZ, V., RABINOV, A. 1991. What are the Limitations of the Situation Calculus? In Essays in Honor of Woody Bledsor. 1991.

GHALLAB, M. 1996. OnChronicles: Representation, On-line Recognition and Learning. *Proceedings of KR.* 1996.

HAYKIN, S. 1999. *Redes Neurais - Princípios e Prática.* 1999.

IEEE STANDARD ELECTRIC POWER SYSTEM. 1998. *Device Function Numbers acc. to IEEE C.37.2-1991.* s.l. : IEEE, 1998.

KOWALSKI, R. A. 1979. *Logic for Problem Solving.* 1979.

KOWALSKI, R. A., SERGOT, M. J. 1986. A Logic-Based Calculus of Events. *New Generation Computing.* 1986, pp. 67-95.

KOWALSKI, R.A. 1986. Database updates in the event calculus. *Technical Report DOC.* 1986.

MACKWORTH, A. K., FREUDER, E. C. 1985. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence.* 1985, Vol. 25.

MARCOS, S. T. e DIAS, I. C. 2005. *As Espécies de Raciocínio: dedução, indução e abdução.* 2005.

MAREK, V. W. e TRUSZCZYNSKI, M. 1999. Stable models and an alternative logic programming paradigm. In The Logic Programming Paradigm. 1999.

McCARTHY, J. e HEYES, P. J. 1969. Some Philosophical Problems from the Standpoint of Artificial lntelligence. *Machine Intelligence 4.* 1969.

McCARTHY, J. 1977. Epistemological Problems of Artificial Intelligence. 1977.

—. 1963. Situations, Actions and Causal Laws. *Stanford Artificial lntelligence Project.* 1963.

MILLER, R., SHANAHAN, M. 2002. Some alternative formulations of the event calculus. 2002.

—. 1999. The event calculus in classical logic—Alternative axiomatisations. *Linköping Electronic Articles in Computer and Information.* 1999.

MOREALE, M. dos S. 2007. *Técnicas para Treinamento de Operadores de Sistema Elétrico Utilizando Simulador Com Base na Interface de Tempo Real (Tese de Mestrado).* Florianópolis, SC, Brasil : Universidade Federal de Santa Catarina, 2007.

NEGNEVITSKY, M. 2002. *Artificial Intelligence. A Guide to Intelligent Systems.* s.l. : Pearson Education, 2002.

NIEMELÄ, I. 1999. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence.* 1999.

OPERADOR NACIONAL DO SISTEMA - ONS. 2010. *Apuração mensal das parcelas variáveis referentes à disponibilidade de instalações da Rede Básica.* s.l. : Operador Nacional do Sistema - ONS, 2010.

—. 2007. *Resolução Normativa N° 270 de 26 de Junho de 2007.* 2007.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Pausible Inference.* 1988.

PEREIRA, S. do L. 2009. *Introdução à Linguagem PROLOG.* 2009.

PIGARI, A. A., TOLENTINO FILHO, E. N. e SANTOS, N. M. 2012. *Raciocínio em IA.* s.l. : http://www.din.uem.br/~ia/intelige/raciocinio2, último acesso em agosto de 2012, 2012.

POSTDAM, UNIVERSITY OF. 2012. Potassco - the Potsdam Answer Set Solving Collection. *Potassco.* [Online] University of Postdam, 2012. [Citado em: 22 de 04 de 2013.] http://potassco.sourceforge.net.

REZENDE, S. O. 2003. *Sistemas Inteligentes - Fundamentos e Aplicações.* s.l. : Manole, 2003.

SADRI, F. 1987. Three recent approaches to temporal reasoning. *Temporal Logics and their Applications.* 1987.

SCHEWEITZER ENGINEERING LABORATORIES, INC. 2013. Relé de Proteção SEL-421. *SEL - Scheweitzer Engineering Laboratories.* [Online] 2013. [Citado em: 12 de 08 de 2013.] http://www.selinc.com.br/produtos/SEL-421.aspx.

—. 2011. *SEL - SEL-421 Protection and Automation System.* [PDF] Pullman, Washington, USA : Scheweitzer Engineering Laboratories, Inc., 2011.

SHANAHAN, M. 2000. *An Abductive Event Calculus Planner.* 2000.

—. 1997a. Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. *Journal of Logic Programming.* 1997a, pp. 207-239.

—. 1997b. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* Cambridge : MIT Press, 1997b.

SILVESTRE, R. S. 2008. Lógica e Sistemas Lógicos. 2008.

THOMAS, E., GIOVAMBATTISTA, I., THOMAS, K. 2009. Answer Set Programming: A Primer? *5th International ReasoningWeb Summer School 2009.* 2009, Vol. 5689.

TRIANGLE MICROWORKS, INC. 2002. *DNP3 Overview.* [PDF] Raleigh, North Carolina, USA : Triangle MicroWorks, Inc, 2002.

WATERMAN, D. A. 1986. *A Guide to Expert Systems.* s.l. : Addison-Wesley, 1986.

ZUBEN, F. J. V. 2011. *Sistema Baseado em Regras e Árvores de Decisão.* 2011.

Anexo A

Código fonte ASP/SEC da modelagem do circuito de Exemplo do Sistema de Proteção de 230kV

```
%------------ Exemplo-Sistema-Protecao-BC-230Kv.lp ----------------
%
% Implementação em ASP / Cálculo de Eventos da versão simplificada de
% um esquema de proteção elétrico de 230Kv, baseado nos esquema de
% proteção da Subestação Barra dos Coqueiros 230Kv.
% Esta implementação tem por objetivo facilitar a aquisição do Modelo
% (Crônica) a ser utilizada no Reconhecimento de Crônicas para achar
% as causas raízes de um evento a ser simulado nesta implementação.
%
% Por : Rafael Jorge Csura Szendrodi <szendro@lemt.ufrj.br>
% Em  : 30/08/2013
%
% Últimas modificações:
%
% Por : Rafael Jorge Csura Szendrodi <szendro@lemt.ufrj.br>
% Em  : 12/11/2013
%

%================== Simplified EC Axioms ==================

holdsAt(F,T) :- initially(F), not clipped(0,F,T), timepoint(T).

holdsAt(F,T) :- happens(E,T0), initiates(E,F,T0), T0<T, not clipped(T0,F,T),
timepoint(T), timepoint(T0).

clipped(T1,F,T2) :- happens(E,T), terminates(E,F,T), T1<=T, T<T2, timepoint(T1),
timepoint(T2), timepoint(T).

%==========================================================

timepoint(0..150).

nivelLogico(0).
nivelLogico(1).

inv(0,1).
inv(1,0).

and(0,0,0).
```

```
and(0,1,0).
and(1,0,0).
and(1,1,1).

and3(A,B,C,D) :- and(A,B,X), and(X,C,D).
and4(A,B,C,D,E) :- and3(A,B,C,X), and(X,D,E).
and5(A,B,C,D,E,F) :- and3(A,B,C,X), and3(X,D,E,F).
and6(A,B,C,D,E,F,G) :- and3(A,B,C,X), and3(D,E,F,Y), and(X,Y,G).
and7(A,B,C,D,E,F,G,H) :- and4(A,B,C,D,X), and3(E,F,G,Y), and(X,Y,H).

or(0,0,0).
or(0,1,1).
or(1,0,1).
or(1,1,1).

or3(A,B,C,D) :- or(A,B,X), or(X,C,D).
or4(A,B,C,D,E) :- or3(A,B,C,X), or(X,D,E).

%============================================================




% === Figura 1 - Coqueiros Simplificado F_15 ===


% === Entradas Circuito F_15 (Secionadora 57AX) ===

% === SECIONADORA 57AX - ABERTA
initially(dI1P(1)).

 terminates(dI1Pup,dI1P(0),T) :- timepoint(T).
   initiates(dI1Pup,dI1P(1),T) :- timepoint(T).
 terminates(dI1Pdw,dI1P(1),T) :- timepoint(T).
   initiates(dI1Pdw,dI1P(0),T) :- timepoint(T).

happens(dI1Pup,1).


% === SECIONADORA 57AX - FECHADA
initially(dI2P(0)).

 terminates(dI2Pup,dI2P(0),T) :- timepoint(T).
   initiates(dI2Pup,dI2P(1),T) :- timepoint(T).
 terminates(dI2Pdw,dI2P(1),T) :- timepoint(T).
   initiates(dI2Pdw,dI2P(0),T) :- timepoint(T).

happens(dI2Pdw,1).
```

```
% === Entradas Circuito F_15 (Secionadora 89AX2) ===

% === SECIONADORA 89AX2 - ABERTA
initially(dI3P(0)).

 terminates(dI3Pup,dI3P(0),T) :- timepoint(T).
   initiates(dI3Pup,dI3P(1),T) :- timepoint(T).
 terminates(dI3Pdw,dI3P(1),T) :- timepoint(T).
   initiates(dI3Pdw,dI3P(0),T) :- timepoint(T).

happens(dI3Pdw,1).
happens(dI3Pup,21).


% === SECIONADORA 89AX2 - FECHADA
initially(dI4P(1)).

 terminates(dI4Pup,dI4P(0),T) :- timepoint(T).
   initiates(dI4Pup,dI4P(1),T) :- timepoint(T).
 terminates(dI4Pdw,dI4P(1),T) :- timepoint(T).
   initiates(dI4Pdw,dI4P(0),T) :- timepoint(T).

happens(dI4Pup,1).
happens(dI4Pdw,20).


%============ Pick-up timer Circuito F_15_1 ============
initially(monoOutput_PU_CF15_1(0)).
happens(upTriger_PU_CF15_1,T3) :- holdsAt(inputIs_PU_CF15_1(0),T1),
holdsAt(inputIs_PU_CF15_1(1),T2), T2=T1+1, T3=T1+30, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF15_1,monoOutput_PU_CF15_1(0),T) :-
happens(upTriger_PU_CF15_1,T), holdsAt(inputIs_PU_CF15_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF15_1,monoOutput_PU_CF15_1(1),T) :-
happens(upTriger_PU_CF15_1,T), holdsAt(inputIs_PU_CF15_1(1),T), timepoint(T).

happens(downTriger_PU_CF15_1,0) :- initially(inputIs_PU_CF15_1(0)).
happens(downTriger_PU_CF15_1,T1) :- holdsAt(inputIs_PU_CF15_1(1),T1),
holdsAt(inputIs_PU_CF15_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF15_1,monoOutput_PU_CF15_1(1),T) :-
happens(downTriger_PU_CF15_1,T), holdsAt(monoOutput_PU_CF15_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF15_1,monoOutput_PU_CF15_1(0),T) :-
happens(downTriger_PU_CF15_1,T), holdsAt(monoOutput_PU_CF15_1(1),T),
timepoint(T).


%============ Pick-up timer Circuito F_15_2 ============
```

```
initially(monoOutput_PU_CF15_2(0)).
happens(upTriger_PU_CF15_2,T3) :- holdsAt(inputIs_PU_CF15_2(0),T1),
holdsAt(inputIs_PU_CF15_2(1),T2), T2=T1+1, T3=T1+30, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF15_2,monoOutput_PU_CF15_2(0),T) :-
happens(upTriger_PU_CF15_2,T), holdsAt(inputIs_PU_CF15_2(1),T), timepoint(T).
 initiates(upTriger_PU_CF15_2,monoOutput_PU_CF15_2(1),T) :-
happens(upTriger_PU_CF15_2,T), holdsAt(inputIs_PU_CF15_2(1),T), timepoint(T).

happens(downTriger_PU_CF15_2,0) :- initially(inputIs_PU_CF15_2(0)).
happens(downTriger_PU_CF15_2,T1) :- holdsAt(inputIs_PU_CF15_2(1),T1),
holdsAt(inputIs_PU_CF15_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF15_2,monoOutput_PU_CF15_2(1),T) :-
happens(downTriger_PU_CF15_2,T), holdsAt(monoOutput_PU_CF15_2(1),T),
timepoint(T).
 initiates(downTriger_PU_CF15_2,monoOutput_PU_CF15_2(0),T) :-
happens(downTriger_PU_CF15_2,T), holdsAt(monoOutput_PU_CF15_2(1),T),
timepoint(T).


% === SAÍDAS Circuito F_15 (Secionadora 57AX) ===

% vI01_15 - Secionadora 57AX Aberta
holdsAt(vI01_15(C),T) :- and(A,X,C), inv(B,X), holdsAt(dI1P(A),T),
holdsAt(dI2P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI02_15 - Secionadora 57AX Fechada
holdsAt(vI02_15(C),T) :- and(X,B,C), inv(A,X), holdsAt(dI1P(A),T),
holdsAt(dI2P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI03_15 - Secionadora 57AX Indefinida
holdsAt(inputIs_PU_CF15_1(E),T) :- or(C,D,E), and(A,B,C), and(X,Y,D), inv(A,X),
inv(B,Y), holdsAt(dI1P(A),T), holdsAt(dI2P(B),T), nivelLogico(X), nivelLogico(Y),
nivelLogico(C), nivelLogico(D), nivelLogico(E), timepoint(T).
holdsAt(vI03_15(F),T) :- holdsAt(monoOutput_PU_CF15_1(F),T), timepoint(T).


% === SAÍDAS Circuito F_15 (Secionadora 89AX2) ===

% vI05_15 - Secionadora 89AX2 Aberta
holdsAt(vI05_15(C),T) :- and(A,X,C), inv(B,X), holdsAt(dI3P(A),T),
holdsAt(dI4P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI06_15 - Secionadora 89AX2 Fechada
holdsAt(vI06_15(C),T) :- and(X,B,C), inv(A,X), holdsAt(dI3P(A),T),
holdsAt(dI4P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI07_15 - Secionadora 89AX2 Indefinida
```

```
holdsAt(inputIs_PU_CF15_2(E),T) :- or(C,D,E), and(A,B,C), and(X,Y,D), inv(A,X),
inv(B,Y), holdsAt(dI3P(A),T), holdsAt(dI4P(B),T), nivelLogico(X), nivelLogico(Y),
nivelLogico(C), nivelLogico(D), nivelLogico(E), timepoint(T).
holdsAt(vI07_15(F),T) :- holdsAt(monoOutput_PU_CF15_2(F),T), timepoint(T).

%= FIM - Figura 1 - Coqueiros Simplificado F_15 ===



% === Figura 2 - Coqueiros Simplificado F_16 ===

% === Entradas Circuito F_16 (Secionadora 89AX4) ===

% === SECIONADORA 89AX4 - ABERTA
initially(dI5P(1)).

 terminates(dI5Pup,dI5P(0),T) :- timepoint(T).
   initiates(dI5Pup,dI5P(1),T) :- timepoint(T).
 terminates(dI5Pdw,dI5P(1),T) :- timepoint(T).
   initiates(dI5Pdw,dI5P(0),T) :- timepoint(T).

happens(dI5Pdw,1).


% === SECIONADORA 89AX4 - FECHADA
initially(dI6P(0)).

 terminates(dI6Pup,dI6P(0),T) :- timepoint(T).
   initiates(dI6Pup,dI6P(1),T) :- timepoint(T).
 terminates(dI6Pdw,dI6P(1),T) :- timepoint(T).
   initiates(dI6Pdw,dI6P(0),T) :- timepoint(T).

happens(dI6Pdw,1).


%============ Pick-up timer Circuito F_16_1 ============
initially(monoOutput_PU_CF16_1(0)).
happens(upTriger_PU_CF16_1,T3) :- holdsAt(inputIs_PU_CF16_1(0),T1),
holdsAt(inputIs_PU_CF16_1(1),T2), T2=T1+1, T3=T1+1, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF16_1,monoOutput_PU_CF16_1(0),T) :-
happens(upTriger_PU_CF16_1,T), holdsAt(inputIs_PU_CF16_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF16_1,monoOutput_PU_CF16_1(1),T) :-
happens(upTriger_PU_CF16_1,T), holdsAt(inputIs_PU_CF16_1(1),T), timepoint(T).

happens(downTriger_PU_CF16_1,0) :- initially(inputIs_PU_CF16_1(0)).
happens(downTriger_PU_CF16_1,T1) :- holdsAt(inputIs_PU_CF16_1(1),T1),
holdsAt(inputIs_PU_CF16_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
```

```
terminates(downTriger_PU_CF16_1,monoOutput_PU_CF16_1(1),T) :-
happens(downTriger_PU_CF16_1,T), holdsAt(monoOutput_PU_CF16_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF16_1,monoOutput_PU_CF16_1(0),T) :-
happens(downTriger_PU_CF16_1,T), holdsAt(monoOutput_PU_CF16_1(1),T),
timepoint(T).


%============ Pick-up timer Circuito F_16_2 ============
initially(monoOutput_PU_CF16_2(0)).
happens(upTriger_PU_CF16_2,T3) :- holdsAt(inputIs_PU_CF16_2(0),T1),
holdsAt(inputIs_PU_CF16_2(1),T2), T2=T1+1, T3=T1+60, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF16_2,monoOutput_PU_CF16_2(0),T) :-
happens(upTriger_PU_CF16_2,T), holdsAt(inputIs_PU_CF16_2(1),T), timepoint(T).
 initiates(upTriger_PU_CF16_2,monoOutput_PU_CF16_2(1),T) :-
happens(upTriger_PU_CF16_2,T), holdsAt(inputIs_PU_CF16_2(1),T), timepoint(T).

happens(downTriger_PU_CF16_2,0) :- initially(inputIs_PU_CF16_2(0)).
happens(downTriger_PU_CF16_2,T1) :- holdsAt(inputIs_PU_CF16_2(1),T1),
holdsAt(inputIs_PU_CF16_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF16_2,monoOutput_PU_CF16_2(1),T) :-
happens(downTriger_PU_CF16_2,T), holdsAt(monoOutput_PU_CF16_2(1),T),
timepoint(T).
 initiates(downTriger_PU_CF16_2,monoOutput_PU_CF16_2(0),T) :-
happens(downTriger_PU_CF16_2,T), holdsAt(monoOutput_PU_CF16_2(1),T),
timepoint(T).


% === SAÍDAS Circuito F_16 (Secionadora 89AX4) ===

% vI09_16 - Secionadora 89AX4 Aberta
holdsAt(vI09_16(C),T) :- and(A,X,C), inv(B,X), holdsAt(dI5P(A),T),
holdsAt(dI6P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI10_16 - Secionadora 89AX4 Fechada
holdsAt(vI10_16(C),T) :- and(Y,B,C), inv(A,Y), holdsAt(dI5P(A),T),
holdsAt(dI6P(B),T), nivelLogico(Y), nivelLogico(C), timepoint(T).

% vI11_16 - Secionadora 89AX4 Indeterminada
holdsAt(inputIs_PU_CF16_1(C),T) :- and(A,B,C), holdsAt(dI5P(A),T),
holdsAt(dI6P(B),T), nivelLogico(C), timepoint(T).
holdsAt(vI11_16(D),T) :- holdsAt(monoOutput_PU_CF16_1(D),T), timepoint(T).

% vI12_16 - Secionadora 89AX4 Intermediária
holdsAt(inputIs_PU_CF16_2(C),T) :- and(X,Y,C), inv(A,X), inv(B,Y),
holdsAt(dI5P(A),T), holdsAt(dI6P(B),T), nivelLogico(X), nivelLogico(Y),
nivelLogico(C), timepoint(T).
holdsAt(vI12_16(D),T) :- holdsAt(monoOutput_PU_CF16_2(D),T), timepoint(T).
```

```
%= FIM - Figura 2 - Coqueiros Simplificado F_16 ===


% === Figura 3 - Coqueiros Simplificado F_17 ===

% === Entradas Circuito F_17 (Secionadora 89AX3) ===

% === SECIONADORA 89AX3 - ABERTA
initially(dI9P(1)).

 terminates(dI9Pup,dI9P(0),T) :- timepoint(T).
   initiates(dI9Pup,dI9P(1),T) :- timepoint(T).
 terminates(dI9Pdw,dI9P(1),T) :- timepoint(T).
   initiates(dI9Pdw,dI9P(0),T) :- timepoint(T).

happens(dI9Pup,1).
happens(dI9Pdw,14).
happens(dI9Pup,21).


% === SECIONADORA 89AX3 - FECHADA
initially(dI10P(0)).

 terminates(dI10Pup,dI10P(0),T) :- timepoint(T).
   initiates(dI10Pup,dI10P(1),T) :- timepoint(T).
 terminates(dI10Pdw,dI10P(1),T) :- timepoint(T).
   initiates(dI10Pdw,dI10P(0),T) :- timepoint(T).

happens(dI10Pdw,1).
happens(dI10Pup,15).
happens(dI10Pdw,20).


% === Entradas Circuito F_17 (Secionadora 89AX1) ===

% === SECIONADORA 89AX1 - ABERTA
initially(dI7P(0)).

 terminates(dI7Pup,dI7P(0),T) :- timepoint(T).
   initiates(dI7Pup,dI7P(1),T) :- timepoint(T).
 terminates(dI7Pdw,dI7P(1),T) :- timepoint(T).
   initiates(dI7Pdw,dI7P(0),T) :- timepoint(T).

happens(dI7Pdw,1).
happens(dI7Pup,12).
```

```
% === SECIONADORA 89AX1 - FECHADA
initially(dI8P(1)).

 terminates(dI8Pup,dIP8(0),T) :- timepoint(T).
   initiates(dI8Pup,dIP8(1),T) :- timepoint(T).
 terminates(dI8Pdw,dIP8(1),T) :- timepoint(T).
   initiates(dI8Pdw,dIP8(0),T) :- timepoint(T).

happens(dI8Pup,1).
happens(dI8Pdw,11).


%============ Pick-up timer Circuito F_17_1 ============
initially(monoOutput_PU_CF17_1(0)).
happens(upTriger_PU_CF17_1,T3) :- holdsAt(inputIs_PU_CF17_1(0),T1),
holdsAt(inputIs_PU_CF17_1(1),T2), T2=T1+1, T3=T1+1, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF17_1,monoOutput_PU_CF17_1(0),T) :-
happens(upTriger_PU_CF17_1,T), holdsAt(inputIs_PU_CF17_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF17_1,monoOutput_PU_CF17_1(1),T) :-
happens(upTriger_PU_CF17_1,T), holdsAt(inputIs_PU_CF17_1(1),T), timepoint(T).

happens(downTriger_PU_CF17_1,0) :- initially(inputIs_PU_CF17_1(0)).
happens(downTriger_PU_CF17_1,T1) :- holdsAt(inputIs_PU_CF17_1(1),T1),
holdsAt(inputIs_PU_CF17_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF17_1,monoOutput_PU_CF17_1(1),T) :-
happens(downTriger_PU_CF17_1,T), holdsAt(monoOutput_PU_CF17_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF17_1,monoOutput_PU_CF17_1(0),T) :-
happens(downTriger_PU_CF17_1,T), holdsAt(monoOutput_PU_CF17_1(1),T),
timepoint(T).


%============ Pick-up timer Circuito F_17_2 ============
initially(monoOutput_PU_CF17_2(0)).
happens(upTriger_PU_CF17_2,T3) :- holdsAt(inputIs_PU_CF17_2(0),T1),
holdsAt(inputIs_PU_CF17_2(1),T2), T2=T1+1, T3=T1+60, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF17_2,monoOutput_PU_CF17_2(0),T) :-
happens(upTriger_PU_CF17_2,T), holdsAt(inputIs_PU_CF17_2(1),T), timepoint(T).
 initiates(upTriger_PU_CF17_2,monoOutput_PU_CF17_2(1),T) :-
happens(upTriger_PU_CF17_2,T), holdsAt(inputIs_PU_CF17_2(1),T), timepoint(T).

happens(downTriger_PU_CF17_2,0) :- initially(inputIs_PU_CF17_2(0)).
happens(downTriger_PU_CF17_2,T1) :- holdsAt(inputIs_PU_CF17_2(1),T1),
holdsAt(inputIs_PU_CF17_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF17_2,monoOutput_PU_CF17_2(1),T) :-
happens(downTriger_PU_CF17_2,T), holdsAt(monoOutput_PU_CF17_2(1),T),
timepoint(T).
```

```
 initiates(downTriger_PU_CF17_2,monoOutput_PU_CF17_2(0),T) :-
happens(downTriger_PU_CF17_2,T), holdsAt(monoOutput_PU_CF17_2(1),T),
timepoint(T).


%============ Pick-up timer Circuito F_17_3 ============
initially(monoOutput_PU_CF17_3(0)).
happens(upTriger_PU_CF17_3,T3) :- holdsAt(inputIs_PU_CF17_3(0),T1),
holdsAt(inputIs_PU_CF17_3(1),T2), T2=T1+1, T3=T1+1, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF17_3,monoOutput_PU_CF17_3(0),T) :-
happens(upTriger_PU_CF17_3,T), holdsAt(inputIs_PU_CF17_3(1),T), timepoint(T).
 initiates(upTriger_PU_CF17_3,monoOutput_PU_CF17_3(1),T) :-
happens(upTriger_PU_CF17_3,T), holdsAt(inputIs_PU_CF17_3(1),T), timepoint(T).

happens(downTriger_PU_CF17_3,0) :- initially(inputIs_PU_CF17_3(0)).
happens(downTriger_PU_CF17_3,T1) :- holdsAt(inputIs_PU_CF17_3(1),T1),
holdsAt(inputIs_PU_CF17_3(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF17_3,monoOutput_PU_CF17_3(1),T) :-
happens(downTriger_PU_CF17_3,T), holdsAt(monoOutput_PU_CF17_3(1),T),
timepoint(T).
 initiates(downTriger_PU_CF17_3,monoOutput_PU_CF17_3(0),T) :-
happens(downTriger_PU_CF17_3,T), holdsAt(monoOutput_PU_CF17_3(1),T),
timepoint(T).


%============ Pick-up timer Circuito F_17_4 ============
initially(monoOutput_PU_CF17_4(0)).
happens(upTriger_PU_CF17_4,T3) :- holdsAt(inputIs_PU_CF17_4(0),T1),
holdsAt(inputIs_PU_CF17_4(1),T2), T2=T1+1, T3=T1+60, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF17_4,monoOutput_PU_CF17_4(0),T) :-
happens(upTriger_PU_CF17_4,T), holdsAt(inputIs_PU_CF17_4(1),T), timepoint(T).
 initiates(upTriger_PU_CF17_4,monoOutput_PU_CF17_4(1),T) :-
happens(upTriger_PU_CF17_4,T), holdsAt(inputIs_PU_CF17_4(1),T), timepoint(T).

happens(downTriger_PU_CF17_4,0) :- initially(inputIs_PU_CF17_4(0)).
happens(downTriger_PU_CF17_4,T1) :- holdsAt(inputIs_PU_CF17_4(1),T1),
holdsAt(inputIs_PU_CF17_4(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF17_4,monoOutput_PU_CF17_4(1),T) :-
happens(downTriger_PU_CF17_4,T), holdsAt(monoOutput_PU_CF17_4(1),T),
timepoint(T).
 initiates(downTriger_PU_CF17_4,monoOutput_PU_CF17_4(0),T) :-
happens(downTriger_PU_CF17_4,T), holdsAt(monoOutput_PU_CF17_4(1),T),
timepoint(T).


% === SAÍDAS Circuito F_17 (Secionadora 89AX3) ===
```

```
% vI17_17 - Secionadora 89AX3 Aberta
holdsAt(vI17_17(C),T) :- and(A,X,C), inv(B,X), holdsAt(dI9P(A),T),
holdsAt(dI10P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI18_17 - Secionadora 89AX3 Fechada
holdsAt(vI18_17(C),T) :- and(X,B,C), inv(A,X), holdsAt(dI9P(A),T),
holdsAt(dI10P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI19_17 - Secionadora 89AX3 Indeterminada
holdsAt(inputIs_PU_CF17_1(C),T) :- and(A,B,C), holdsAt(dI9P(A),T),
holdsAt(dI10P(B),T), nivelLogico(C), timepoint(T).
holdsAt(vI19_17(D),T) :- holdsAt(monoOutput_PU_CF17_1(D),T), timepoint(T).

% vI20_17 - Secionadora 89AX3 Intermediária
holdsAt(inputIs_PU_CF17_2(C),T) :- and(X,Y,C), inv(A,X), inv(B,Y),
holdsAt(dI9P(A),T), holdsAt(dI10P(B),T), nivelLogico(X), nivelLogico(Y),
nivelLogico(C), timepoint(T).
holdsAt(vI20_17(D),T) :- holdsAt(monoOutput_PU_CF17_2(D),T), timepoint(T).


% === SAÍDAS Circuito F_17 (Secionadora 89AX1) ===

% vI13_17 - Secionadora 89AX1 Aberta
holdsAt(vI13_17(C),T) :- and(A,X,C), inv(B,X), holdsAt(dI7P(A),T),
holdsAt(dI8P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI14_17 - Secionadora 89AX1 Fechada
holdsAt(vI14_17(C),T) :- and(X,B,C), inv(A,X), holdsAt(dI7P(A),T),
holdsAt(dI8P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI15_17 - Secionadora 89AX1 Indeterminada
holdsAt(inputIs_PU_CF17_3(C),T) :- and(A,B,C), holdsAt(dI7P(A),T),
holdsAt(dI8P(B),T), nivelLogico(C), timepoint(T).
holdsAt(vI15_17(D),T) :- holdsAt(monoOutput_PU_CF17_3(D),T), timepoint(T).

% vI16_17 - Secionadora 89AX1 Intermediária
holdsAt(inputIs_PU_CF17_4(C),T) :- and(X,Y,C), inv(A,X), inv(B,Y),
holdsAt(dI7P(A),T), holdsAt(dI8P(B),T), nivelLogico(X), nivelLogico(Y),
nivelLogico(C), timepoint(T).
holdsAt(vI16_17(D),T) :- holdsAt(monoOutput_PU_CF17_4(D),T), timepoint(T).

%= FIM - Figura 3 - Coqueiros Simplificado F_17 ===


% === Figura 4 - Coqueiros Simplificado F_18 ===

% === Entradas Circuito F_18 (Disjuntor 52AX) ===
```

```
% === DISJUNTOR 52AX - ABERTO
initially(dI11P(0)).

 terminates(dI11Pup,dI11P(0),T) :- timepoint(T).
   initiates(dI11Pup,dI11P(1),T) :- timepoint(T).
 terminates(dI11Pdw,dI11P(1),T) :- timepoint(T).
   initiates(dI11Pdw,dI11P(0),T) :- timepoint(T).

happens(dI11Pdw,1).
happens(dI11Pup,8).


% === DISJUNTOR 52AX - FECHAD0
initially(dI12P(1)).

 terminates(dI12Pup,dI12P(0),T) :- timepoint(T).
   initiates(dI12Pup,dI12P(1),T) :- timepoint(T).
 terminates(dI12Pdw,dI12P(1),T) :- timepoint(T).
   initiates(dI12Pdw,dI12P(0),T) :- timepoint(T).


happens(dI12Pup,1).
happens(dI12Pdw,7).


%============ Pick-up timer Circuito F_18 ============
initially(monoOutput_PU_CF18(0)).
happens(upTriger_PU_CF18,T3) :- holdsAt(inputIs_PU_CF18(0),T1),
holdsAt(inputIs_PU_CF18(1),T2), T2=T1+1, T3=T1+1, timepoint(T1), timepoint(T2),
timepoint(T3).
terminates(upTriger_PU_CF18,monoOutput_PU_CF18(0),T) :-
happens(upTriger_PU_CF18,T), holdsAt(inputIs_PU_CF18(1),T), timepoint(T).
 initiates(upTriger_PU_CF18,monoOutput_PU_CF18(1),T) :-
happens(upTriger_PU_CF18,T), holdsAt(inputIs_PU_CF18(1),T), timepoint(T).

happens(downTriger_PU_CF18,0) :- initially(inputIs_PU_CF18(0)).
happens(downTriger_PU_CF18,T1) :- holdsAt(inputIs_PU_CF18(1),T1),
holdsAt(inputIs_PU_CF18(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF18,monoOutput_PU_CF18(1),T) :-
happens(downTriger_PU_CF18,T), holdsAt(monoOutput_PU_CF18(1),T),
timepoint(T).
 initiates(downTriger_PU_CF18,monoOutput_PU_CF18(0),T) :-
happens(downTriger_PU_CF18,T), holdsAt(monoOutput_PU_CF18(1),T),
timepoint(T).


% === SAÍDAS Circuito F_18 (Secionadora 52AX) ===

% vI21_18 - Disjuntor 52AX Aberto
```

```
holdsAt(vI21_18(C),T) :- and(A,X,C), inv(B,X), holdsAt(dIP11(A),T),
holdsAt(dIP12(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI22_18 - Disjuntor 52AX Aberto
holdsAt(vI22_18(C),T) :- and(X,B,C), inv(A,X), holdsAt(dI11P(A),T),
holdsAt(dI12P(B),T), nivelLogico(X), nivelLogico(C), timepoint(T).

% vI23_18 - Disjuntor 52AX Indefinida
holdsAt(inputIs_PU_CF18(E),T) :- or(C,D,E), and(A,B,C), and(X,Y,D), inv(A,X),
inv(B,Y), holdsAt(dI11P(A),T), holdsAt(dI12P(B),T), nivelLogico(X), nivelLogico(Y),
nivelLogico(C), nivelLogico(D), nivelLogico(E), timepoint(T).
holdsAt(vI23_18(F),T) :- holdsAt(monoOutput_PU_CF18(F),T), timepoint(T).

%= FIM - Figura 4 - Coqueiros Simplificado F_18 ===



% === Figura 5 - Coqueiros Simplificado F_80 ===

% === Entradas Circuito F_80 ===

% COMANDO FECHAR:
% === (IHM-UCC) Seleção UCC L/R - Remoto
initially(eP01_90(1)).

 terminates(eP01_90up,eP01_90(0),T) :- timepoint(T).
   initiates(eP01_90up,eP01_90(1),T) :- timepoint(T).
 terminates(eP01_90dw,eP01_90(1),T) :- timepoint(T).
   initiates(eP01_90dw,eP01_90(0),T) :- timepoint(T).

happens(eP01_90up,1).


% === (IHM-UCC) Fechar Secionadora 89AX1
initially(eP02_90(0)).

 terminates(eP02_90up,eP02_90(0),T) :- timepoint(T).
   initiates(eP02_90up,eP02_90(1),T) :- timepoint(T).
 terminates(eP02_90dw,eP02_90(1),T) :- timepoint(T).
   initiates(eP02_90dw,eP02_90(0),T) :- timepoint(T).

happens(eP02_90dw,1).


% === (IHM-COL) Fechar Secionadora 89AX1
initially(iECA(0)).

 terminates(iECAup,iECA(0),T) :- timepoint(T).
   initiates(iECAup,iECA(1),T) :- timepoint(T).
```

```
 terminates(iECAdw,iECA(1),T) :- timepoint(T).
   initiates(iECAdw,iECA(0),T) :- timepoint(T).

happens(iECAdw,1).


% === (IHM-UCC) Abrir Secionadora 89AX1
initially(eP03_90(0)).

 terminates(eP03_90up,eP03_90(0),T) :- timepoint(T).
   initiates(eP03_90up,eP03_90(1),T) :- timepoint(T).
 terminates(eP03_90dw,eP03_90(1),T) :- timepoint(T).
   initiates(eP03_90dw,eP03_90(0),T) :- timepoint(T).

happens(eP03_90dw,1).


% === (IHM-COL) Abrir Secionadora 89AX1
initially(iECB(0)).

 terminates(iECBup,iECB(0),T) :- timepoint(T).
   initiates(iECBup,iECB(1),T) :- timepoint(T).
 terminates(iECBdw,iECB(1),T) :- timepoint(T).
   initiates(iECBdw,iECB(0),T) :- timepoint(T).

happens(iECBdw,1).
happens(iECBup,9).
happens(iECBdw,12).


% COLOCAR/RETIRAR CARTÕES:
% === (IHM-UCC) Colocar Cartão Vermelho - Não Opere
initially(eP04_90(0)).

 terminates(eP04_90up,eP04_90(0),T) :- timepoint(T).
   initiates(eP04_90up,eP04_90(1),T) :- timepoint(T).
 terminates(eP04_90dw,eP04_90(1),T) :- timepoint(T).
   initiates(eP04_90dw,eP04_90(0),T) :- timepoint(T).

happens(eP04_90dw,1).


% === (IHM-COL) Colocar Cartão Vermelho - Não Opere
initially(iECC(0)).

 terminates(iECCup,iECC(0),T) :- timepoint(T).
   initiates(iECCup,iECC(1),T) :- timepoint(T).
 terminates(iECCdw,iECC(1),T) :- timepoint(T).
   initiates(iECCdw,iECC(0),T) :- timepoint(T).
```

```
happens(iECCdw,1).


% === (IHM-UCC) Retirar Cartão Vermelho - Não Opere
initially(eP05_90(0)).

 terminates(eP05_90up,eP05_90(0),T) :- timepoint(T).
   initiates(eP05_90up,eP05_90(1),T) :- timepoint(T).
 terminates(eP05_90dw,eP05_90(1),T) :- timepoint(T).
   initiates(eP05_90dw,eP05_90(0),T) :- timepoint(T).

happens(eP05_90dw,1).


% === (IHM-COL) Retirar Cartão Vermelho - Não Opere
initially(iECD(0)).

 terminates(iECDup,iECD(0),T) :- timepoint(T).
   initiates(iECDup,iECD(1),T) :- timepoint(T).
 terminates(iECDdw,iECD(1),T) :- timepoint(T).
   initiates(iECDdw,iECD(0),T) :- timepoint(T).

happens(iECDdw,1).


% === SAÍDAS Circuito F_80 ===

% vI01_80 - Seleção UCC L/R - Remoto
holdsAt(vI01_80(A),T) :- holdsAt(eP01_90(A),T), timepoint(T).

% vI02_80 - Fechar Secionadora 89AX1
holdsAt(vI02_80(R),T) :- or(J,K,R), and(X,B,J), and(A,C,K), inv(A,X),
holdsAt(eP01_90(A),T), holdsAt(eP02_90(B),T), holdsAt(iECA(C),T), nivelLogico(J),
nivelLogico(K), nivelLogico(R), nivelLogico(X), timepoint(T).

% vI03_80 - Abrir Secionadora 89AX1
holdsAt(vI03_80(S),T) :- or(L,M,S), and(X,D,L), and(A,E,M), inv(A,X),
holdsAt(eP01_90(A),T), holdsAt(eP03_90(D),T), holdsAt(iECB(E),T), nivelLogico(L),
nivelLogico(M), nivelLogico(S), nivelLogico(X), timepoint(T).

% vI04_80 - Colocar Cartão Vermelho - Não Opere Secionadora 89AX1
holdsAt(vI04_80(U),T) :- or(N,O,U), and(X,F,N), and(A,G,O), inv(A,X),
holdsAt(eP01_90(A),T), holdsAt(eP04_90(F),T), holdsAt(iECC(G),T), nivelLogico(N),
nivelLogico(O), nivelLogico(U), nivelLogico(X), timepoint(T).

% vI05_80 - Retirar Cartão Vermelho - Não Opere Secionadora 89AX1
```

```
holdsAt(vI05_80(V),T) :- or(P,Q,V), and(X,H,P), and(A,I,M), inv(A,X),
holdsAt(eP05_90(A),T), holdsAt(eP05_90(H),T), holdsAt(iECD(I),T), nivelLogico(P),
nivelLogico(Q), nivelLogico(V), nivelLogico(X), timepoint(T).

%= FIM - Figura 5 - Coqueiros Simplificado F_80  ===



% === Figura 6 - Coqueiros Simplificado F_81 ===

% === Entradas Circuito F_81 ===
% vI01_80 (Saída Circuito F_80)


% COMANDO FECHAR:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)


% === (IHM-UCC) Fechar Secionadora 89AX3
initially(eP06_90(0)).

 terminates(eP06_90up,eP06_90(0),T) :- timepoint(T).
   initiates(eP06_90up,eP06_90(1),T) :- timepoint(T).
 terminates(eP06_90dw,eP06_90(1),T) :- timepoint(T).
   initiates(eP06_90dw,eP06_90(0),T) :- timepoint(T).

happens(eP06_90dw,1).


% === (IHM-COL) Fechar Secionadora 89AX3
initially(iECE(0)).

 terminates(iECEup,iECE(0),T) :- timepoint(T).
   initiates(iECEup,iECE(1),T) :- timepoint(T).
 terminates(iECEdw,iECE(1),T) :- timepoint(T).
   initiates(iECEdw,iECE(0),T) :- timepoint(T).

happens(iECEdw,1).
happens(iECEup,13).
happens(iECEdw,15).


% === (IHM-UCC) Abrir Secionadora 89AX3
initially(eP07_90(0)).

 terminates(eP07_90up,eP07_90(0),T) :- timepoint(T).
   initiates(eP07_90up,eP07_90(1),T) :- timepoint(T).
 terminates(eP07_90dw,eP07_90(1),T) :- timepoint(T).
```

```
  initiates(eP07_90dw,eP07_90(0),T) :- timepoint(T).

happens(eP07_90dw,1).


% === (IHM-COL) Abrir Secionadora 89AX3
initially(iECF(0)).

 terminates(iECFup,iECF(0),T) :- timepoint(T).
  initiates(iECFup,iECF(1),T) :- timepoint(T).
 terminates(iECFdw,iECF(1),T) :- timepoint(T).
  initiates(iECFdw,iECF(0),T) :- timepoint(T).

happens(iECFdw,1).
happens(iECFup,19).
happens(iECFdw,21).


% COLOCAR/RETIRAR CARTÕES:
% === (IHM-UCC) Colocar Cartão Vermelho - Não Opere
initially(eP08_90(0)).

 terminates(eP08_90up,eP08_90(0),T) :- timepoint(T).
  initiates(eP08_90up,eP08_90(1),T) :- timepoint(T).
 terminates(eP08_90dw,eP08_90(1),T) :- timepoint(T).
  initiates(eP08_90dw,eP08_90(0),T) :- timepoint(T).

happens(eP08_90dw,1).


% === (IHM-COL) Colocar Cartão Vermelho - Não Opere
initially(iECG(0)).

 terminates(iECGup,iECG(0),T) :- timepoint(T).
  initiates(iECGup,iECG(1),T) :- timepoint(T).
 terminates(iECGdw,iECG(1),T) :- timepoint(T).
  initiates(iECGdw,iECG(0),T) :- timepoint(T).

happens(iECGdw,1).


% === (IHM-UCC) Retirar Cartão Vermelho - Não Opere
initially(eP09_90(0)).

 terminates(eP09_90up,eP09_90(0),T) :- timepoint(T).
  initiates(eP09_90up,eP09_90(1),T) :- timepoint(T).
 terminates(eP09_90dw,eP09_90(1),T) :- timepoint(T).
  initiates(eP09_90dw,eP09_90(0),T) :- timepoint(T).
```

```
happens(eP09_90dw,1).


% === (IHM-COL) Retirar Cartão Vermelho - Não Opere
initially(iECH(0)).

 terminates(iECHup,iECH(0),T) :- timepoint(T).
   initiates(iECHup,iECH(1),T) :- timepoint(T).
 terminates(iECHdw,iECH(1),T) :- timepoint(T).
   initiates(iECHdw,iECH(0),T) :- timepoint(T).

happens(iECHdw,1).


% === SAÍDAS Circuito F_81 ===

% vI06_81 - Fechar Secionadora 89AX3
holdsAt(vI06_81(R),T) :- or(J,K,R), and(X,B,J), and(A,C,K), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP06_90(B),T), holdsAt(iECE(C),T), nivelLogico(J),
nivelLogico(K), nivelLogico(R), nivelLogico(X), timepoint(T).

% vI07_81 - Abrir Secionadora 89AX3
holdsAt(vI07_81(S),T) :- or(L,M,S), and(X,D,L), and(A,E,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP07_90(D),T), holdsAt(iECF(E),T), nivelLogico(L),
nivelLogico(M), nivelLogico(S), nivelLogico(X), timepoint(T).

% vI08_81 - Colocar Cartão Vermelho - Não Opere Secionadora 89AX3
holdsAt(vI08_81(U),T) :- or(N,O,U), and(X,F,N), and(A,G,O), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP08_90(F),T), holdsAt(iECG(G),T), nivelLogico(N),
nivelLogico(O), nivelLogico(U), nivelLogico(X), timepoint(T).

% vI09_81 - Retirar Cartão Vermelho - Não Opere Secionadora 89AX3
holdsAt(vI09_81(V),T) :- or(P,Q,V), and(X,H,P), and(A,I,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP09_90(H),T), holdsAt(iECH(I),T), nivelLogico(P),
nivelLogico(Q), nivelLogico(V), nivelLogico(X), timepoint(T).

%= FIM - Figura 6 - Coqueiros Simplificado F_81  ===



% === Figura 7 - Coqueiros Simplificado F_82 ===

% === Entradas Circuito F_82 ===
% vI01_80 (Saída Circuito F_80)


% COMANDO FECHAR:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)
```

```
% === (IHM-UCC) Fechar Secionadora 89AX2
initially(eP10_90(0)).

 terminates(eP10_90up,eP10_90(0),T) :- timepoint(T).
   initiates(eP10_90up,eP10_90(1),T) :- timepoint(T).
 terminates(eP10_90dw,eP10_90(1),T) :- timepoint(T).
   initiates(eP10_90dw,eP10_90(0),T) :- timepoint(T).

happens(eP10_90dw,1).


% === (IHM-COL) Fechar Secionadora 89AX2
initially(iECI(0)).

 terminates(iECIup,iECI(0),T) :- timepoint(T).
   initiates(iECIup,iECI(1),T) :- timepoint(T).
 terminates(iECIdw,iECI(1),T) :- timepoint(T).
   initiates(iECIdw,iECI(0),T) :- timepoint(T).

happens(iECIdw,1).


% === (IHM-UCC) Abrir Secionadora 89AX2
initially(eP11_90(0)).

 terminates(eP11_90up,eP11_90(0),T) :- timepoint(T).
   initiates(eP11_90up,eP11_90(1),T) :- timepoint(T).
 terminates(eP11_90dw,eP11_90(1),T) :- timepoint(T).
   initiates(eP11_90dw,eP11_90(0),T) :- timepoint(T).

happens(eP11_90dw,1).


% === (IHM-COL) Abrir Secionadora 89AX2
initially(iECJ(0)).

 terminates(iECJup,iECJ(0),T) :- timepoint(T).
   initiates(iECJup,iECJ(1),T) :- timepoint(T).
 terminates(iECJdw,iECJ(1),T) :- timepoint(T).
   initiates(iECJdw,iECJ(0),T) :- timepoint(T).

happens(iECJdw,1).
happens(iECJup,19).
happens(iECJdw,21).


% COLOCAR/RETIRAR CARTÕES:
```

```
% === (IHM-UCC) Colocar Cartão Vermelho - Não Opere
initially(eP12_90(0)).

 terminates(eP12_90up,eP12_90(0),T) :- timepoint(T).
   initiates(eP12_90up,eP12_90(1),T) :- timepoint(T).
 terminates(eP12_90dw,eP12_90(1),T) :- timepoint(T).
   initiates(eP12_90dw,eP12_90(0),T) :- timepoint(T).

happens(eP12_90dw,1).


% === (IHM-COL) Colocar Cartão Vermelho - Não Opere
initially(iECK(0)).

 terminates(iECKup,iECK(0),T) :- timepoint(T).
   initiates(iECKup,iECK(1),T) :- timepoint(T).
 terminates(iECKdw,iECK(1),T) :- timepoint(T).
   initiates(iECKdw,iECK(0),T) :- timepoint(T).

happens(iECKdw,1).


% === (IHM-UCC) Retirar Cartão Vermelho - Não Opere
initially(eP13_90(0)).

 terminates(eP13_90up,eP13_90(0),T) :- timepoint(T).
   initiates(eP13_90up,eP13_90(1),T) :- timepoint(T).
 terminates(eP13_90dw,eP13_90(1),T) :- timepoint(T).
   initiates(eP13_90dw,eP13_90(0),T) :- timepoint(T).

happens(eP13_90dw,1).


% === (IHM-COL) Retirar Cartão Vermelho - Não Opere
initially(iECL(0)).

 terminates(iECLup,iECL(0),T) :- timepoint(T).
   initiates(iECLup,iECL(1),T) :- timepoint(T).
 terminates(iECLdw,iECL(1),T) :- timepoint(T).
   initiates(iECLdw,iECL(0),T) :- timepoint(T).

happens(iECLdw,1).


% === SAÍDAS Circuito F_82 ===

% vI10_82 - Fechar Secionadora 89AX2
```

```
holdsAt(vI10_82(R),T) :- or(J,K,R), and(X,B,J), and(A,C,K), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP10_90(B),T), holdsAt(iECI(C),T), nivelLogico(J),
nivelLogico(K), nivelLogico(R), nivelLogico(X), timepoint(T).

% vI11_82 - Abrir Secionadora 89AX2
holdsAt(vI11_82(S),T) :- or(L,M,S), and(X,D,L), and(A,E,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP11_90(D),T), holdsAt(iECJ(E),T), nivelLogico(L),
nivelLogico(M), nivelLogico(S), nivelLogico(X), timepoint(T).

% vI12_82 - Colocar Cartão Vermelho - Não Opere Secionadora 89AX2
holdsAt(vI12_82(U),T) :- or(N,O,U), and(X,F,N), and(A,G,O), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP12_90(F),T), holdsAt(iECK(G),T), nivelLogico(N),
nivelLogico(O), nivelLogico(U), nivelLogico(X), timepoint(T).

% vI13_82 - Retirar Cartão Vermelho - Não Opere Secionadora 89AX2
holdsAt(vI13_82(V),T) :- or(P,Q,V), and(X,H,P), and(A,I,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP13_90(H),T), holdsAt(iECL(I),T), nivelLogico(P),
nivelLogico(Q), nivelLogico(V), nivelLogico(X), timepoint(T).

%= FIM - Figura 7 - Coqueiros Simplificado F_82  ===



% === Figura 8 - Coqueiros Simplificado F_83 ===

% === Entradas Circuito F_83 ===
% vI01_80 (Saída Circuito F_80)


% COMANDO FECHAR:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)


% === (IHM-UCC) Fechar Secionadora 57AX
initially(eP14_90(0)).

 terminates(eP14_90up,eP14_90(0),T) :- timepoint(T).
   initiates(eP14_90up,eP14_90(1),T) :- timepoint(T).
 terminates(eP14_90dw,eP14_90(1),T) :- timepoint(T).
   initiates(eP14_90dw,eP14_90(0),T) :- timepoint(T).

happens(eP14_90dw,1).


% === (IHM-COL) Fechar Secionadora 57AX
initially(iECM(0)).

 terminates(iECMup,iECM(0),T) :- timepoint(T).
```

```
  initiates(iECMup,iECM(1),T) :- timepoint(T).
 terminates(iECMdw,iECM(1),T) :- timepoint(T).
  initiates(iECMdw,iECM(0),T) :- timepoint(T).

happens(iECMdw,1).


% === (IHM-UCC) Abrir Secionadora 57AX
initially(eP15_90(0)).

 terminates(eP15_90up,eP15_90(0),T) :- timepoint(T).
  initiates(eP15_90up,eP15_90(1),T) :- timepoint(T).
 terminates(eP15_90dw,eP15_90(1),T) :- timepoint(T).
  initiates(eP15_90dw,eP15_90(0),T) :- timepoint(T).

happens(eP15_90dw,1).


% === (IHM-COL) Abrir Secionadora 57AX
initially(iECN(0)).

 terminates(iECNup,iECN(0),T) :- timepoint(T).
  initiates(iECNup,iECN(1),T) :- timepoint(T).
 terminates(iECNdw,iECN(1),T) :- timepoint(T).
  initiates(iECNdw,iECN(0),T) :- timepoint(T).

happens(iECNdw,1).


% COLOCAR/RETIRAR CARTÕES:
% === (IHM-UCC) Colocar Cartão Vermelho - Não Opere
initially(eP16_90(0)).

 terminates(eP16_90up,eP16_90(0),T) :- timepoint(T).
  initiates(eP16_90up,eP16_90(1),T) :- timepoint(T).
 terminates(eP16_90dw,eP16_90(1),T) :- timepoint(T).
  initiates(eP16_90dw,eP16_90(0),T) :- timepoint(T).

happens(eP16_90dw,1).


% === (IHM-COL) Colocar Cartão Vermelho - Não Opere
initially(iECO(0)).

 terminates(iECOup,iECO(0),T) :- timepoint(T).
  initiates(iECOup,iECO(1),T) :- timepoint(T).
 terminates(iECOdw,iECO(1),T) :- timepoint(T).
  initiates(iECOdw,iECO(0),T) :- timepoint(T).
```

```
happens(iECOdw,1).


% === (IHM-UCC) Retirar Cartão Vermelho - Não Opere
initially(eP17_90(0)).

 terminates(eP17_90up,eP17_90(0),T) :- timepoint(T).
   initiates(eP17_90up,eP17_90(1),T) :- timepoint(T).
 terminates(eP17_90dw,eP17_90(1),T) :- timepoint(T).
   initiates(eP17_90dw,eP17_90(0),T) :- timepoint(T).

happens(eP17_90dw,1).


% === (IHM-COL) Retirar Cartão Vermelho - Não Opere
initially(iECP(0)).

 terminates(iECPup,iECP(0),T) :- timepoint(T).
   initiates(iECPup,iECP(1),T) :- timepoint(T).
 terminates(iECPdw,iECP(1),T) :- timepoint(T).
   initiates(iECPdw,iECP(0),T) :- timepoint(T).

happens(iECPdw,1).


% === SAÍDAS Circuito F_83 ===

% vI14_83 - Fechar Secionadora 57AX
holdsAt(vI14_83(R),T) :- or(J,K,R), and(X,B,J), and(A,C,K), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP14_90(B),T), holdsAt(iECM(C),T), nivelLogico(J),
nivelLogico(K), nivelLogico(R), nivelLogico(X), timepoint(T).

% vI15_83 - Abrir Secionadora 57AX
holdsAt(vI15_83(S),T) :- or(L,M,S), and(X,D,L), and(A,E,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP15_90(D),T), holdsAt(iECN(E),T), nivelLogico(L),
nivelLogico(M), nivelLogico(S), nivelLogico(X), timepoint(T).

% vI16_83 - Colocar Cartão Vermelho - Não Opere Secionadora 57AX
holdsAt(vI16_83(U),T) :- or(N,O,U), and(X,F,N), and(A,G,O), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP16_90(F),T), holdsAt(iECO(G),T), nivelLogico(N),
nivelLogico(O), nivelLogico(U), nivelLogico(X), timepoint(T).

% vI17_83 - Retirar Cartão Vermelho - Não Opere Secionadora 57AX
holdsAt(vI17_83(V),T) :- or(P,Q,V), and(X,H,P), and(A,I,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP17_90(H),T), holdsAt(iECP(I),T), nivelLogico(P),
nivelLogico(Q), nivelLogico(V), nivelLogico(X), timepoint(T).

%= FIM - Figura 8 - Coqueiros Simplificado F_83  ===
```

```
% === Figura 9 - Coqueiros Simplificado F_84 ===

% === Entradas Circuito F_84 ===
% vI01_80 (Saída Circuito F_80)


% COMANDO FECHAR:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)


% === (IHM-UCC) Fechar Secionadora 89AX4
initially(eP18_90(0)).

 terminates(eP18_90up,eP18_90(0),T) :- timepoint(T).
   initiates(eP18_90up,eP18_90(1),T) :- timepoint(T).
 terminates(eP18d_90w,eP18_90(1),T) :- timepoint(T).
   initiates(eP18_90dw,eP18_90(0),T) :- timepoint(T).

happens(eP18_90dw,1).


% === (IHM-COL) Fechar Secionadora 89AX4
initially(iECQ(0)).

 terminates(iECQup,iECQ(0),T) :- timepoint(T).
   initiates(iECQup,iECQ(1),T) :- timepoint(T).
 terminates(iECQdw,iECQ(1),T) :- timepoint(T).
   initiates(iECQdw,iECQ(0),T) :- timepoint(T).

happens(iECQdw,1).


% === (IHM-UCC) Abrir Secionadora 89AX4
initially(eP19_90(0)).

 terminates(eP19_90up,eP19_90(0),T) :- timepoint(T).
   initiates(eP19_90up,eP19_90(1),T) :- timepoint(T).
 terminates(eP19_90dw,eP19_90(1),T) :- timepoint(T).
   initiates(eP19_90dw,eP19_90(0),T) :- timepoint(T).

happens(eP19_90dw,1).


% === (IHM-COL) Abrir Secionadora 89AX4
initially(iECR(0)).
```

```
 terminates(iECRup,iECR(0),T) :- timepoint(T).
   initiates(iECRup,iECR(1),T) :- timepoint(T).
 terminates(iECRdw,iECR(1),T) :- timepoint(T).
   initiates(iECRdw,iECR(0),T) :- timepoint(T).

happens(iECRdw,1).


% COLOCAR/RETIRAR CARTÕES:
% === (IHM-UCC) Colocar Cartão Vermelho - Não Opere
initially(eP20_90(0)).

 terminates(eP20_90up,eP20_90(0),T) :- timepoint(T).
   initiates(eP20_90up,eP20_90(1),T) :- timepoint(T).
 terminates(eP20_90dw,eP20_90(1),T) :- timepoint(T).
   initiates(eP20_90dw,eP20_90(0),T) :- timepoint(T).

happens(eP20_90dw,1).


% === (IHM-COL) Colocar Cartão Vermelho - Não Opere
initially(iECS(0)).

 terminates(iECSup,iECS(0),T) :- timepoint(T).
   initiates(iECSup,iECS(1),T) :- timepoint(T).
 terminates(iECSdw,iECS(1),T) :- timepoint(T).
   initiates(iECSdw,iECS(0),T) :- timepoint(T).

happens(iECSdw,1).


% === (IHM-UCC) Retirar Cartão Vermelho - Não Opere
initially(eP21_90(0)).

 terminates(eP21_90up,eP21_90(0),T) :- timepoint(T).
   initiates(eP21_90up,eP21_90(1),T) :- timepoint(T).
 terminates(eP21_90dw,eP21_90(1),T) :- timepoint(T).
   initiates(eP21_90dw,eP21_90(0),T) :- timepoint(T).

happens(eP21_90dw,1).


% === (IHM-COL) Retirar Cartão Vermelho - Não Opere
initially(iECT(0)).

 terminates(iECTup,iECT(0),T) :- timepoint(T).
   initiates(iECTup,iECT(1),T) :- timepoint(T).
 terminates(iECTdw,iECT(1),T) :- timepoint(T).
   initiates(iECTdw,iECT(0),T) :- timepoint(T).
```

```
happens(iECTdw,1).


% === SAÍDAS Circuito F_84 ===

% vI18_84 - Fechar Secionadora 89AX4
holdsAt(vI18_84(R),T) :- or(J,K,R), and(X,B,J), and(A,C,K), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP18_90(B),T), holdsAt(iECQ(C),T), nivelLogico(J),
nivelLogico(K), nivelLogico(R), nivelLogico(X), timepoint(T).

% vI19_84 - Abrir Secionadora 89AX4
holdsAt(vI19_84(S),T) :- or(L,M,S), and(X,D,L), and(A,E,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP19_90(D),T), holdsAt(iECR(E),T), nivelLogico(L),
nivelLogico(M), nivelLogico(S), nivelLogico(X), timepoint(T).

% vI20_84 - Colocar Cartão Vermelho - Não Opere Secionadora 89AX4
holdsAt(vI20_84(U),T) :- or(N,O,U), and(X,F,N), and(A,G,O), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP20_90(F),T), holdsAt(iECS(G),T), nivelLogico(N),
nivelLogico(O), nivelLogico(U), nivelLogico(X), timepoint(T).

% vI21_84 - Retirar Cartão Vermelho - Não Opere Secionadora 89AX4
holdsAt(vI21_84(V),T) :- or(P,Q,V), and(X,H,P), and(A,I,M), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP21_90(H),T), holdsAt(iECT(I),T), nivelLogico(P),
nivelLogico(Q), nivelLogico(V), nivelLogico(X), timepoint(T).

%= FIM - Figura 9 - Coqueiros Simplificado F_84  ===



% === Figura 10 - Coqueiros Simplificado F_85 ===

% === Entradas Circuito F_85 ===
% vI01_80 (Saída Circuito F_80)
% vI68_29 (Religamento Disjuntor 52AX - Ligado - Saída Circuito F_29)


% COMANDO FECHAR:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)


% === (IHM-UCC) Fechar Disjuntor 52AX
initially(eP22_90(0)).

 terminates(eP22_90up,eP22_90(0),T) :- timepoint(T).
   initiates(eP22_90up,eP22_90(1),T) :- timepoint(T).
 terminates(eP22_90dw,eP22_90(1),T) :- timepoint(T).
   initiates(eP22_90dw,eP22_90(0),T) :- timepoint(T).
```

```
happens(eP22_90dw,1).


% === (IHM-COL) Fechar Disjuntor 52AX
initially(iECU(0)).

 terminates(iECUup,iECU(0),T) :- timepoint(T).
   initiates(iECUup,iECU(1),T) :- timepoint(T).
 terminates(iECUdw,iECU(1),T) :- timepoint(T).
   initiates(iECUdw,iECU(0),T) :- timepoint(T).

happens(iECUdw,1).
happens(iECUup,16).
happens(iECUdw,19).


% === (IHM-UCC) Abrir Disjuntor 52AX
initially(eP23_90(0)).

 terminates(eP23_90up,eP23_90(0),T) :- timepoint(T).
   initiates(eP23_90up,eP23_90(1),T) :- timepoint(T).
 terminates(eP23_90dw,eP23_90(1),T) :- timepoint(T).
   initiates(eP23_90dw,eP23_90(0),T) :- timepoint(T).

happens(eP23_90dw,1).


% === (IHM-COL) Abrir Disjuntor 52AX
initially(iECV(0)).

 terminates(iECVup,iECV(0),T) :- timepoint(T).
   initiates(iECVup,iECV(1),T) :- timepoint(T).
 terminates(iECVdw,iECV(1),T) :- timepoint(T).
   initiates(iECVdw,iECV(0),T) :- timepoint(T).

happens(iECVdw,1).
happens(iECVup,6).
happens(iECVdw,8).


% COLOCAR/RETIRAR CARTÕES:
% === (IHM-UCC) Colocar Cartão Amarelo - Linha Viva
initially(eP24_90(0)).

 terminates(eP24_90up,eP24_90(0),T) :- timepoint(T).
   initiates(eP24_90up,eP24_90(1),T) :- timepoint(T).
 terminates(eP24_90dw,eP24_90(1),T) :- timepoint(T).
   initiates(eP24_90dw,eP24_90(0),T) :- timepoint(T).
```

```
happens(eP24_90dw,1).


% === (IHM-COL) Colocar Cartão Amarelo - Linha Viva
initially(iECW(0)).

 terminates(iECWup,iECW(0),T) :- timepoint(T).
   initiates(iECWup,iECW(1),T) :- timepoint(T).
 terminates(iECWdw,iECW(1),T) :- timepoint(T).
   initiates(iECWdw,iECW(0),T) :- timepoint(T).

happens(iECWdw,1).


% === (IHM-UCC) Retirar Cartão Amarelo - Linha Viva
initially(eP25_90(0)).

 terminates(eP25_90up,eP25_90(0),T) :- timepoint(T).
   initiates(eP25_90up,eP25_90(1),T) :- timepoint(T).
 terminates(eP25_90dw,eP25_90(1),T) :- timepoint(T).
   initiates(eP25_90dw,eP25_90(0),T) :- timepoint(T).

happens(eP25_90dw,1).


% === (IHM-COL) Retirar Cartão Amarelo - Linha Viva
initially(iECX(0)).

 terminates(iECXup,iECX(0),T) :- timepoint(T).
   initiates(iECXup,iECX(1),T) :- timepoint(T).
 terminates(iECXdw,iECX(1),T) :- timepoint(T).
   initiates(iECXdw,iECX(0),T) :- timepoint(T).

happens(iECXdw,1).


% === (IHM-UCC) Colocar Cartão Vermelho - Não Opere
initially(eP26_90(0)).

 terminates(eP26_90up,eP26_90(0),T) :- timepoint(T).
   initiates(eP26_90up,eP26_90(1),T) :- timepoint(T).
 terminates(eP26_90dw,eP26_90(1),T) :- timepoint(T).
   initiates(eP26_90dw,eP26_90(0),T) :- timepoint(T).

happens(eP26_90dw,1).


% === (IHM-COL) Colocar Cartão Vermelho - Não Opere
```

```
initially(iECY(0)).

 terminates(iECYup,iECY(0),T) :- timepoint(T).
   initiates(iECYup,iECY(1),T) :- timepoint(T).
 terminates(iECYdw,iECY(1),T) :- timepoint(T).
   initiates(iECYdw,iECY(0),T) :- timepoint(T).

happens(iECYdw,1).


% === (IHM-UCC) Retirar Cartão Vermelho - Não Opere
initially(eP27_90(0)).

 terminates(eP27_90up,eP27_90(0),T) :- timepoint(T).
   initiates(eP27_90up,eP27_90(1),T) :- timepoint(T).
 terminates(eP27_90dw,eP27_90(1),T) :- timepoint(T).
   initiates(eP27_90dw,eP27_90(0),T) :- timepoint(T).

happens(eP27_90dw,1).


% === (IHM-COL) Retirar Cartão Vermelho - Não Opere
initially(iECZ(0)).

 terminates(iECZup,iECZ(0),T) :- timepoint(T).
   initiates(iECZup,iECZ(1),T) :- timepoint(T).
 terminates(iECZdw,iECZ(1),T) :- timepoint(T).
   initiates(iECZdw,iECZ(0),T) :- timepoint(T).

happens(iECZdw,1).


% === SAÍDAS Circuito F_85 ===

% vI22_85 - Fechar Disjuntor 52AX
holdsAt(vI22_85(AA),T) :- or(N,O,AA), and(X,B,N), and(A,C,O), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP22_90(B),T), holdsAt(iECU(C),T), nivelLogico(N),
nivelLogico(O), nivelLogico(AA), nivelLogico(X), timepoint(T).

% vI23_85 - Abrir Disjuntor 52AX
holdsAt(vI22_85(AB),T) :- or(P,Q,AB), and(X,D,P), and(A,E,Q), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP23_90(D),T), holdsAt(iECV(E),T), nivelLogico(P),
nivelLogico(Q), nivelLogico(AB), nivelLogico(X), timepoint(T).

% vI24_85 - Colocar Cartão Amarelo - Linha Viva Disjuntor 52AX
holdsAt(vI24_85(AC),T) :- or(R,S,AC), and3(X,Y,F,R), and3(A,Y,G,S), inv(A,X),
inv(ZZ,Y), holdsAt(vI01_80(A),T), holdsAt(vI68_29(ZZ),T), holdsAt(eP24_90(F),T),
holdsAt(iECW(G),T), nivelLogico(R), nivelLogico(S), nivelLogico(AC),
nivelLogico(X), timepoint(T).
```

```
% vI25_85 - Retirar Cartão Amarelo - Linha Viva Disjuntor 52AX
holdsAt(vI25_85(AD),T) :- or(U,V,AD), and(X,H,U), and(A,I,V), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP25_90(H),T), holdsAt(iECX(I),T), nivelLogico(U),
nivelLogico(V), nivelLogico(AD), nivelLogico(X), timepoint(T).

% vI26_85 - Colocar Cartão Vermelho - Não Opere Disjuntor 52AX
holdsAt(vI26_85(AE),T) :- or(W,X,AE), and(B,J,W), and(A,K,X), inv(A,B),
holdsAt(vI01_80(A),T), holdsAt(eP26_90(J),T), holdsAt(iECY(K),T), nivelLogico(W),
nivelLogico(X), nivelLogico(AE), nivelLogico(B), timepoint(T).

% vI27_85 - Retirar Cartão Vermelho - Não Opere Disjuntor 52AX
holdsAt(vI27_85(AF),T) :- or(Y,Z,AF), and(X,L,Y), and(A,M,Z), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP27_90(L),T), holdsAt(iECZ(M),T), nivelLogico(Y),
nivelLogico(Z), nivelLogico(AF), nivelLogico(X), timepoint(T).

%= FIM - Figura 10 - Coqueiros Simplificado F_85  ===



% === Figura 11 - Coqueiros Simplificado F_86 ===

% === Entradas Circuito F_86 ===
% vI01_80 (Saída Circuito F_80)


% DESBLOQUEIO DISJUNTOR 52AX:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)


% === (IHM-UCC) Desbloqueio Disjuntor 52AX
initially(eP28_90(0)).

 terminates(eP28_90up,eP28_90(0),T) :- timepoint(T).
   initiates(eP28_90up,eP28_90(1),T) :- timepoint(T).
 terminates(eP28_90dw,eP28_90(1),T) :- timepoint(T).
   initiates(eP28_90dw,eP28_90(0),T) :- timepoint(T).

happens(eP28_90dw,1).


% === (IHM-COL) Desbloqueio Disjuntor 52AX
initially(iECAA(0)).

 terminates(iECAAup,iECAA(0),T) :- timepoint(T).
   initiates(iECAAup,iECAA(1),T) :- timepoint(T).
 terminates(iECAAdw,iECAA(1),T) :- timepoint(T).
   initiates(iECAAdw,iECAA(0),T) :- timepoint(T).
```

```
happens(iECAAdw,1).


% === SAÍDAS Circuito F_86 ===

% vI28_86 - Fechar Desbloqueio Disjuntor 52AX
holdsAt(vI28_86(F),T) :- or(D,E,F), and(X,B,D), and(A,C,E), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP28_90(B),T), holdsAt(iECAA(C),T),
nivelLogico(D), nivelLogico(E), nivelLogico(F), nivelLogico(X), timepoint(T).

%= FIM - Figura 11 - Coqueiros Simplificado F_86  ===



% === Figura 12 - Coqueiros Simplificado F_87 ===

% === Entradas Circuito F_87 ===

% SELECAO TRANSFERÊNCIA PROTEÇÃO AUTOMÁTICA:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)


% SELEÇÃO PROTEÇÃO NORMAL "N"
% === (IHM-UCC) Seleção Transferência Proteção Normal "N"
initially(ePN_90(0)).

 terminates(ePN_90up,ePN_90(0),T) :- timepoint(T).
   initiates(ePN_90up,ePN_90(1),T) :- timepoint(T).
 terminates(ePN_90dw,ePN_90(1),T) :- timepoint(T).
   initiates(ePN_90dw,ePN_90(0),T) :- timepoint(T).

happens(ePN_90dw,1).


% === (IHM-COL) Seleção Transferência Proteção Normal "N"
initially(iECAB(1)).

 terminates(iECABup,iECAB(0),T) :- timepoint(T).
   initiates(iECABup,iECAB(1),T) :- timepoint(T).
 terminates(iECABdw,iECAB(1),T) :- timepoint(T).
   initiates(iECABdw,iECAB(0),T) :- timepoint(T).

happens(iECABup,1).


% SELEÇÃO PROTEÇÃO EM TRANSFERÊNCIA "ET"
% === (IHM-COL) Seleção Transferência Proteção em Transferência "ET"
```

```
initially(ePET_90(0)).

 terminates(ePET_90up,ePET_90(0),T) :- timepoint(T).
   initiates(ePET_90up,ePET_90(1),T) :- timepoint(T).
 terminates(ePET_90dw,ePET_90(1),T) :- timepoint(T).
   initiates(ePET_90dw,ePET_90(0),T) :- timepoint(T).

happens(ePET_90dw,1).


% === (IHM-COL) Seleção Transferência Proteção em Transferência "ET"
initially(iECAC(0)).

 terminates(iECACup,iECAC(0),T) :- timepoint(T).
   initiates(iECACup,iECAC(1),T) :- timepoint(T).
 terminates(iECACdw,iECAC(1),T) :- timepoint(T).
   initiates(iECACdw,iECAC(0),T) :- timepoint(T).

happens(iECACdw,1).


% SELEÇÃO PROTEÇÃO TRANSFERIDA "T"
% === (IHM-COL) Seleção Transferência Proteção Transferida "T"
initially(ePT_90(0)).

 terminates(ePT_90up,ePT_90(0),T) :- timepoint(T).
   initiates(ePT_90up,ePT_90(1),T) :- timepoint(T).
 terminates(ePT_90dw,ePT_90(1),T) :- timepoint(T).
   initiates(ePT_90dw,ePT_90(0),T) :- timepoint(T).

happens(ePT_90dw,1).


% === (IHM-COL) Seleção Transferência Proteção Transferida "T"
initially(iECAD(0)).

 terminates(iECADup,iECAD(0),T) :- timepoint(T).
   initiates(iECADup,iECAD(1),T) :- timepoint(T).
 terminates(iECADdw,iECAD(1),T) :- timepoint(T).
   initiates(iECADdw,iECAD(0),T) :- timepoint(T).

happens(iECADdw,1).


% === SAÍDAS Circuito F_87 ===

% vI29_87 - Seleção Transferência Proteção Normal "N"
```

```
holdsAt(vI29_87(O),T) :- or(I,J,O), and(NA,C,I), and(A,D,J), inv(A,NA),
holdsAt(vI01_80(A),T), holdsAt(ePN_90(C),T), holdsAt(iECAB(D),T),
nivelLogico(O), nivelLogico(I), nivelLogico(J), nivelLogico(NA), timepoint(T).

% vI30_87 - Seleção Transferência Proteção Normal "ET"
holdsAt(vI30_87(P),T) :- or(K,L,P), and(NA,E,K), and(A,F,L), inv(A,NA),
holdsAt(vI01_80(A),T), holdsAt(ePET_90(E),T), holdsAt(iECAC(F),T),
nivelLogico(P), nivelLogico(K), nivelLogico(L), nivelLogico(NA), timepoint(T).

% vI31_87 - Seleção Transferência Proteção Transferida "T"
holdsAt(vI31_87(Q),T) :- or(M,N,Q), and(NA,G,M), and(A,H,N), inv(A,NA),
holdsAt(vI01_80(A),T), holdsAt(ePT_90(G),T), holdsAt(iECAD(H),T),
nivelLogico(Q), nivelLogico(M), nivelLogico(N), nivelLogico(NA), timepoint(T).

%= FIM - Figura 12 - Coqueiros Simplificado F_87  ===



% === Figura 13 - Coqueiros Simplificado F_88 ===

% === Entradas Circuito F_88 ===

% SELECAO RELIGAMENTO DISJUNTOR 52AX:
% === (IHM-UCC) Seleção UCC L/R - Remoto
% vI01_80 (Saída Circuito F_80)
% vI39_24 (Cartão Amarelo colocado Disjuntor 52AX - Saída Circuito F_24)


% === (IHM-UCC) Selecionar Religamento Monopolar Disjuntor 52AX
initially(eP32_90(0)).

 terminates(eP32_90up,eP32_90(0),T) :- timepoint(T).
   initiates(eP32_90up,eP32_90(1),T) :- timepoint(T).
 terminates(eP32_90dw,eP32_90(1),T) :- timepoint(T).
   initiates(eP32_90dw,eP32_90(0),T) :- timepoint(T).

happens(eP32_90dw,1).


% === (IHM-COL) Selecionar Religamento Monopolar Disjuntor 52AX
initially(iECAE(0)).

 terminates(iECAEup,iECAE(0),T) :- timepoint(T).
   initiates(iECAEup,iECAE(1),T) :- timepoint(T).
 terminates(iECAEdw,iECAE(1),T) :- timepoint(T).
   initiates(iECAEdw,iECAE(0),T) :- timepoint(T).

happens(iECAEdw,1).
```

```
% SELEÇÃO RELIGAMENTO TRIPOLAR DISJUNTOR 52AX
% === (IHM-UCC) Selecionar Religamento Tripolar Disjuntor 52AX
initially(eP33_90(0)).

 terminates(eP33_90up,eP33_90(0),T) :- timepoint(T).
   initiates(eP33_90up,eP33_90(1),T) :- timepoint(T).
 terminates(eP33_90dw,eP33_90(1),T) :- timepoint(T).
   initiates(eP33_90dw,eP33_90(0),T) :- timepoint(T).

happens(eP33_90dw,1).


% === (IHM-COL) Selecionar Religamento Tripolar Disjuntor 52AX
initially(iECAF(1)).

 terminates(iECAFup,iECAF(0),T) :- timepoint(T).
   initiates(iECAFup,iECAF(1),T) :- timepoint(T).
 terminates(iECAFdw,iECAF(1),T) :- timepoint(T).
   initiates(iECAFdw,iECAF(0),T) :- timepoint(T).

happens(iECAFup,1).


% RELIGAMENTO DISJUNTOR 52AX DESLIGAMENTO:
% === (IHM-UCC) Religamento Disjuntor 52AX Desligado
initially(eP34_90(0)).

 terminates(eP34_90up,eP34_90(0),T) :- timepoint(T).
   initiates(eP34_90up,eP34_90(1),T) :- timepoint(T).
 terminates(eP34_90dw,eP34_90(1),T) :- timepoint(T).
   initiates(eP34_90dw,eP34_90(0),T) :- timepoint(T).

happens(eP34_90dw,1).


% === (IHM-COL) Religamento Disjuntor 52AX Desligado
initially(iECAG(0)).

 terminates(iECAGup,iECAG(0),T) :- timepoint(T).
   initiates(iECAGup,iECAG(1),T) :- timepoint(T).
 terminates(iECAGdw,iECAG(1),T) :- timepoint(T).
   initiates(iECAGdw,iECAG(0),T) :- timepoint(T).

happens(iECAGdw,1).


% === SAÍDAS Circuito F_88 ===
```

```
% vI32_88 - Ativar Religamento Monopolar Disjuntor 52AX
holdsAt(vI32_88(O),T) :- or(I,J,O), and3(X,Y,C,I), and3(A,Y,D,J), inv(A,X), inv(B,Y),
holdsAt(vI01_80(A),T), holdsAt(vI39_24(B),T), holdsAt(eP32_90(C),T),
holdsAt(iECAE(D),T), nivelLogico(I), nivelLogico(J), nivelLogico(O), nivelLogico(X),
nivelLogico(Y), timepoint(T).

% vI33_88 - Ativar Religamento Tripolar Disjuntor 52AX
holdsAt(vI33_88(P),T) :- or(K,L,P), and3(X,Y,E,K), and3(A,Y,F,L), inv(A,X),
inv(B,Y), holdsAt(vI01_80(A),T), holdsAt(vI39_24(B),T), holdsAt(eP33_90(E),T),
holdsAt(iECAF(F),T), nivelLogico(K), nivelLogico(L), nivelLogico(P),
nivelLogico(X), nivelLogico(Y), timepoint(T).

% vI34_88 - Religamento Disjuntor 52AX Desligado
holdsAt(vI34_88(Q),T) :- or(M,N,Q), and(X,G,M), and(A,H,N), inv(A,X),
holdsAt(vI01_80(A),T), holdsAt(eP34_90(G),T), holdsAt(iECAG(H),T),
nivelLogico(M), nivelLogico(N), nivelLogico(Q), nivelLogico(X), timepoint(T).

%= FIM - Figura 13 - Coqueiros Simplificado F_88  ===



% === Figura 14 - Coqueiros Simplificado F_53 ===

% === Entradas Circuito F_53 ===

% === IM64 CH1 INPUT4 - RECEPÇÃO TDD MANTIDO
initially(dDB99(0)).

 terminates(dDB99up,dDB99(0),T) :- timepoint(T).
   initiates(dDB99up,dDB99(1),T) :- timepoint(T).
 terminates(dDB99dw,dDB99(1),T) :- timepoint(T).
   initiates(dDB99dw,dDB99(0),T) :- timepoint(T).

happens(dDB99dw,1).

% === TRIP-59I
initially(dDB691(0)).

 terminates(dDB691up,dDB691(0),T) :- timepoint(T).
   initiates(dDB691up,dDB691(1),T) :- timepoint(T).
 terminates(dDB691dw,dDB691(1),T) :- timepoint(T).
   initiates(dDB691dw,dDB691(0),T) :- timepoint(T).

happens(dDB691dw,1).
happens(dDB691up,18).

% === TRIP-59T
initially(dDB695(0)).
```

```
 terminates(dDB695up,dDB695(0),T) :- timepoint(T).
   initiates(dDB695up,dDB695(1),T) :- timepoint(T).
 terminates(dDB695dw,dDB695(1),T) :- timepoint(T).
   initiates(dDB695dw,dDB695(0),T) :- timepoint(T).

happens(dDB695dw,1).

% === OPEROU PROTEÇÃO FALHA DISJUNTOR 52AX/PROTEÇÃO EFP
initially(oPTO12(0)).

 terminates(oPTO12up,oPTO12(0),T) :- timepoint(T).
   initiates(oPTO12up,oPTO12(1),T) :- timepoint(T).
 terminates(oPTO12dw,oPTO12(1),T) :- timepoint(T).
   initiates(oPTO12dw,oPTO12(0),T) :- timepoint(T).

happens(oPTO12dw,1).
happens(oPTO12up,18).

% === SAÍDAS Circuito F_53 ===

% v01_53 - Virtual Output 1 - Bloqueio Fechamento Disjuntor 52AX (UCD1/UCDR)
holdsAt(v01_53(C),T) :- or(A,B,C), holdsAt(dDB691(A),T), holdsAt(dDB695(B),T),
nivelLogico(C), timepoint(T).

% rELAY1_53 - [453] Partida Falha Disjuntor 52AX Trifásico
holdsAt(rELAY1_53(C),T) :- or(A,B,C), holdsAt(dDB99(A),T),
holdsAt(v01_53(B),T), nivelLogico(C), timepoint(T).

% dDB115_53 - IM64 CH1 OUTPUT4 Envio TDD Mantido
holdsAt(dDB115_53(C),T) :- or(A,B,C), holdsAt(v01_53(A),T),
holdsAt(oPTO12(B),T), nivelLogico(C), timepoint(T).

%= FIM - Figura 14 - Coqueiros Simplificado F_53  ===



% === Figura 15 - Coqueiros Simplificado F_63 ===

% === Entradas Circuito F_63 ===
% Mesmas da F_53

% === SAÍDAS Circuito F_63 ===

% v01_63 - Virtual Output 1 - Bloqueio Fechamento Disjuntor 52AX (UCD1/UCDR)
holdsAt(v01_63(C),T) :- or(A,B,C), holdsAt(dDB691(A),T), holdsAt(dDB695(B),T),
nivelLogico(C), timepoint(T).

% rELAY1_63 - [463] Partida Falha Disjuntor 52AX Trifásico
```

```
holdsAt(rELAY1_63(C),T) :- or(A,B,C), holdsAt(dDB99(A),T),
holdsAt(v01_63(B),T), nivelLogico(C), timepoint(T).

% dDB115_63 - IM64 CH1 OUTPUT4 Envio TDD Mantido
holdsAt(dDB115_63(C),T) :- or(A,B,C), holdsAt(v01_63(A),T),
holdsAt(oPTO12(B),T), nivelLogico(C), timepoint(T).

%= FIM - Figura 15 - Coqueiros Simplificado F_63  ===



% Circuito-Comum à algumas figuras!

% === Falta VCA Motor
initially(dI11O(0)).

 terminates(dI11Oup,dI11O(0),T) :- timepoint(T).
   initiates(dI11Oup,dI11O(1),T) :- timepoint(T).
 terminates(dI11Odw,dI11O(1),T) :- timepoint(T).
   initiates(dI11Odw,dI11O(0),T) :- timepoint(T).

happens(dI11Odw,1).


% === Falta VCC Cont. Secionadora
initially(dI5M(0)).

 terminates(dI5Mup,dI5M(0),T) :- timepoint(T).
   initiates(dI5Mup,dI5M(1),T) :- timepoint(T).
 terminates(dI5Mdw,dI5M(1),T) :- timepoint(T).
   initiates(dI5Mdw,dI5M(0),T) :- timepoint(T).

happens(dI5Mdw,1).


% === Falta VCC Cont. Secionadora
initially(dI9O(0)).

 terminates(dI9Oup,dI9O(0),T) :- timepoint(T).
   initiates(dI9Oup,dI9O(1),T) :- timepoint(T).
 terminates(dI9Odw,dI9O(1),T) :- timepoint(T).
   initiates(dI9Odw,dI9O(0),T) :- timepoint(T).

happens(dI9Odw,1).


% === SAÍDAS Circuito-Comum ===
```

```
holdsAt(fVCC_VCA(E),T) :- inv(D,E), or3(A,B,C,D), holdsAt(dI11O(A),T),
holdsAt(dI5M(B),T), holdsAt(dI9O(C),T), nivelLogico(D), nivelLogico(E),
timepoint(T).

%= FIM - Circuito-Comum à algumas figuras!



% === Figura 16 - Coqueiros Simplificado F_19 ===

% === Entradas Circuito F_19 ===
% vI02_80 (Fechar Secionadora 89AX1 - Saída Circuito F_80)
% vI14_17 (Secionadora 89AX1 Fechada - Saída Circuito F_17)
% vI04_80 (Colocar Cartão Vermelho - Não Opere Secionadora 89AX1 - Saída
Circuito F_80)
% vI05_80 (Retirar Cartão Vermelho - Não Opere Secionadora 89Ax1 - Saída Circuito
F_80)
% vI18_17 (Secionadora 89AX3 Fechada - Saída Circuito F_17)
% vI09_16 (Secionadora 89AX4 Aberta - Saída Circuito F_16)
% vI21_18 (Disjuntor 52AX Aberto - Saída Circuito F_18)
% vI17_17 (Secionadora 89AX3 Aberta - Saída Circuito F_17)
% vI41_25 (Desbloqueio Por Falha Disjuntor 52AX - Saída Circuito F_25)
% vI13_17 (Secionadora 89AX1 Aberta - Saída Circuito F_17)
% vI03_80 (Abrir Secionadora 89AX1 - Saída Circuito F_80)
% fVCC_VCA (Falta VCA/VCC Motor - Comum)

% === (IHM-UCD1) Seleção LOCAL/REMOTO-REMOTO
initially(eP01_32(1)).

 terminates(eP01_32up,eP01_32(0),T) :- timepoint(T).
   initiates(eP01_32up,eP01_32(1),T) :- timepoint(T).
 terminates(eP01_32dw,eP01_32(1),T) :- timepoint(T).
   initiates(eP01_32dw,eP01_32(0),T) :- timepoint(T).

happens(eP01_32up,1).


% === (IHM-UCD1) Fechar Secionadora 89AX1
initially(eP02_32(0)).

 terminates(eP02_32up,eP02_32(0),T) :- timepoint(T).
   initiates(eP02_32up,eP02_32(1),T) :- timepoint(T).
 terminates(eP02_32dw,eP02_32(1),T) :- timepoint(T).
   initiates(eP02_32dw,eP02_32(0),T) :- timepoint(T).

happens(eP02_32dw,1).


% === (IHM-UCD1) Retirar Cartões
```

```
initially(eP03_32(0)).

 terminates(eP03_32up,eP03_32(0),T) :- timepoint(T).
   initiates(eP03_32up,eP03_32(1),T) :- timepoint(T).
 terminates(eP03_32dw,eP03_32(1),T) :- timepoint(T).
   initiates(eP03_32dw,eP03_32(0),T) :- timepoint(T).

happens(eP03_32dw,1).


% === (IHM-UCD1) Secionadora 89AX1 Chave 43 L/R-Remoto
initially(dI14O(1)).

 terminates(dI14Oup,dI14O(0),T) :- timepoint(T).
   initiates(dI14Oup,dI14O(1),T) :- timepoint(T).
 terminates(dI14Odw,dI14O(1),T) :- timepoint(T).
   initiates(dI14Odw,dI14O(0),T) :- timepoint(T).

happens(dI14Oup,1).


% === (IHM-UCD1) Abrir Secionadora 89AX1
initially(eP04_32(0)).

 terminates(eP04_32up,eP04_32(0),T) :- timepoint(T).
   initiates(eP04_32up,eP04_32(1),T) :- timepoint(T).
 terminates(eP04_32dw,eP04_32(1),T) :- timepoint(T).
   initiates(eP04_32dw,eP04_32(0),T) :- timepoint(T).

happens(eP04_32dw,1).


% === Disjuntor 52AX - Baixa Pressão SF6 2o Estágio
initially(dI12N_405(0)).

 terminates(dI12N_405up,dI12N_405(0),T) :- timepoint(T).
   initiates(dI12N_405up,dI12N_405(1),T) :- timepoint(T).
 terminates(dI12N_405dw,dI12N_405(1),T) :- timepoint(T).
   initiates(dI12N_405dw,dI12N_405(0),T) :- timepoint(T).

happens(dI12N_405dw,1).

%============ Flip-flop SR Circuito F19_1 ============
initially(qout_CF19_1(0)).
happens(i2_CF19_1_sup,T1) :- holdsAt(sin_CF19_1(0),T1),
holdsAt(sin_CF19_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF19_1_sdw,T1) :- holdsAt(sin_CF19_1(1),T1),
holdsAt(sin_CF19_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
```

```
happens(i2_CF19_1_rup,T1) :- holdsAt(rin_CF19_1(0),T1),
holdsAt(rin_CF19_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF19_1_rdw,T1) :- holdsAt(rin_CF19_1(1),T1),
holdsAt(rin_CF19_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF19_1_sup,qout_CF19_1(0),T) :- holdsAt(rin_CF19_1(0),T),
holdsAt(qout_CF19_1(0),T), timepoint(T).
 initiates(i2_CF19_1_sup,qout_CF19_1(1),T) :- holdsAt(rin_CF19_1(0),T),
holdsAt(qout_CF19_1(0),T), timepoint(T).
terminates(i2_CF19_1_rup,qout_CF19_1(1),T) :- holdsAt(sin_CF19_1(0),T),
holdsAt(qout_CF19_1(1),T), timepoint(T).
 initiates(i2_CF19_1_rup,qout_CF19_1(0),T) :- holdsAt(sin_CF19_1(0),T),
holdsAt(qout_CF19_1(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF19_1_rup,qout_CF19_1(1),T) :- holdsAt(sin_CF19_1(1),T),
holdsAt(qout_CF19_1(1),T), timepoint(T).
 initiates(i2_CF19_1_rup,qout_CF19_1(0),T) :- holdsAt(sin_CF19_1(1),T),
holdsAt(qout_CF19_1(1),T), timepoint(T).

%============ Pick-up timer Circuito F19_1 ============
initially(monoOutput_PU_CF19_1(0)).
happens(upTriger_PU_CF19_1,T3) :- holdsAt(inputIs_PU_CF19_1(0),T1),
holdsAt(inputIs_PU_CF19_1(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF19_1,monoOutput_PU_CF19_1(0),T) :-
happens(upTriger_PU_CF19_1,T), holdsAt(inputIs_PU_CF19_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF19_1,monoOutput_PU_CF19_1(1),T)  :-
happens(upTriger_PU_CF19_1,T), holdsAt(inputIs_PU_CF19_1(1),T), timepoint(T).

happens(downTriger_PU_CF19_1,0) :- initially(inputIs_PU_CF19_1(0)).
happens(downTriger_PU_CF19_1,T1) :- holdsAt(inputIs_PU_CF19_1(1),T1),
holdsAt(inputIs_PU_CF19_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF19_1,monoOutput_PU_CF19_1(1),T) :-
happens(downTriger_PU_CF19_1,T), holdsAt(monoOutput_PU_CF19_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF19_1,monoOutput_PU_CF19_1(0),T) :-
happens(downTriger_PU_CF19_1,T), holdsAt(monoOutput_PU_CF19_1(1),T),
timepoint(T).


% === SAÍDAS Circuito F_19 ===
% Partes comuns a algumas saídas
holdsAt(sin_CF19_1(U),T) :- and(B,E,U), holdsAt(eP01_32(B),T),
holdsAt(vI04_80(E),T), nivelLogico(U), timepoint(T).
holdsAt(rin_CF19_1(X),T) :- or(V,W,X), and(B,F,V), and(NB,G,W), inv(B,NB),
holdsAt(eP01_32(B),T), holdsAt(vI05_80(F),T), holdsAt(eP03_32(G),T),
nivelLogico(V), nivelLogico(W), nivelLogico(X), nivelLogico(NB), timepoint(T).
holdsAt(inputIs_PU_CF19_1(NZF),T) :- inv(ZF,NZF), and(N,Q,ZF),
holdsAt(vI13_17(N),T), holdsAt(dI12N_405(Q),T), nivelLogico(ZF),
nivelLogico(NZF), timepoint(T).
```

```
holdsAt(zN_19(ZN),T) :- holdsAt(monoOutput_PU_CF19_1(ZN),T), timepoint(T).
holdsAt(zE_19(ZE),T) :- or4(ZA,ZB,ZO,M,ZE), and(I,J,ZA), and(K,L,ZB),
and(K,Q,ZO), holdsAt(vI18_17(I),T), holdsAt(vI09_16(J),T), holdsAt(vI21_18(K),T),
holdsAt(vI17_17(L),T), holdsAt(dI12N_405(Q),T), holdsAt(vI41_25(M),T),
nivelLogico(ZA), nivelLogico(ZB), nivelLogico(ZO), nivelLogico(ZE), timepoint(T).
holdsAt(zH_19(ZH),T) :- and4(NZ,H,ZE,ZN,ZH), inv(Z,NZ),
holdsAt(qout_CF19_1(Z),T), holdsAt(dI14O(H),T), holdsAt(zE_19(ZE),T),
holdsAt(zN_19(ZN),T), nivelLogico(ZH), nivelLogico(NZ), timepoint(T).

% dO3D - Permissão de Manobra Secionadora 89AX1
holdsAt(dO3D(ZL),T) :- and3(ZN,NZP,ZE,ZL), holdsAt(zN_19(ZN),T),
holdsAt(fVCC_VCA(NZP),T), holdsAt(zE_19(ZE),T), nivelLogico(ZL), timepoint(T).

% vI24_19 - Permissão Fechar Secionadora 89AX1
holdsAt(vI24_19(ZI),T) :- and3(ND,NZP,ZH,ZI), inv(D,ND), inv(ZP,NZP),
holdsAt(vI14_17(D),T), holdsAt(fVCC_VCA(NZP),T), holdsAt(zH_19(ZH),T),
nivelLogico(ZI), nivelLogico(ND), timepoint(T).

% dO2D - Fechar Secionadora 89AX1
holdsAt(do2D(ZK),T) :- and(Y,ZI,ZK), or(R,S,Y), and(A,B,R), and(NB,C,S),
inv(B,NB), holdsAt(vI02_80(A),T), holdsAt(eP01_32(B),T), holdsAt(eP02_32(C),T),
holdsAt(vI24_19(ZI),T), nivelLogico(ZK), nivelLogico(Y), nivelLogico(R),
nivelLogico(S), nivelLogico(NB), timepoint(T).

% vI25_19 - Permissão para abrir Secionadora 89AX1 (IHM-UCC)
holdsAt(vI25_19(ZJ),T) :- and3(ZH,NZP,NN,ZJ), inv(N,NN), holdsAt(zH_19(ZH),T),
holdsAt(fVCC_VCA(NZP),T), holdsAt(vI13_17(N),T), nivelLogico(ZJ),
nivelLogico(NN), timepoint(T).

% dO1D - Abrir Secionadora 89AX1
holdsAt(dO1D(ZM),T) :- and(ZJ,ZG,ZM), or(ZC,ZD,ZG), and(O,B,ZC),
and(NB,P,ZD), inv(B,NB), holdsAt(vI25_19(ZJ),T), holdsAt(vI03_80(O),T),
holdsAt(eP01_32(B),T), holdsAt(eP04_32(P),T), nivelLogico(ZG), nivelLogico(ZM),
nivelLogico(ZC), nivelLogico(ZD), nivelLogico(NB), timepoint(T).

% vI26_19 - Disjuntor 52AX Baixa pressão SF6 - 2o Estágio
holdsAt(vI26_19(Q),T) :- holdsAt(dI12N_405(Q),T), timepoint(T).

%= FIM - Figura 16 - Coqueiros Simplificado F_19  ===



% === Figura 17 - Coqueiros Simplificado F_20 ===

% === Entradas Circuito F_20 ===
% vI06_81 (Fechar Secionadora 89AX3 - Saída Circuito F_81)
% eP01_32 (Seleção Local/Remoto-Remoto - Declarada no Circuito F_19)
% vI18_17 (Secionadora 89AX3 Fechada - Saída Circuito F_17)
```

```
% vI08_81 (Colocar Cartão Vermelho - Não Opere Secionadora 89AX3 - Saída
Circuito F_81)
% vI09_81 (Retirar Cartão Vermelho - Não Opere Secionadora 89Ax3 - Saída Circuito
F_81)
% eP03_32 (Retirar Cartões - Declarada no Circuito F19)
% vI14_17 (Secionadora 89AX1 Fechada - Saída Circuito F_17)
% vI09_16 (Secionadora 89AX4 Aberta - Saída Circuito F_16)
% vI21_18 (Disjuntor 52AX Aberto - Saída Circuito F_18)
% vI13_17 (Secionadora 89AX1 Aberta - Saída Circuito F_17)
% vI41_25 (Desbloqueio Por Falha Disjuntor 52AX - Saída Circuito F_25)
% vI17_17 (Secionadora 89AX3 Aberta - Saída Circuito F_17)
% vI07_81 (Abrir Secionadora 89AX3 - Saída Circuito F_81)
% vI26_19 (Disjuntor 52AX - Baixa Pressão SF6 2o Estágio - Saída Circuito F_19)
% fVCC_VCA (Falta VCA/VCC Motor - Comum)

% === (IHM-UCD1) Fechar Secionadora 89AX3
initially(eP05_32(0)).

 terminates(eP05_32up,eP05_32(0),T) :- timepoint(T).
   initiates(eP05_32up,eP05_32(1),T) :- timepoint(T).
 terminates(eP05_32dw,eP05_32(1),T) :- timepoint(T).
   initiates(eP05_32dw,eP05_32(0),T) :- timepoint(T).

happens(eP05_32dw,1).


% === (IHM-UCD1) Secionadora 89AX3 Chave 43 L/R-Remoto
initially(dI2N(1)).

 terminates(dI2Nup,dI2N(0),T) :- timepoint(T).
   initiates(dI2Nup,dI2N(1),T) :- timepoint(T).
 terminates(dI2Ndw,dI2N(1),T) :- timepoint(T).
   initiates(dI2Ndw,dI2N(0),T) :- timepoint(T).

happens(dI2Nup,1).


% === (IHM-UCD1) Abrir Secionadora 89AX3
initially(eP06_32(0)).

 terminates(eP06_32up,eP06_32(0),T) :- timepoint(T).
   initiates(eP06_32up,eP06_32(1),T) :- timepoint(T).
 terminates(eP06_32dw,eP06_32(1),T) :- timepoint(T).
   initiates(eP06_32dw,eP06_32(0),T) :- timepoint(T).

happens(eP06_32dw,1).


%============ Flip-flop SR Circuito F20_1 ============
```

```
initially(qout_CF20_1(0)).
happens(i2_CF20_1_sup,T1) :- holdsAt(sin_CF20_1(0),T1),
holdsAt(sin_CF20_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF20_1_sdw,T1) :- holdsAt(sin_CF20_1(1),T1),
holdsAt(sin_CF20_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF20_1_rup,T1) :- holdsAt(rin_CF20_1(0),T1),
holdsAt(rin_CF20_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF20_1_rdw,T1) :- holdsAt(rin_CF20_1(1),T1),
holdsAt(rin_CF20_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF20_1_sup,qout_CF20_1(0),T) :- holdsAt(rin_CF20_1(0),T),
holdsAt(qout_CF20_1(0),T), timepoint(T).
 initiates(i2_CF20_1_sup,qout_CF20_1(1),T) :- holdsAt(rin_CF20_1(0),T),
holdsAt(qout_CF20_1(0),T), timepoint(T).
terminates(i2_CF20_1_rup,qout_CF20_1(1),T) :- holdsAt(sin_CF20_1(0),T),
holdsAt(qout_CF20_1(1),T), timepoint(T).
 initiates(i2_CF20_1_rup,qout_CF20_1(0),T) :- holdsAt(sin_CF20_1(0),T),
holdsAt(qout_CF20_1(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF20_1_rup,qout_CF20_1(1),T) :- holdsAt(sin_CF20_1(1),T),
holdsAt(qout_CF20_1(1),T), timepoint(T).
 initiates(i2_CF20_1_rup,qout_CF20_1(0),T) :- holdsAt(sin_CF20_1(1),T),
holdsAt(qout_CF20_1(1),T), timepoint(T).

%============ Pick-up timer Circuito F20_1 ============
initially(monoOutput_PU_CF20_1(0)).
happens(upTriger_PU_CF20_1,T3) :- holdsAt(inputIs_PU_CF20_1(0),T1),
holdsAt(inputIs_PU_CF20_1(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF20_1,monoOutput_PU_CF20_1(0),T) :-
happens(upTriger_PU_CF20_1,T), holdsAt(inputIs_PU_CF20_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF20_1,monoOutput_PU_CF20_1(1),T)  :-
happens(upTriger_PU_CF20_1,T), holdsAt(inputIs_PU_CF20_1(1),T), timepoint(T).

happens(downTriger_PU_CF20_1,0) :- initially(inputIs_PU_CF20_1(0)).
happens(downTriger_PU_CF20_1,T1) :- holdsAt(inputIs_PU_CF20_1(1),T1),
holdsAt(inputIs_PU_CF20_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF20_1,monoOutput_PU_CF20_1(1),T) :-
happens(downTriger_PU_CF20_1,T), holdsAt(monoOutput_PU_CF20_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF20_1,monoOutput_PU_CF20_1(0),T)  :-
happens(downTriger_PU_CF20_1,T), holdsAt(monoOutput_PU_CF20_1(1),T),
timepoint(T).


% === SAÍDAS Circuito F_20 ===
% Partes comuns a algumas saídas
holdsAt(sin_CF20_1(U),T) :- and(B,E,U), holdsAt(eP01_32(B),T),
holdsAt(vI08_81(E),T), nivelLogico(U), timepoint(T).
```

```
holdsAt(rin_CF20_1(X),T) :- or(V,W,X), and(B,F,V), and(NB,G,W), inv(B,NB),
holdsAt(eP01_32(B),T), holdsAt(vI09_81(F),T), holdsAt(eP03_32(G),T),
nivelLogico(V), nivelLogico(W), nivelLogico(X), nivelLogico(NB), timepoint(T).
holdsAt(inputIs_PU_CF20_1(NZF),T) :- inv(ZF,NZF), and(N,Q,ZF),
holdsAt(vI17_17(N),T), holdsAt(vI26_19(Q),T), nivelLogico(ZF), nivelLogico(NZF),
timepoint(T).
holdsAt(zN_20(ZN),T) :- holdsAt(monoOutput_PU_CF20_1(ZN),T), timepoint(T).
holdsAt(zE_20(ZE),T) :- or4(ZA,ZB,ZO,M,ZE), and(I,J,ZA), and(K,L,ZB),
and(K,Q,ZO), holdsAt(vI14_17(I),T), holdsAt(vI09_16(J),T), holdsAt(vI21_18(K),T),
holdsAt(vI13_17(L),T), holdsAt(vI26_19(Q),T), holdsAt(vI41_25(M),T),
nivelLogico(ZA), nivelLogico(ZB), nivelLogico(ZO), nivelLogico(ZE), timepoint(T).
holdsAt(zH_20(ZH),T) :- and4(NZ,H,ZE,ZN,ZH), inv(Z,NZ),
holdsAt(qout_CF20_1(Z),T), holdsAt(dI2N(H),T), holdsAt(zE_20(ZE),T),
holdsAt(zN_20(ZN),T), nivelLogico(ZH), nivelLogico(NZ), timepoint(T).

% dO9D - Permissão de Manobra Secionadora 89AX3
holdsAt(dO9D(ZL),T) :- and3(ZN,NZP,ZE,ZL), holdsAt(zN_20(ZN),T),
holdsAt(fVCC_VCA(NZP),T), holdsAt(zE_20(ZE),T), nivelLogico(ZL), timepoint(T).

% vI27_20 - Permissão Fechar Secionadora 89AX3
holdsAt(vI27_20(ZI),T) :- and3(ND,NZP,ZH,ZI), inv(D,ND), inv(ZP,NZP),
holdsAt(vI18_17(D),T), holdsAt(fVCC_VCA(NZP),T), holdsAt(zH_20(ZH),T),
nivelLogico(ZI), nivelLogico(ND), timepoint(T).

% dO8D - Fechar Secionadora 89AX3
holdsAt(do8D(ZK),T) :- and(Y,ZI,ZK), or(R,S,Y), and(A,B,R), and(NB,C,S),
inv(B,NB), holdsAt(vI06_81(A),T), holdsAt(eP01_32(B),T), holdsAt(eP05_32(C),T),
holdsAt(vI27_20(ZI),T), nivelLogico(ZK), nivelLogico(Y), nivelLogico(R),
nivelLogico(S), nivelLogico(NB), timepoint(T).

% vI28_20 - Permissão para abrir Secionadora 89AX3 (IHM-UCC)
holdsAt(vI28_20(ZJ),T) :- and3(ZH,NZP,NN,ZJ), inv(N,NN), holdsAt(zH_20(ZH),T),
holdsAt(fVCC_VCA(NZP),T), holdsAt(vI17_17(N),T), nivelLogico(ZJ),
nivelLogico(NN), timepoint(T).

% dO7D - Abrir Secionadora 89AX3
holdsAt(dO7D(ZM),T) :- and(ZJ,ZG,ZM), or(ZC,ZD,ZG), and(O,B,ZC),
and(NB,P,ZD), inv(B,NB), holdsAt(vI28_20(ZJ),T), holdsAt(vI07_81(O),T),
holdsAt(eP01_32(B),T), holdsAt(eP06_32(P),T), nivelLogico(ZG), nivelLogico(ZM),
nivelLogico(ZC), nivelLogico(ZD), nivelLogico(NB), timepoint(T).

%= FIM - Figura 17 - Coqueiros Simplificado F_20  ===



% === Figura 18 - Coqueiros Simplificado F_21 ===

% === Entradas Circuito F_21 ===
% vI10_82 (Fechar Secionadora 89AX2 - Saída Circuito F_82)
```

```
% eP01_32 (Seleção Local/Remoto-Remoto - Declarada no Circuito F_19)
% vI06_15 (Secionadora 89AX2 Fechada - Saída Circuito F_15)
% vI12_82 (Colocar Cartão Vermelho - Não Opere Secionadora 89AX2 - Saída
Circuito F_82)
% vI13_82 (Retirar Cartão Vermelho - Não Opere Secionadora 89Ax2 - Saída Circuito
F_82)
% eP03_32 (Retirar Cartões - Declarada no Circuito F19)
% vI21_18 (Disjuntor 52AX Aberto - Saída Circuito F_18)
% vI41_25 (Desbloqueio Por Falha Disjuntor 52AX - Saída Circuito F_25)
% vI01_15 (Secionadora 57AX Aberta - Saída Circuito F_15)
% vI26_19 (Disjuntor 52AX - Baixa Pressão SF6 2o Estágio - Saída Circuito F_19)
% vI05_15 (Secionadora 89AX2 Aberta - Saída Circuito F_15)
% vI11_82 (Abrir Secionadora 89AX2 - Saída Circuito F_82)
% fVCC_VCA (Falta VCA/VCC Motor - Comum)

% === (IHM-UCD1) Fechar Secionadora 89AX2
initially(eP07_32(0)).

 terminates(eP07_32up,eP07_32(0),T) :- timepoint(T).
   initiates(eP07_32up,eP07_32(1),T) :- timepoint(T).
 terminates(eP07_32dw,eP07_32(1),T) :- timepoint(T).
   initiates(eP07_32dw,eP07_32(0),T) :- timepoint(T).

happens(eP07_32dw,1).


% === (IHM-UCD1) Secionadora 89AX2 Chave 43 L/R-Remoto
initially(dI2O(0)).

 terminates(dI2Oup,dI2O(0),T) :- timepoint(T).
   initiates(dI2Oup,dI2O(1),T) :- timepoint(T).
 terminates(dI2Odw,dI2O(1),T) :- timepoint(T).
   initiates(dI2Odw,dI2O(0),T) :- timepoint(T).

happens(dI2Odw,1).


% === (IHM-UCD1) Abrir Secionadora 89AX2
initially(eP08_32(0)).

 terminates(eP08_32up,eP08_32(0),T) :- timepoint(T).
   initiates(eP08_32up,eP08_32(1),T) :- timepoint(T).
 terminates(eP08_32dw,eP08_32(1),T) :- timepoint(T).
   initiates(eP08_32dw,eP08_32(0),T) :- timepoint(T).

happens(eP08_32dw,1).


%============ Flip-flop SR Circuito F21_1 ============
```

```
initially(qout_CF21_1(0)).
happens(i2_CF21_1_sup,T1) :- holdsAt(sin_CF21_1(0),T1),
holdsAt(sin_CF21_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF21_1_sdw,T1) :- holdsAt(sin_CF21_1(1),T1),
holdsAt(sin_CF21_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF21_1_rup,T1) :- holdsAt(rin_CF21_1(0),T1),
holdsAt(rin_CF21_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF21_1_rdw,T1) :- holdsAt(rin_CF21_1(1),T1),
holdsAt(rin_CF21_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF21_1_sup,qout_CF21_1(0),T) :- holdsAt(rin_CF21_1(0),T),
holdsAt(qout_CF21_1(0),T), timepoint(T).
 initiates(i2_CF21_1_sup,qout_CF21_1(1),T) :- holdsAt(rin_CF21_1(0),T),
holdsAt(qout_CF21_1(0),T), timepoint(T).
terminates(i2_CF21_1_rup,qout_CF21_1(1),T) :- holdsAt(sin_CF21_1(0),T),
holdsAt(qout_CF21_1(1),T), timepoint(T).
 initiates(i2_CF21_1_rup,qout_CF21_1(0),T) :- holdsAt(sin_CF21_1(0),T),
holdsAt(qout_CF21_1(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF21_1_rup,qout_CF21_1(1),T) :- holdsAt(sin_CF21_1(1),T),
holdsAt(qout_CF21_1(1),T), timepoint(T).
 initiates(i2_CF21_1_rup,qout_CF21_1(0),T) :- holdsAt(sin_CF21_1(1),T),
holdsAt(qout_CF21_1(1),T), timepoint(T).

%============ Pick-up timer Circuito F21_1 ============
initially(monoOutput_PU_CF21_1(0)).
happens(upTriger_PU_CF21_1,T3) :- holdsAt(inputIs_PU_CF21_1(0),T1),
holdsAt(inputIs_PU_CF21_1(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF21_1,monoOutput_PU_CF21_1(0),T) :-
happens(upTriger_PU_CF21_1,T), holdsAt(inputIs_PU_CF21_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF21_1,monoOutput_PU_CF21_1(1),T)  :-
happens(upTriger_PU_CF21_1,T), holdsAt(inputIs_PU_CF21_1(1),T), timepoint(T).

happens(downTriger_PU_CF21_1,0) :- initially(inputIs_PU_CF21_1(0)).
happens(downTriger_PU_CF21_1,T1) :- holdsAt(inputIs_PU_CF21_1(1),T1),
holdsAt(inputIs_PU_CF21_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF21_1,monoOutput_PU_CF21_1(1),T) :-
happens(downTriger_PU_CF21_1,T), holdsAt(monoOutput_PU_CF21_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF21_1,monoOutput_PU_CF21_1(0),T)  :-
happens(downTriger_PU_CF21_1,T), holdsAt(monoOutput_PU_CF21_1(1),T),
timepoint(T).


% === SAÍDAS Circuito F_21 ===
% Partes comuns a algumas saídas
holdsAt(sin_CF21_1(U),T) :- and(B,E,U), holdsAt(eP01_32(B),T),
holdsAt(vI12_82(E),T), nivelLogico(U), timepoint(T).
```

```
holdsAt(rin_CF21_1(X),T) :- or(V,W,X), and(B,F,V), and(NB,G,W), inv(B,NB),
holdsAt(eP01_32(B),T), holdsAt(vI13_82(F),T), holdsAt(eP03_32(G),T),
nivelLogico(V), nivelLogico(W), nivelLogico(X), nivelLogico(NB), timepoint(T).
holdsAt(inputIs_PU_CF21_1(NZF),T) :- inv(ZF,NZF), and(N,Q,ZF),
holdsAt(vI05_15(N),T), holdsAt(vI26_19(Q),T), nivelLogico(ZF), nivelLogico(NZF),
timepoint(T).
holdsAt(zN_21(ZN),T) :- holdsAt(monoOutput_PU_CF21_1(ZN),T), timepoint(T).
holdsAt(zE_21(ZE),T) :- or(K,M,ZE), holdsAt(vI21_18(K),T), holdsAt(vI41_25(M),T),
nivelLogico(ZE), timepoint(T).
holdsAt(zH_21(ZH),T) :- and4(NZ,H,ZE,ZN,ZH), inv(Z,NZ),
holdsAt(qout_CF21_1(Z),T), holdsAt(dI2O(H),T), holdsAt(zE_21(ZE),T),
holdsAt(zN_21(ZN),T), nivelLogico(ZH), nivelLogico(NZ), timepoint(T).

% dO6D - Permissão de Manobra Secionadora 89AX2
holdsAt(dO6D(ZL),T) :- and4(ZE,ZN,NZP,L,ZL), holdsAt(zE_21(ZE),T),
holdsAt(zN_21(ZN),T), holdsAt(fVCC_VCA(NZP),T), holdsAt(vI01_15(L),T),
nivelLogico(ZL), timepoint(T).

% vI29_21 - Permissão Fechar Secionadora 89AX2
holdsAt(vI29_21(ZI),T) :- and3(ND,NZP,ZH,ZI), inv(D,ND), inv(ZP,NZP),
holdsAt(vI06_15(D),T), holdsAt(fVCC_VCA(NZP),T), holdsAt(zH_21(ZH),T),
nivelLogico(ZI), nivelLogico(ND), timepoint(T).

% dO5D - Fechar Secionadora 89AX2
holdsAt(do5D(ZK),T) :- and(Y,ZI,ZK), or(R,S,Y), and(A,B,R), and(NB,C,S),
inv(B,NB), holdsAt(vI10_82(A),T), holdsAt(eP01_32(B),T), holdsAt(eP07_32(C),T),
holdsAt(vI29_21(ZI),T), nivelLogico(ZK), nivelLogico(Y), nivelLogico(R),
nivelLogico(S), nivelLogico(NB), timepoint(T).

% vI30_21 - Permissão para abrir Secionadora 89AX2 (IHM-UCC)
holdsAt(vI30_21(ZJ),T) :- and3(ZH,NZP,NN,ZJ), inv(N,NN), holdsAt(zH_21(ZH),T),
holdsAt(fVCC_VCA(NZP),T), holdsAt(vI05_15(N),T), nivelLogico(ZJ),
nivelLogico(NN), timepoint(T).

% dO4D - Abrir Secionadora 89AX2
holdsAt(dO4D(ZM),T) :- and(ZJ,ZG,ZM), or(ZC,ZD,ZG), and(O,B,ZC),
and(NB,P,ZD), inv(B,NB), holdsAt(vI30_21(ZJ),T), holdsAt(vI11_82(O),T),
holdsAt(eP01_32(B),T), holdsAt(eP08_32(P),T), nivelLogico(ZG), nivelLogico(ZM),
nivelLogico(ZC), nivelLogico(ZD), nivelLogico(NB), timepoint(T).

%= FIM - Figura 18 - Coqueiros Simplificado F_21  ===



% === Figura 19 - Coqueiros Simplificado F_22 ===

% === Entradas Circuito F_22 ===
% vI58_28 (TPAX (Enrolamento 1X) - ABerto) - Saída Circuito F_28)
% vI05_15 (Secionadora 89AX2 Aberta - Saída Circuito F_15)
```

```
% vI09_16 (Secionadora 89AX4 Aberta - Saída Circuito F_16)
% dI11O (Falta VCA Motor - Entrada Circuito-Comum)
% dI5M (Falta VCC Circuito Comando - Entrada Circuito-Comum)
% dI9M (Falta VCC Cont. Secionadora - Entrada Circuito-Comum)

% === (IHM-UCD1) Bloco de Teste em Serviço
initially(dI9M(0)).

 terminates(dI9Mup,dI9M(0),T) :- timepoint(T).
   initiates(dI9Mup,dI9M(1),T) :- timepoint(T).
 terminates(dI9Mdw,dI9M(1),T) :- timepoint(T).
   initiates(dI9Mdw,dI9M(0),T) :- timepoint(T).

happens(dI9Mdw,1).


% === Tensão na LT Fases A, B, V
initially(vI70_34(0)).

 terminates(vI70_34up,vI70_34(0),T) :- timepoint(T).
   initiates(vI70_34up,vI70_34(1),T) :- timepoint(T).
 terminates(vI70_34dw,vI70_34(1),T) :- timepoint(T).
   initiates(vI70_34dw,vI70_34(0),T) :- timepoint(T).

happens(vI70_34dw,1).


% === SAÍDAS Circuito F_22 ===

% dO3F - Permissão de Manobra Secionadora 57AX
holdsAt(dO3F(J),T) :- and7(NA,B,C,I,NF,NG,H,J), and(D,NE,I), inv(A,NA),
inv(E,NE), inv(F,NF), inv(G,NG), holdsAt(vI58_28(A),T), holdsAt(vI05_15(B),T),
holdsAt(vI09_16(C),T), holdsAt(dI9M(D),T), holdsAt(vI70_34(E),T),
holdsAt(dI11O(F),T), holdsAt(dI5M(G),T), holdsAt(dI9M(H),T), nivelLogico(J),
nivelLogico(NA), nivelLogico(NE), nivelLogico(NF), nivelLogico(NG),
nivelLogico(I), timepoint(T).

%= FIM - Figura 19 - Coqueiros Simplificado F_22  ===



% === Figura 20 - Coqueiros Simplificado F_23 ===

% === Entradas Circuito F_23 ===
% vI18_84 (Fechar Secionadora 89AX4 - Saída Circuito F_84)
% eP01_32 (Seleção Local/Remoto-Remoto - Declarada no Circuito F_19)
% vI44_26 (Tranferência de Proteção em Manual - Saída Circuito F_26)
% vI51_27 (Proteção em Transferência - Saída Circuito F_27)
% vI45_27 (Transferência de Proteção em Automático - Saída Circuito F_27)
```

```
% vI10_16 (Secionadora 89AX4 Fechada - Saída Circuito F_16)
% vI20_84 (Colocar Cartão Vermelho - Não Opere Secionadora 89AX4 - Saída
Circuito F_84)
% vI21_84 (Retirar Cartão Vermelho - Não Opere Secionadora 89Ax4 - Saída Circuito
F_84)
% eP03_32 (Retirar Cartões - Declarada no Circuito F19)
% vI17_17 (Secionadora 89AX3 Aberta - Saída Circuito F_17)
% vI14_17 (Secionadora 89AX1 Fechada - Saída Circuito F_17)
% vI13_17 (Secionadora 89AX1 Aberta - Saída Circuito F_17)
% vI18_17 (Secionadora 89AX3 Fechada - Saída Circuito F_17)
% vI06_15 (Secionadora 89AX2 Fechada - Saída Circuito F_15)
% vI22_18 (Disjuntor 52AX Fechado - Saída Circuito F_18)
% vI01_15 (Secionadora 57AX Aberta - Saída Circuito F_15)
% vI09_16 (Secionadora 89AX4 Aberta - Saída Circuito F_16)
% vI19_84 (Abrir Secionadora 89AX4 - Saída Circuito F_84)
% fVCC_VCA (Falta VCA/VCC Motor - Comum)

% === (IHM-UCD1) Fechar Secionadora 89AX4
initially(eP11_32(0)).

 terminates(eP11_32up,eP11_32(0),T) :- timepoint(T).
   initiates(eP11_32up,eP11_32(1),T) :- timepoint(T).
 terminates(eP11_32dw,eP11_32(1),T) :- timepoint(T).
   initiates(eP11_32dw,eP11_32(0),T) :- timepoint(T).

happens(eP11_32dw,1).


% === (IHM-UCD1) Secionadora 89AX4 Chave 43 L/R-Remoto
initially(dI10O(1)).

 terminates(dI10Oup,dI10O(0),T) :- timepoint(T).
   initiates(dI10Oup,dI10O(1),T) :- timepoint(T).
 terminates(dI10Odw,dI10O(1),T) :- timepoint(T).
   initiates(dI10Odw,dI10O(0),T) :- timepoint(T).

happens(dI10Oup,1).


% === (IHM-UCD1) Abrir Secionadora 89AX4
initially(eP12_32(0)).

 terminates(eP12_32up,eP12_32(0),T) :- timepoint(T).
   initiates(eP12_32up,eP12_32(1),T) :- timepoint(T).
 terminates(eP12_32dw,eP12_32(1),T) :- timepoint(T).
   initiates(eP12_32dw,eP12_32(0),T) :- timepoint(T).

happens(eP12_32dw,1).
```

```
%============ Flip-flop SR Circuito F23_1 ============
initially(qout_CF23_1(0)).
happens(i2_CF23_1_sup,T1) :- holdsAt(sin_CF23_1(0),T1),
holdsAt(sin_CF23_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF23_1_sdw,T1) :- holdsAt(sin_CF23_1(1),T1),
holdsAt(sin_CF23_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF23_1_rup,T1) :- holdsAt(rin_CF23_1(0),T1),
holdsAt(rin_CF23_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF23_1_rdw,T1) :- holdsAt(rin_CF23_1(1),T1),
holdsAt(rin_CF23_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF23_1_sup,qout_CF23_1(0),T) :- holdsAt(rin_CF23_1(0),T),
holdsAt(qout_CF23_1(0),T), timepoint(T).
 initiates(i2_CF23_1_sup,qout_CF23_1(1),T) :- holdsAt(rin_CF23_1(0),T),
holdsAt(qout_CF23_1(0),T), timepoint(T).
terminates(i2_CF23_1_rup,qout_CF23_1(1),T) :- holdsAt(sin_CF23_1(0),T),
holdsAt(qout_CF23_1(1),T), timepoint(T).
 initiates(i2_CF23_1_rup,qout_CF23_1(0),T) :- holdsAt(sin_CF23_1(0),T),
holdsAt(qout_CF23_1(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF23_1_rup,qout_CF23_1(1),T) :- holdsAt(sin_CF23_1(1),T),
holdsAt(qout_CF23_1(1),T), timepoint(T).
 initiates(i2_CF23_1_rup,qout_CF23_1(0),T) :- holdsAt(sin_CF23_1(1),T),
holdsAt(qout_CF23_1(1),T), timepoint(T).

%============ Drop-out timer Circuito F23_1 ====================
initially(monoOutput_DO_CF23_1(0)).
happens(upTriger_DO_CF23_1,0) :- initially(inputIs_DO_CF23_1(1)).
happens(upTriger_DO_CF23_1,T2) :- holdsAt(inputIs_DO_CF23_1(0),T1),
holdsAt(inputIs_DO_CF23_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(upTriger_DO_CF23_1,monoOutput_DO_CF23_1(0),T) :-
happens(upTriger_DO_CF23_1,T), holdsAt(inputIs_DO_CF23_1(1),T), timepoint(T).
initiates(upTriger_DO_CF23_1,monoOutput_DO_CF23_1(1),T)  :-
happens(upTriger_DO_CF23_1,T), holdsAt(inputIs_DO_CF23_1(1),T), timepoint(T).

happens(downTriger_DO_CF23_1,T3) :- holdsAt(inputIs_DO_CF23_1(1),T1),
holdsAt(inputIs_DO_CF23_1(0),T2), T2=T1+1, T3=T1+3, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(downTriger_DO_CF23_1,monoOutput_DO_CF23_1(1),T) :-
happens(downTriger_DO_CF23_1,T), holdsAt(monoOutput_DO_CF23_1(1),T),
timepoint(T).
initiates(downTriger_DO_CF23_1,monoOutput_DO_CF23_1(0),T)  :-
happens(downTriger_DO_CF23_1,T), holdsAt(monoOutput_DO_CF23_1(1),T),
timepoint(T).

%============ Drop-out timer Circuito F23_2 ====================
initially(monoOutput_DO_CF23_2(0)).
happens(upTriger_DO_CF23_2,0) :- initially(inputIs_DO_CF23_2(1)).
```

```
happens(upTriger_DO_CF23_2,T2) :- holdsAt(inputIs_DO_CF23_2(0),T1),
holdsAt(inputIs_DO_CF23_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(upTriger_DO_CF23_2,monoOutput_DO_CF23_2(0),T) :-
happens(upTriger_DO_CF23_2,T), holdsAt(inputIs_DO_CF23_2(1),T), timepoint(T).
initiates(upTriger_DO_CF23_2,monoOutput_DO_CF23_2(1),T)  :-
happens(upTriger_DO_CF23_2,T), holdsAt(inputIs_DO_CF23_2(1),T), timepoint(T).

happens(downTriger_DO_CF23_2,T3) :- holdsAt(inputIs_DO_CF23_2(1),T1),
holdsAt(inputIs_DO_CF23_2(0),T2), T2=T1+1, T3=T1+3, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(downTriger_DO_CF23_2,monoOutput_DO_CF23_2(1),T) :-
happens(downTriger_DO_CF23_2,T), holdsAt(monoOutput_DO_CF23_2(1),T),
timepoint(T).
initiates(downTriger_DO_CF23_2,monoOutput_DO_CF23_2(0),T)  :-
happens(downTriger_DO_CF23_2,T), holdsAt(monoOutput_DO_CF23_2(1),T),
timepoint(T).


% === SAÍDAS Circuito F_23 ===
% Partes comuns a algumas saídas
holdsAt(inputIs_DO_CF23_1(Y),T) :- or(R,S,Y), and(A,B,R), and(NB,C,S),
inv(B,NB), holdsAt(vI06_81(A),T), holdsAt(eP01_32(B),T), holdsAt(eP05_32(C),T),
nivelLogico(Y), nivelLogico(R), nivelLogico(S), nivelLogico(NB), timepoint(T).
holdsAt(sin_CF23_1(U),T) :- and(B,E,U), holdsAt(eP01_32(B),T),
holdsAt(vI20_84(E),T), nivelLogico(U), timepoint(T).
holdsAt(rin_CF23_1(X),T) :- or(V,W,X), and(B,F,V), and(NB,G,W), inv(B,NB),
holdsAt(eP01_32(B),T), holdsAt(vI21_84(F),T), holdsAt(eP03_32(G),T),
nivelLogico(V), nivelLogico(W), nivelLogico(X), nivelLogico(NB), timepoint(T).
holdsAt(inputIs_DO_CF23_2(ZG),T) :- or(ZC,ZD,ZG), and(O,B,ZC), and(NB,P,ZD),
inv(B,NB), holdsAt(vI19_16(O),T), holdsAt(eP01_32(B),T), holdsAt(eP12_32(P),T),
nivelLogico(ZG), nivelLogico(ZC), nivelLogico(ZD), nivelLogico(NB), timepoint(T).
holdsAt(dE_23(DE),T) :- or(DC,DD,DE), and(DA,DB,DD), holdsAt(vI44_26(DA),T),
holdsAt(vI51_27(DB),T), holdsAt(vI45_27(DC),T), nivelLogico(DE),
nivelLogico(DD), timepoint(T).
holdsAt(zE_23(ZE),T) :- and(ME,MF,ZE), and3(MD,M,MA,MF), or(MB,MC,MD),
and(I,J,MB), and(K,L,MC), holdsAt(vI17_17(I),T), holdsAt(vI14_17(J),T),
holdsAt(vI13_17(K),T), holdsAt(vI18_17(L),T), holdsAt(vI06_15(M),T),
holdsAt(vI22_18(MA),T), holdsAt(vI01_15(ME),T), nivelLogico(ZE),
nivelLogico(MF), nivelLogico(MD), nivelLogico(MC), nivelLogico(MB),
timepoint(T).
holdsAt(zH_23(ZH),T) :- and3(NZ,H,ZE,ZH), inv(Z,NZ), holdsAt(qout_CF23_1(Z),T),
holdsAt(dI10O(H),T), holdsAt(zE_23(ZE),T), nivelLogico(ZH), nivelLogico(NZ),
timepoint(T).

% dO2E - Permissão de Manobra Secionadora 89AX4
holdsAt(dO2E(ZL),T) :- and3(DE,NZP,ZE,ZL), holdsAt(dE_23(DE),T),
holdsAt(fVCC_VCA(NZP),T), holdsAt(zE_23(ZE),T), nivelLogico(ZL), timepoint(T).

% vI33_22 - Permissão Fechar Secionadora 89AX4
```

```
holdsAt(vI33_23(ZI),T) :- and4(DE,ND,NZP,ZH,ZI), inv(D,ND),
holdsAt(dE_23(DE),T), holdsAt(vI10_16(D),T), holdsAt(fVCC_VCA(NZP),T),
holdsAt(zH_23(ZH),T), nivelLogico(ZI), nivelLogico(ND), timepoint(T).

% dO1E - Fechar Secionadora 89AX4
holdsAt(dO1E(ZK),T) :- and(YP,ZI,ZK), holdsAt(monoOutput_DO_CF23_1(YP),T),
holdsAt(vI33_23(ZI),T), nivelLogico(ZK), timepoint(T).
holdsAt(vI35_23(ZK),T) :- holdsAt(dO1E(ZK),T), timepoint(T).

% vI34_33 - Permissão para abrir Secionadora 89AX4 (IHM-UCC)
holdsAt(vI34_33(ZJ),T) :- and3(ZH,NZP,NN,ZJ), inv(N,NN), holdsAt(zH_23(ZH),T),
holdsAt(fVCC_VCA(NZP),T), holdsAt(vI09_16(N),T), nivelLogico(ZJ),
nivelLogico(NN), timepoint(T).

% dO10D - Abrir Secionadora 89AX4
holdsAt(dO10D(ZM),T) :- and(ZJ,ZGP,ZM), holdsAt(vI34_33(ZJ),T),
nivelLogico(ZM), holdsAt(monoOutput_DO_CF23_2(ZGP),T), timepoint(T).

%= FIM - Figura 20 - Coqueiros Simplificado F_23  ===



% === Figura 21 - Coqueiros Simplificado F_24 ===

% === Entradas Circuito F_24 ===
% vI22_85   (Fechar Disjuntor 52AX - Saída Circuito F_85)
% eP01_32   (Seleção Local/Remoto-Remoto - Declarada no Circuito F_19)
% vI22_18   (Intertravamento: Disjuntor 52AX Fechado - Saída Circuito F_18)
% vI59_28   (Permissão Fechamento Disjuntor 52AX - Saída Circuito F_28)
% vI60_28   (Sincronismo Verificado - Disjuntor Isolado - Soleção de Potência Normal
- Saída Circuito F_28)
% vI50_27   (Proteção da Seção AX - Normal(N) - Saída Circuito F_27)
% vI51_27   (Proteção da Seção AX - Em Transferência (ET) - Saída Circuito F_27)
% vI43_25   (Bloqueio Fechamento Disjuntor 52AX - Saída Circuito F_25)
% vI24_85   (Colocar Cartão Amarelo - Linha Viva Disjuntor 52AX - Saída Circuito
F_85)
% vI25_85   (Retirar Cartão Amarelo - Linha Viva Disjuntor 52AX - Saída Circuito
F_85)
% eP03_32   (Retirar Cartões - Declarada no Circuito F19)
% vI26_85   (Colocar Cartão Vermelho - Não Opere Disjuntor 52AX - Saída Circuito
F_85)
% vI27_85   (Retirar Cartão Vermelho - Não Opere Disjuntor 52AX - Saída Circuito
F_85)
% dI12N_406 (Baixa Pressão SF6 2o Estágio - Declarado no Circuito F_19)
% vI21_18   (Disjuntor 52AX Aberto - Saída Circuito F_18)
% vI23_85   (Abrir Disjuntor 52AX - Saída Circuito F_85)

% === (IHM-UCD1) Fechar Disjuntor 52AX
initially(eP13_32(0)).
```

```
 terminates(eP13_32up,eP13_32(0),T) :- timepoint(T).
   initiates(eP13_32up,eP13_32(1),T) :- timepoint(T).
 terminates(eP13_32dw,eP13_32(1),T) :- timepoint(T).
   initiates(eP13_32dw,eP13_32(0),T) :- timepoint(T).

happens(eP13_32dw,1).


% === Mola Descarregada
initially(vI14_405(0)).

 terminates(vI14_405up,vI14_405(0),T) :- timepoint(T).
   initiates(vI14_405up,vI14_405(1),T) :- timepoint(T).
 terminates(vI14_405dw,vI14_405(1),T) :- timepoint(T).
   initiates(vI14_405dw,vI14_405(0),T) :- timepoint(T).

happens(vI14_405dw,1).
happens(vI14_405up,18).


% === Falta VCC Circuito Fechamento
initially(dI7M_406(0)).

 terminates(dI7M_406up,dI7M_406(0),T) :- timepoint(T).
   initiates(dI7M_406up,dI7M_406(1),T) :- timepoint(T).
 terminates(dI7M_406dw,dI7M_406(1),T) :- timepoint(T).
   initiates(dI7M_406dw,dI7M_406(0),T) :- timepoint(T).

happens(dI7M_406dw,1).
happens(dI7M_406up,18).


% === Discordância de Polos
initially(dI13N_405(0)).

 terminates(dI13N_405up,dI13N_405(0),T) :- timepoint(T).
   initiates(dI13N_405up,dI13N_405(1),T) :- timepoint(T).
 terminates(dI13N_405dw,dI13N_405(1),T) :- timepoint(T).
   initiates(dI13N_405dw,dI13N_405(0),T) :- timepoint(T).

happens(dI13N_405dw,1).


% === Falta VCC Circuito Comando
initially(dI14N_406(0)).

 terminates(dI14N_406up,dI14N_406(0),T) :- timepoint(T).
   initiates(dI14N_406up,dI14N_406(1),T) :- timepoint(T).
```

```
 terminates(dI14N_406dw,dI14N_406(1),T) :- timepoint(T).
   initiates(dI14N_406dw,dI14N_406(0),T) :- timepoint(T).

happens(dI14N_406dw,1).


% === Falha Abertura 1
initially(dI17N_406(0)).

 terminates(dI17N_406up,dI17N_406(0),T) :- timepoint(T).
   initiates(dI17N_406up,dI17N_406(1),T) :- timepoint(T).
 terminates(dI17N_406dw,dI17N_406(1),T) :- timepoint(T).
   initiates(dI17N_406dw,dI17N_406(0),T) :- timepoint(T).

happens(dI17N_406dw,1).


% === Falha Abertura 2
initially(dI18N_406(0)).

 terminates(dI18N_406up,dI18N_406(0),T) :- timepoint(T).
   initiates(dI18N_406up,dI18N_406(1),T) :- timepoint(T).
 terminates(dI18N_406dw,dI18N_406(1),T) :- timepoint(T).
   initiates(dI18N_406dw,dI18N_406(0),T) :- timepoint(T).

happens(dI18N_406dw,1).


% === Disjuntor 52AX Chave 43 L/R - Remoto
initially(dI15N_405(1)).

 terminates(dI15N_405up,dI15N_405(0),T) :- timepoint(T).
   initiates(dI15N_405up,dI15N_405(1),T) :- timepoint(T).
 terminates(dI15N_405dw,dI15N_405(1),T) :- timepoint(T).
   initiates(dI15N_405dw,dI15N_405(0),T) :- timepoint(T).

happens(dI15N_405up,1).


% === (IHM-UCD1) Abrir Disjuntor 52AX
initially(eP14_32(0)).

 terminates(eP14_32up,eP14_32(0),T) :- timepoint(T).
   initiates(eP14_32up,eP14_32(1),T) :- timepoint(T).
 terminates(eP14_32dw,eP14_32(1),T) :- timepoint(T).
   initiates(eP14_32dw,eP14_32(0),T) :- timepoint(T).

happens(eP14_32dw,1).
```

```
%============ Flip-flop SR Circuito F24_1 ============
initially(qout_CF24_1(0)).
happens(i2_CF24_1_sup,T1) :- holdsAt(sin_CF24_1(0),T1),
holdsAt(sin_CF24_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_1_sdw,T1) :- holdsAt(sin_CF24_1(1),T1),
holdsAt(sin_CF24_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_1_rup,T1) :- holdsAt(rin_CF24_1(0),T1),
holdsAt(rin_CF24_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_1_rdw,T1) :- holdsAt(rin_CF24_1(1),T1),
holdsAt(rin_CF24_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF24_1_sup,qout_CF24_1(0),T) :- holdsAt(rin_CF24_1(0),T),
holdsAt(qout_CF24_1(0),T), timepoint(T).
 initiates(i2_CF24_1_sup,qout_CF24_1(1),T) :- holdsAt(rin_CF24_1(0),T),
holdsAt(qout_CF24_1(0),T), timepoint(T).
terminates(i2_CF24_1_rup,qout_CF24_1(1),T) :- holdsAt(sin_CF24_1(0),T),
holdsAt(qout_CF24_1(1),T), timepoint(T).
 initiates(i2_CF24_1_rup,qout_CF24_1(0),T) :- holdsAt(sin_CF24_1(0),T),
holdsAt(qout_CF24_1(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF24_1_rup,qout_CF24_1(1),T) :- holdsAt(sin_CF24_1(1),T),
holdsAt(qout_CF24_1(1),T), timepoint(T).
 initiates(i2_CF24_1_rup,qout_CF24_1(0),T) :- holdsAt(sin_CF24_1(1),T),
holdsAt(qout_CF24_1(1),T), timepoint(T).

%============ Flip-flop SR Circuito F24_2 ============
initially(qout_CF24_2(0)).
happens(i2_CF24_2_sup,T1) :- holdsAt(sin_CF24_2(0),T1),
holdsAt(sin_CF24_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_2_sdw,T1) :- holdsAt(sin_CF24_2(1),T1),
holdsAt(sin_CF24_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_2_rup,T1) :- holdsAt(rin_CF24_2(0),T1),
holdsAt(rin_CF24_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_2_rdw,T1) :- holdsAt(rin_CF24_2(1),T1),
holdsAt(rin_CF24_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF24_2_sup,qout_CF24_2(0),T) :- holdsAt(rin_CF24_2(0),T),
holdsAt(qout_CF24_2(0),T), timepoint(T).
 initiates(i2_CF24_2_sup,qout_CF24_2(1),T) :- holdsAt(rin_CF24_2(0),T),
holdsAt(qout_CF24_2(0),T), timepoint(T).
terminates(i2_CF24_2_rup,qout_CF24_2(1),T) :- holdsAt(sin_CF24_2(0),T),
holdsAt(qout_CF24_2(1),T), timepoint(T).
 initiates(i2_CF24_2_rup,qout_CF24_2(0),T) :- holdsAt(sin_CF24_2(0),T),
holdsAt(qout_CF24_2(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF24_2_rup,qout_CF24_2(1),T) :- holdsAt(sin_CF24_2(1),T),
holdsAt(qout_CF24_2(1),T), timepoint(T).
 initiates(i2_CF24_2_rup,qout_CF24_2(0),T) :- holdsAt(sin_CF24_2(1),T),
holdsAt(qout_CF24_2(1),T), timepoint(T).
```

```
%============ Flip-flop SR Circuito F24_3 ============
initially(qout_CF24_3(0)).
happens(i2_CF24_3_sup,T1) :- holdsAt(sin_CF24_3(0),T1),
holdsAt(sin_CF24_3(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_3_sdw,T1) :- holdsAt(sin_CF24_3(1),T1),
holdsAt(sin_CF24_3(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_3_rup,T1) :- holdsAt(rin_CF24_3(0),T1),
holdsAt(rin_CF24_3(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF24_3_rdw,T1) :- holdsAt(rin_CF24_3(1),T1),
holdsAt(rin_CF24_3(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF24_3_sup,qout_CF24_3(0),T) :- holdsAt(rin_CF24_3(0),T),
holdsAt(qout_CF24_3(0),T), timepoint(T).
 initiates(i2_CF24_3_sup,qout_CF24_3(1),T) :- holdsAt(rin_CF24_3(0),T),
holdsAt(qout_CF24_3(0),T), timepoint(T).
terminates(i2_CF24_3_rup,qout_CF24_3(1),T) :- holdsAt(sin_CF24_3(0),T),
holdsAt(qout_CF24_3(1),T), timepoint(T).
 initiates(i2_CF24_3_rup,qout_CF24_3(0),T) :- holdsAt(sin_CF24_3(0),T),
holdsAt(qout_CF24_3(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF24_3_rup,qout_CF24_3(1),T) :- holdsAt(sin_CF24_3(1),T),
holdsAt(qout_CF24_3(1),T), timepoint(T).
 initiates(i2_CF24_3_rup,qout_CF24_3(0),T) :- holdsAt(sin_CF24_3(1),T),
holdsAt(qout_CF24_3(1),T), timepoint(T).

%============ Pick-up timer Circuito F24_1 ============
initially(monoOutput_PU_CF24_1(0)).
happens(upTriger_PU_CF24_1,T3) :- holdsAt(inputIs_PU_CF24_1(0),T1),
holdsAt(inputIs_PU_CF24_1(1),T2), T2=T1+1, T3=T1+60, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF24_1,monoOutput_PU_CF24_1(0),T) :-
happens(upTriger_PU_CF24_1,T), holdsAt(inputIs_PU_CF24_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF24_1,monoOutput_PU_CF24_1(1),T)  :-
happens(upTriger_PU_CF24_1,T), holdsAt(inputIs_PU_CF24_1(1),T), timepoint(T).

happens(downTriger_PU_CF24_1,0) :- initially(inputIs_PU_CF24_1(0)).
happens(downTriger_PU_CF24_1,T1) :- holdsAt(inputIs_PU_CF24_1(1),T1),
holdsAt(inputIs_PU_CF24_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF24_1,monoOutput_PU_CF24_1(1),T) :-
happens(downTriger_PU_CF24_1,T), holdsAt(monoOutput_PU_CF24_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF24_1,monoOutput_PU_CF24_1(0),T)  :-
happens(downTriger_PU_CF24_1,T), holdsAt(monoOutput_PU_CF24_1(1),T),
timepoint(T).


% === SAÍDAS Circuito F_24 ===
% Partes comuns a algumas saídas
```

holdsAt(sin_CF24_1(BA),T) :- or(AA,AB,BA), and(A,B,AA), and(NB,C,AB),
inv(B,NB), holdsAt(vI22_85(A),T), holdsAt(eP01_32(B),T), holdsAt(eP13_32(C),T),
nivelLogico(BA), nivelLogico(AA), nivelLogico(AB), nivelLogico(NB), timepoint(T).
holdsAt(inputIs_PU_CF24_1(CA),T) :- holdsAt(cA_24(CA),T), timepoint(T).
holdsAt(dF(DF),T) :- holdsAt(monoOutput_PU_CF24_1(DF),T), timepoint(T).
holdsAt(rin_CF24_1(AC),T) :- or(D,DF,AC), holdsAt(vI22_18(D),T),
holdsAt(dF(DF),T), nivelLogico(AC), timepoint(T).
holdsAt(sin_CF24_2(AE),T) :- and(B,L,AE), holdsAt(eP01_32(B),T),
holdsAt(vI24_85(L),T), nivelLogico(AE), timepoint(T).
holdsAt(aG(AG),T) :- and(NB,N,AG), holdsAt(eP01_32(B),T),
holdsAt(eP03_32(N),T), nivelLogico(NB), nivelLogico(AG), timepoint(T).
holdsAt(rin_CF24_2(BB),T) :- or(AF,AG,BB), and(B,M,AF), holdsAt(aG(AG),T),
holdsAt(eP01_32(B),T), holdsAt(vI25_85(M),T), nivelLogico(BB), nivelLogico(AF),
timepoint(T).
holdsAt(sin_CF24_3(AH),T) :- and(B,O,AH), holdsAt(eP01_32(B),T),
holdsAt(vI26_85(O),T), nivelLogico(AH), timepoint(T).
holdsAt(rin_CF24_3(BC),T) :- or(AG,AI,BC), and(B,P,AI), holdsAt(aG(AG),T),
holdsAt(eP01_32(B),T), holdsAt(vI27_85(P),T), nivelLogico(BC), nivelLogico(AI),
timepoint(T).
holdsAt(cA_24(CA),T) :- holdsAt(qout_CF24_1(CA),T), timepoint(T).
holdsAt(cD_24(CD),T) :- holdsAt(qout_CF24_2(CD),T), timepoint(T).
holdsAt(dE_24(DE),T) :- and3(NCE,NBD,W,DE), or4(Q,R,S,AJ,BD), and(T,V,AJ),
inv(CE,NCE), inv(BD,NBD), holdsAt(qout_CF24_3(CE),T),
holdsAt(dI12N_405(Q),T), holdsAt(dI13N_405(R),T), holdsAt(dI14N_406(S),T),
holdsAt(dI17N_406(T),T), holdsAt(dI18N_406(V),T), holdsAt(dI15N_405(W),T),
nivelLogico(DE), nivelLogico(BD), nivelLogico(AJ), nivelLogico(NCE),
nivelLogico(NBD), timepoint(T).
holdsAt(dC_24(DC),T) :- and6(NI,NJ,NK,NCD,NDE,X,DC), inv(I,NI), inv(J,NJ),
inv(K,NK), inv(CD,NCD), inv(DE,NDE), holdsAt(vI14_405(I),T),
holdsAt(dI7M_406(J),T), holdsAt(vI43_25(K),T), holdsAt(cD_24(CD),T),
holdsAt(dE_24(DE),T), holdsAt(vI21_18(X),T), nivelLogico(DC), nivelLogico(NI),
nivelLogico(NJ), nivelLogico(NK), nivelLogico(NCD), nivelLogico(NDE),
timepoint(T).
holdsAt(dB_24(DB),T) :- and(J,NDC,DB), inv(DC,NDC), holdsAt(dI7M_406(J),T),
holdsAt(dC_24(DC),T), nivelLogico(DB), nivelLogico(NDC), timepoint(T).

% vI36_24 - Fechamento Manual do Disjuntor Abortado
holdsAt(vI36_24(DF),T) :- holdsAt(dF(DF),T), timepoint(T).

% vI37_24 - Comando/Permissão Fechamento Disjuntor 52AX
holdsAt(vI37_24(EA),T) :- and(CA,DC,EA), holdsAt(cA_24(CA),T),
holdsAt(dC_24(DC),T), nivelLogico(EA), timepoint(T).

% dO4E_24 - Fechar Disjuntor 52AX
holdsAt(dO4E_24(EB),T) :- and5(CA,NDB,E,AD,DC,EB), or3(F,G,H,AD),
inv(DB,NDB), holdsAt(cA_24(CA),T), holdsAt(dB_24(DB),T),
holdsAt(vI59_28(E),T), holdsAt(vI60_28(F),T), holdsAt(vI50_28(G),T),
holdsAt(vI51_27(H),T), holdsAt(dC_24(DC),T), nivelLogico(EB), nivelLogico(AD),
nivelLogico(NDB), timepoint(T).

```
% vO2_24 - Intertravamento Falha VCC
holdsAt(vO2_24(DB),T) :- holdsAt(dB_24(DB),T), timepoint(T).

% vI38_24 - Permissão de Fechamento Disjuntor 52AX (IHM-UCC)
holdsAt(vI38_24(DC),T) :- holdsAt(dC_24(DC),T), timepoint(T).

% vI39_24 - Cartão Amarelo Colocado Disjuntor 52AX
holdsAt(vI39_24(CD),T) :- holdsAt(cD_24(CD),T), timepoint(T).

% vI40_24 - Permissão de Abertura Disjuntor 52AX (IHM-UCC)
holdsAt(vI40_24(EC),T) :- and(DE,NX,EC), inv(X,NX), holdsAt(dE_24(DE),T),
holdsAt(vI21_18(X),T), nivelLogico(EC), nivelLogico(NX), timepoint(T).

% dO3E_24 - Abrir Disjuntor 52AX
holdsAt(dO3E_24(ED),T) :- and3(DE,NX,BE,ED), or(AK,AL,BE), and(Y,B,AK),
and(NB,Z,AL), inv(X,NX), inv(B,NB), holdsAt(dE_24(DE),T),
holdsAt(vI21_18(X),T), holdsAt(vI23_85(Y),T), holdsAt(eP01_32(B),T),
holdsAt(eP14_32(Z),T), nivelLogico(ED), nivelLogico(NX), nivelLogico(BE),
nivelLogico(AK), nivelLogico(AL), nivelLogico(NB), timepoint(T).

% vO3_24 - Virtual output 3 - Disjuntor 52AX Chave 43 L/R - Remoto
holdsAt(vO3_24(W),T) :- holdsAt(dI15N_405(W),T), timepoint(T).

%= FIM - Figura 21 - Coqueiros Simplificado F_24  ===



% === Figura 22 - Coqueiros Simplificado F_25 ===

% === Entradas Circuito F_25 ===
% vI64_28 (Disjuntor 52AX Isolado - Saída Circuito F_28)
% v01_53 e v01_63 (Proteção Sobretensão UPD1 e UPD2 - Saídas Circuitos F_53 e
F_63)
% vI53_27 (Relé 43N Atuado - Saída Circuito F_27)
% vI28_86 (Desbloqueio Disjunto 52AX - Saída Circuito F_86)
% eP01_32 (Seleção Local/Remoto-Remoto - Declarada no Circuito F_19)

% === Falha Disjuntor 52AX
initially(dI4L(0)).

 terminates(dI4Lup,dI4L(0),T) :- timepoint(T).
   initiates(dI4Lup,dI4L(1),T) :- timepoint(T).
 terminates(dI4Ldw,dI4L(1),T) :- timepoint(T).
   initiates(dI4Ldw,dI4L(0),T) :- timepoint(T).

happens(dI4Ldw,1).
happens(dI4Lup,18).
```

```
% === Mola Fechamento Descarregada Disjuntor 52AX
initially(dI14N_405(0)).

 terminates(dI14N_405up,dI14N_405(0),T) :- timepoint(T).
   initiates(dI14N_405up,dI14N_405(1),T) :- timepoint(T).
 terminates(dI14N_405dw,dI14N_405(1),T) :- timepoint(T).
   initiates(dI14N_405dw,dI14N_405(0),T) :- timepoint(T).

happens(dI14N_405dw,1).
happens(dI14N_405up,18).


% === (IHM-UCD1) Desbloqueio Disjuntor 52AX
initially(eP15_32(0)).

 terminates(eP15_32up,eP15_32(0),T) :- timepoint(T).
   initiates(eP15_32up,eP15_32(1),T) :- timepoint(T).
 terminates(eP15_32dw,eP15_32(1),T) :- timepoint(T).
   initiates(eP15_32dw,eP15_32(0),T) :- timepoint(T).

happens(eP15_32dw,1).


%============ Flip-flop SR Circuito F25_1 ============
initially(qout_CF25_1(0)).
happens(i2_CF25_1_sup,T1) :- holdsAt(sin_CF25_1(0),T1),
holdsAt(sin_CF25_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF25_1_sdw,T1) :- holdsAt(sin_CF25_1(1),T1),
holdsAt(sin_CF25_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF25_1_rup,T1) :- holdsAt(rin_CF25_1(0),T1),
holdsAt(rin_CF25_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF25_1_rdw,T1) :- holdsAt(rin_CF25_1(1),T1),
holdsAt(rin_CF25_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF25_1_sup,qout_CF25_1(0),T) :- holdsAt(rin_CF25_1(0),T),
holdsAt(qout_CF25_1(0),T), timepoint(T).
 initiates(i2_CF25_1_sup,qout_CF25_1(1),T) :- holdsAt(rin_CF25_1(0),T),
holdsAt(qout_CF25_1(0),T), timepoint(T).
terminates(i2_CF25_1_rup,qout_CF25_1(1),T) :- holdsAt(sin_CF25_1(0),T),
holdsAt(qout_CF25_1(1),T), timepoint(T).
 initiates(i2_CF25_1_rup,qout_CF25_1(0),T) :- holdsAt(sin_CF25_1(0),T),
holdsAt(qout_CF25_1(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF25_1_rup,qout_CF25_1(1),T) :- holdsAt(sin_CF25_1(1),T),
holdsAt(qout_CF25_1(1),T), timepoint(T).
 initiates(i2_CF25_1_rup,qout_CF25_1(0),T) :- holdsAt(sin_CF25_1(1),T),
holdsAt(qout_CF25_1(1),T), timepoint(T).

%============ Flip-flop SR Circuito F25_2 ============
```

```
initially(qout_CF25_2(0)).
happens(i2_CF25_2_sup,T1) :- holdsAt(sin_CF25_2(0),T1),
holdsAt(sin_CF25_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF25_2_sdw,T1) :- holdsAt(sin_CF25_2(1),T1),
holdsAt(sin_CF25_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF25_2_rup,T1) :- holdsAt(rin_CF25_2(0),T1),
holdsAt(rin_CF25_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF25_2_rdw,T1) :- holdsAt(rin_CF25_2(1),T1),
holdsAt(rin_CF25_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF25_2_sup,qout_CF25_2(0),T) :- holdsAt(rin_CF25_2(0),T),
holdsAt(qout_CF25_2(0),T), timepoint(T).
 initiates(i2_CF25_2_sup,qout_CF25_2(1),T) :- holdsAt(rin_CF25_2(0),T),
holdsAt(qout_CF25_2(0),T), timepoint(T).
terminates(i2_CF25_2_rup,qout_CF25_2(1),T) :- holdsAt(sin_CF25_2(0),T),
holdsAt(qout_CF25_2(1),T), timepoint(T).
 initiates(i2_CF25_2_rup,qout_CF25_2(0),T) :- holdsAt(sin_CF25_2(0),T),
holdsAt(qout_CF25_2(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF25_2_rup,qout_CF25_2(1),T) :- holdsAt(sin_CF25_2(1),T),
holdsAt(qout_CF25_2(1),T), timepoint(T).
 initiates(i2_CF25_2_rup,qout_CF25_2(0),T) :- holdsAt(sin_CF25_2(1),T),
holdsAt(qout_CF25_2(1),T), timepoint(T).

%============ Drop-out timer Circuito F25_1 ====================
initially(monoOutput_DO_CF25_1(0)).
happens(upTriger_DO_CF25_1,0) :- initially(inputIs_DO_CF25_1(1)).
happens(upTriger_DO_CF25_1,T2) :- holdsAt(inputIs_DO_CF25_1(0),T1),
holdsAt(inputIs_DO_CF25_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(upTriger_DO_CF25_1,monoOutput_DO_CF25_1(0),T) :-
happens(upTriger_DO_CF25_1,T), holdsAt(inputIs_DO_CF25_1(1),T), timepoint(T).
initiates(upTriger_DO_CF25_1,monoOutput_DO_CF25_1(1),T)  :-
happens(upTriger_DO_CF25_1,T), holdsAt(inputIs_DO_CF25_1(1),T), timepoint(T).

happens(downTriger_DO_CF25_1,T3) :- holdsAt(inputIs_DO_CF25_1(1),T1),
holdsAt(inputIs_DO_CF25_1(0),T2), T2=T1+1, T3=T1+5, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(downTriger_DO_CF25_1,monoOutput_DO_CF25_1(1),T) :-
happens(downTriger_DO_CF25_1,T), holdsAt(monoOutput_DO_CF25_1(1),T),
timepoint(T).
initiates(downTriger_DO_CF25_1,monoOutput_DO_CF25_1(0),T)  :-
happens(downTriger_DO_CF25_1,T), holdsAt(monoOutput_DO_CF25_1(1),T),
timepoint(T).


% === SAÍDAS Circuito F_25 ===

% vI41_25 - Desbloqueio Secionadoras Isoladoras (Falha Disjuntor 52AX)
holdsAt(sin_CF25_1(A),T) :- holdsAt(dI4L(A),T), timepoint(T).
holdsAt(rin_CF25_1(B),T) :- holdsAt(vI64_28(B),T), timepoint(T).
```

```
holdsAt(inputIs_DO_CF25_1(C),T) :- holdsAt(qout_CF25_1(C),T), timepoint(T).
holdsAt(vI41_25(D),T) :- holdsAt(monoOutput_DO_CF25_1(D),T), timepoint(T).

% vI43_25 - Bloqueio Fechamento Disjuntor 52AX
holdsAt(sin_CF25_2(M),T) :- and(J,F,M), or(C,E,J), or(A,B,C),
holdsAt(vI01_53(A),T), holdsAt(vI01_63(B),T), holdsAt(dI14N_405(E),T),
holdsAt(vI53_27(F),T), nivelLogico(J), nivelLogico(M), nivelLogico(C), timepoint(T).
holdsAt(rin_CF25_2(N),T) :- or(K,L,N), and(G,H,K), and(X,I,L), inv(H,X),
holdsAt(vI28_86(G),T), holdsAt(eP01_32(H),T), holdsAt(eP15_32(I),T),
nivelLogico(K), nivelLogico(L), nivelLogico(N), nivelLogico(X), timepoint(T).
holdsAt(vI43_25(P),T) :- holdsAt(qout_CF25_2(P),T), timepoint(T).

%= FIM - Figura 22 - Coqueiros Simplificado F_25  ===



% === Figura 23 - Coqueiros Simplificado F_26 ===

% === Entradas Circuito F_26 ===
% vI49_27   (Reset da Função "T" - Automática - Saída Circuito F_27)
% vI46_27   (Set da Função "N" - Automática - Saída Circuito F_27)
% vI48_27   (Set da Função "T" - Automática - Saída Circuito F_27)
% vI47_27   (Reset da Função "N" - Automática - Saída Circuito F_27)
% eP01_32   (Seleção Local/Remoto-Remoto - Declarada no Circuito F_19)
% vI29_87   (Seleção Transferência Proteção Normal "N" - Saída Circuito F_87)
% vI09_16   (Secionadora 84AX4 Aberta - Saída Circuito F_16)
% vI51_27   (Proteção em Transferência "ET" - Saída Circuito F_27)
% vI30_87   (Seleção Transferência Proteção em Transferência "ET" - Saída Circuito
F_87)
% vI31_87   (Seleção Transferência Proteção Transferida "T" - Saída Circuito F_87)
% vI10_16   (Secionadora 84AX4 Fechada - Saída Circuito F_16)
% vI62_28   (Seção AX Aberta - Saída Circuito F_28)
% vI54_27   (Transferência de Proteção Relé "43T" Atuado - Saída Circuito F_27)
% vI63_28   (Disjuntor 52AX Fechamento para Manobra - Saída Circuito F_28)
% vI16_24   (Comando Permissão Fechamento Disjuntor 52AX - Saída Circuito F_24)

% === (IHM-UCD1) Seleção Transferência Proteção Normal "N"
initially(eP16_32(0)).

 terminates(eP16_32up,eP16_32(0),T) :- timepoint(T).
   initiates(eP16_32up,eP16_32(1),T) :- timepoint(T).
 terminates(eP16_32dw,eP16_32(1),T) :- timepoint(T).
   initiates(eP16_32dw,eP16_32(0),T) :- timepoint(T).

happens(eP16_32dw,1).


% === (IHM-UCD1) Seleção Transferência Proteção Em Transferência "ET"
initially(eP17_32(0)).
```

```
 terminates(eP17_32up,eP17_32(0),T) :- timepoint(T).
   initiates(eP17_32up,eP17_32(1),T) :- timepoint(T).
 terminates(eP17_32dw,eP17_32(1),T) :- timepoint(T).
   initiates(eP17_32dw,eP17_32(0),T) :- timepoint(T).

happens(eP17_32dw,1).


% === (IHM-UCD1) Seleção Transferência Proteção Transferida "T"
initially(eP18_32(0)).

 terminates(eP18_32up,eP18_32(0),T) :- timepoint(T).
   initiates(eP18_32up,eP18_32(1),T) :- timepoint(T).
 terminates(eP18_32dw,eP18_32(1),T) :- timepoint(T).
   initiates(eP18_32dw,eP18_32(0),T) :- timepoint(T).

happens(eP18_32dw,1).


% === Nenhum Relé 43T Atuado
initially(vI33_26(0)).

 terminates(vI33_26up,vI33_26(0),T) :- timepoint(T).
   initiates(vI33_26up,vI33_26(1),T) :- timepoint(T).
 terminates(vI33_26dw,vI33_26(1),T) :- timepoint(T).
   initiates(vI33_26dw,vI33_26(0),T) :- timepoint(T).

happens(vI33_26dw,1).


% === Todas Secionadoras BY PASS Adjacentes Seção AX Abertas
initially(vI35_26(0)).

 terminates(vI35_26up,vI35_26(0),T) :- timepoint(T).
   initiates(vI35_26up,vI35_26(1),T) :- timepoint(T).
 terminates(vI35_26dw,vI35_26(1),T) :- timepoint(T).
   initiates(vI35_26dw,vI35_26(0),T) :- timepoint(T).

happens(vI35_26dw,1).


% === Todas Secionadoras BARRA 2 Adjacentes Seção AX Abertas
initially(vI34_26(0)).

 terminates(vI34_26up,vI34_26(0),T) :- timepoint(T).
   initiates(vI34_26up,vI34_26(1),T) :- timepoint(T).
 terminates(vI34_26dw,vI34_26(1),T) :- timepoint(T).
   initiates(vI34_26dw,vI34_26(0),T) :- timepoint(T).
```

```
happens(vI34_26dw,1).


% === Transferência de Proteção em Manual
initially(vI31_26(0)).

 terminates(vI31_26up,vI31_26(0),T) :- timepoint(T).
   initiates(vI31_26up,vI31_26(1),T) :- timepoint(T).
 terminates(vI31_26dw,vI31_26(1),T) :- timepoint(T).
   initiates(vI31_26dw,vI31_26(0),T) :- timepoint(T).

happens(vI31_26dw,1).


%============ Pick-up timer Circuito F26_1 ============
initially(monoOutput_PU_CF26_1(0)).
happens(upTriger_PU_CF26_1,T3) :- holdsAt(inputIs_PU_CF26_1(0),T1),
holdsAt(inputIs_PU_CF26_1(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF26_1,monoOutput_PU_CF26_1(0),T) :-
happens(upTriger_PU_CF26_1,T), holdsAt(inputIs_PU_CF26_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF26_1,monoOutput_PU_CF26_1(1),T)  :-
happens(upTriger_PU_CF26_1,T), holdsAt(inputIs_PU_CF26_1(1),T), timepoint(T).

happens(downTriger_PU_CF26_1,0) :- initially(inputIs_PU_CF26_1(0)).
happens(downTriger_PU_CF26_1,T1) :- holdsAt(inputIs_PU_CF26_1(1),T1),
holdsAt(inputIs_PU_CF26_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF26_1,monoOutput_PU_CF26_1(1),T) :-
happens(downTriger_PU_CF26_1,T), holdsAt(monoOutput_PU_CF26_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF26_1,monoOutput_PU_CF26_1(0),T)  :-
happens(downTriger_PU_CF26_1,T), holdsAt(monoOutput_PU_CF26_1(1),T),
timepoint(T).

%============ Pick-up timer Circuito F26_2 ============
initially(monoOutput_PU_CF26_2(0)).
happens(upTriger_PU_CF26_2,T3) :- holdsAt(inputIs_PU_CF26_2(0),T1),
holdsAt(inputIs_PU_CF26_2(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF26_2,monoOutput_PU_CF26_2(0),T) :-
happens(upTriger_PU_CF26_2,T), holdsAt(inputIs_PU_CF26_2(1),T), timepoint(T).
 initiates(upTriger_PU_CF26_2,monoOutput_PU_CF26_2(1),T)  :-
happens(upTriger_PU_CF26_2,T), holdsAt(inputIs_PU_CF26_2(1),T), timepoint(T).

happens(downTriger_PU_CF26_2,0) :- initially(inputIs_PU_CF26_2(0)).
happens(downTriger_PU_CF26_2,T1) :- holdsAt(inputIs_PU_CF26_2(1),T1),
holdsAt(inputIs_PU_CF26_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
```

```
terminates(downTriger_PU_CF26_2,monoOutput_PU_CF26_2(1),T) :-
happens(downTriger_PU_CF26_2,T), holdsAt(monoOutput_PU_CF26_2(1),T),
timepoint(T).
 initiates(downTriger_PU_CF26_2,monoOutput_PU_CF26_2(0),T)  :-
happens(downTriger_PU_CF26_2,T), holdsAt(monoOutput_PU_CF26_2(1),T),
timepoint(T).


% === SAÍDAS Circuito F_26 ===
% Partes comuns a algumas saídas
holdsAt(cA_26(CA),T) :- and3(BA,U,AC,CA), or(AA,AB,BA), and(H,I,AC),
and(E,F,AA), and(NE,G,AB), inv(E,NE), holdsAt(eP01_32(E),T),
holdsAt(vI29_87(F),T), holdsAt(eP16_32(G),T), holdsAt(vI09_16(H),T),
holdsAt(vI51_27(I),T), holdsAt(vI31_26(U),T), nivelLogico(CA), nivelLogico(BA),
nivelLogico(AA), nivelLogico(AB), nivelLogico(AC), nivelLogico(NE), timepoint(T).
holdsAt(cB_26(CB),T) :- and3(BB,U,BD,CA), or(AD,AE,BB), and(E,J,AD),
and(NE,K,AE), inv(E,NE), or(AI,S,BD), and4(H,P,Q,R,AI), holdsAt(eP01_32(E),T),
holdsAt(vI30_87(J),T), holdsAt(eP17_32(K),T), holdsAt(vI33_26(P),T),
holdsAt(vI35_26(Q),T), holdsAt(vI34_26(R),T), holdsAt(vI54_27(S),T),
holdsAt(vI31_26(U),T), nivelLogico(CB), nivelLogico(BB), nivelLogico(BC),
nivelLogico(AD), nivelLogico(AE), nivelLogico(AI), timepoint(T).
holdsAt(inputIs_PU_CF26_1(W),T) :- holdsAt(vI16_24(W),T), timepoint(T).
holdsAt(dF_26(DF),T) :- and(AH,NDE,DF), inv(DE,NDE), and3(I,N,O,AH),
and(U,CD,DE), or(NO,BE,CD), inv(O,NO), and(V,AF,BE), holdsAt(vI51_27(I),T),
holdsAt(vI10_16(N),T), holdsAt(vI62_28(O),T), holdsAt(vI31_26(U),T),
holdsAt(vI63_28(V),T), holdsAt(monoOutput_PU_CF26_1(AF),T), nivelLogico(DF),
nivelLogico(AH), nivelLogico(BC), nivelLogico(DE), nivelLogico(NDE),
nivelLogico(CD), nivelLogico(BE), timepoint(T).
holdsAt(inputIs_PU_CF26_2(DF),T) :- holdsAt(dF_26(DF),T), timepoint(T).
holdsAt(cC_26(CC),T) :- and3(BC,U,DG,CC), or(AF,AG,BC), and(E,L,AF),
and(NE,M,AG), inv(E,NE), holdsAt(eP01_32(E),T), holdsAt(vI31_87(L),T),
holdsAt(eP18_32(M),T), holdsAt(vI31_26(U),T),
holdsAt(monoOutput_PU_CF26_2(DG),T), nivelLogico(CC), nivelLogico(BC),
timepoint(T).

% dO7E - Desatuar Relé "43T"
holdsAt(dO7E(DA),T) :- or(A,CA,DA), holdsAt(vI49_27(A),T),
holdsAt(cA_26(CA),T), nivelLogico(DA), timepoint(T).

% dO5E - Atuar Relé "43N"
holdsAt(dO5E(DB),T) :- or3(CA,B,CB,DB), holdsAt(cA_26(CA),T),
holdsAt(vI46_27(B),T), holdsAt(cB_26(CB),T), nivelLogico(DB), timepoint(T).

% dO6E - Desatuar Relé "43N"
holdsAt(dO6E(DC),T) :- or(D,CC,DC), holdsAt(vI47_27(D),T),
holdsAt(cC_26(CC),T), nivelLogico(DC), timepoint(T).

% dO8E - Atuar Relé "43T"
```

```
holdsAt(dO8E(DD),T) :- or3(CB,C,CC,DD), holdsAt(cB_26(CB),T),
holdsAt(vI48_27(C),T), holdsAt(cC_26(CC),T), nivelLogico(DD), timepoint(T).

% vI44_26 - Transferência de Proteção em Manual
holdsAt(vI44_26(U),T) :- holdsAt(vI31_26(U),T), timepoint(T).

%= FIM - Figura 23 - Coqueiros Simplificado F_26  ===



% === Figura 24 - Coqueiros Simplificado F_27 ===

% === Entradas Circuito F_27 ===
% vI62_28   (Seção AX Aberta - Saída Circuito F_28)
% vI63_28   (Disjuntor 52AX Fechamento para Manobra - Saída Circuito F_28)
% vI38_24   (Comando/Permissão Fechamento Disjuntor 52AX - Saída Circuito F_24)
% vI10_16   (Secionadora 84AX4 Fechada - Saída Circuito F_16)
% vI35_23   (Fechar Secionadora 89AX4 - Saída Circuito F_23)
% vI09_16   (Secionadora 84AX4 Aberta - Saída Circuito F_16)
% vI14_17   (Secionadora 89AX1 Fechada - Saída Circuito F_17)
% vI18_17   (Secionadora 89AX3 Fechada - Saída Circuito F_17)

% === Transferência de Proteção em Automático
initially(vI32_27(1)).

 terminates(vI32_27up,vI32_27(0),T) :- timepoint(T).
   initiates(vI32_27up,vI32_27(1),T) :- timepoint(T).
 terminates(vI32_27dw,vI32_27(1),T) :- timepoint(T).
   initiates(vI32_27dw,vI32_27(0),T) :- timepoint(T).

happens(vI32_27up,1).


% === Transferência de Proteção Relé 43N - Atuado
initially(dI11L_409(0)).

 terminates(dI11L_409up,dI11L_409(0),T) :- timepoint(T).
   initiates(dI11L_409up,dI11L_409(1),T) :- timepoint(T).
 terminates(dI11L_409dw,dI11L_409(1),T) :- timepoint(T).
   initiates(dI11L_409dw,dI11L_409(0),T) :- timepoint(T).

happens(dI11L_409dw,1).
happens(dI11L_409up,17).


% === Transferência de Proteção Relé 43T - Atuado
initially(dI12L_409(0)).

 terminates(dI12L_409up,dI12L_409(0),T) :- timepoint(T).
```

```
  initiates(dI12L_409up,dI12L_409(1),T) :- timepoint(T).
terminates(dI12L_409dw,dI12L_409(1),T) :- timepoint(T).
  initiates(dI12L_409dw,dI12L_409(0),T) :- timepoint(T).

happens(dI12L_409dw,1).


%============ Drop-out timer Circuito F27_1 ====================
initially(monoOutput_DO_CF27_1(0)).
happens(upTriger_DO_CF27_1,0) :- initially(inputIs_DO_CF27_1(1)).
happens(upTriger_DO_CF27_1,T2) :- holdsAt(inputIs_DO_CF27_1(0),T1),
holdsAt(inputIs_DO_CF27_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(upTriger_DO_CF27_1,monoOutput_DO_CF27_1(0),T) :-
happens(upTriger_DO_CF27_1,T), holdsAt(inputIs_DO_CF27_1(1),T), timepoint(T).
initiates(upTriger_DO_CF27_1,monoOutput_DO_CF27_1(1),T)  :-
happens(upTriger_DO_CF27_1,T), holdsAt(inputIs_DO_CF27_1(1),T), timepoint(T).

happens(downTriger_DO_CF27_1,T3) :- holdsAt(inputIs_DO_CF27_1(1),T1),
holdsAt(inputIs_DO_CF27_1(0),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(downTriger_DO_CF27_1,monoOutput_DO_CF27_1(1),T) :-
happens(downTriger_DO_CF27_1,T), holdsAt(monoOutput_DO_CF27_1(1),T),
timepoint(T).
initiates(downTriger_DO_CF27_1,monoOutput_DO_CF27_1(0),T)  :-
happens(downTriger_DO_CF27_1,T), holdsAt(monoOutput_DO_CF27_1(1),T),
timepoint(T).

%============ Drop-out timer Circuito F27_2 ====================
initially(monoOutput_DO_CF27_2(0)).
happens(upTriger_DO_CF27_2,0) :- initially(inputIs_DO_CF27_2(1)).
happens(upTriger_DO_CF27_2,T2) :- holdsAt(inputIs_DO_CF27_2(0),T1),
holdsAt(inputIs_DO_CF27_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(upTriger_DO_CF27_2,monoOutput_DO_CF27_2(0),T) :-
happens(upTriger_DO_CF27_2,T), holdsAt(inputIs_DO_CF27_2(1),T), timepoint(T).
initiates(upTriger_DO_CF27_2,monoOutput_DO_CF27_2(1),T)  :-
happens(upTriger_DO_CF27_2,T), holdsAt(inputIs_DO_CF27_2(1),T), timepoint(T).

happens(downTriger_DO_CF27_2,T3) :- holdsAt(inputIs_DO_CF27_2(1),T1),
holdsAt(inputIs_DO_CF27_2(0),T2), T2=T1+1, T3=T1+30, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(downTriger_DO_CF27_2,monoOutput_DO_CF27_2(1),T) :-
happens(downTriger_DO_CF27_2,T), holdsAt(monoOutput_DO_CF27_2(1),T),
timepoint(T).
initiates(downTriger_DO_CF27_2,monoOutput_DO_CF27_2(0),T)  :-
happens(downTriger_DO_CF27_2,T), holdsAt(monoOutput_DO_CF27_2(1),T),
timepoint(T).

%============ Pick-up timer Circuito F27_1 ============
initially(monoOutput_PU_CF27_1(0)).
```

```
happens(upTriger_PU_CF27_1,T3) :- holdsAt(inputIs_PU_CF27_1(0),T1),
holdsAt(inputIs_PU_CF27_1(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF27_1,monoOutput_PU_CF27_1(0),T) :-
happens(upTriger_PU_CF27_1,T), holdsAt(inputIs_PU_CF27_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF27_1,monoOutput_PU_CF27_1(1),T)  :-
happens(upTriger_PU_CF27_1,T), holdsAt(inputIs_PU_CF27_1(1),T), timepoint(T).

happens(downTriger_PU_CF27_1,0) :- initially(inputIs_PU_CF27_1(0)).
happens(downTriger_PU_CF27_1,T1) :- holdsAt(inputIs_PU_CF27_1(1),T1),
holdsAt(inputIs_PU_CF27_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF27_1,monoOutput_PU_CF27_1(1),T) :-
happens(downTriger_PU_CF27_1,T), holdsAt(monoOutput_PU_CF27_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF27_1,monoOutput_PU_CF27_1(0),T)  :-
happens(downTriger_PU_CF27_1,T), holdsAt(monoOutput_PU_CF27_1(1),T),
timepoint(T).

%============ Pick-up timer Circuito F27_2 ============
initially(monoOutput_PU_CF27_2(0)).
happens(upTriger_PU_CF27_2,T3) :- holdsAt(inputIs_PU_CF27_2(0),T1),
holdsAt(inputIs_PU_CF27_2(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF27_2,monoOutput_PU_CF27_2(0),T) :-
happens(upTriger_PU_CF27_2,T), holdsAt(inputIs_PU_CF27_2(1),T), timepoint(T).
 initiates(upTriger_PU_CF27_2,monoOutput_PU_CF27_2(1),T)  :-
happens(upTriger_PU_CF27_2,T), holdsAt(inputIs_PU_CF27_2(1),T), timepoint(T).

happens(downTriger_PU_CF27_2,0) :- initially(inputIs_PU_CF27_2(0)).
happens(downTriger_PU_CF27_2,T1) :- holdsAt(inputIs_PU_CF27_2(1),T1),
holdsAt(inputIs_PU_CF27_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF27_2,monoOutput_PU_CF27_2(1),T) :-
happens(downTriger_PU_CF27_2,T), holdsAt(monoOutput_PU_CF27_2(1),T),
timepoint(T).
 initiates(downTriger_PU_CF27_2,monoOutput_PU_CF27_2(0),T)  :-
happens(downTriger_PU_CF27_2,T), holdsAt(monoOutput_PU_CF27_2(1),T),
timepoint(T).

%============ Pick-up timer Circuito F27_3 ============
initially(monoOutput_PU_CF27_3(0)).
happens(upTriger_PU_CF27_3,T3) :- holdsAt(inputIs_PU_CF27_3(0),T1),
holdsAt(inputIs_PU_CF27_3(1),T2), T2=T1+1, T3=T1+1, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF27_3,monoOutput_PU_CF27_3(0),T) :-
happens(upTriger_PU_CF27_3,T), holdsAt(inputIs_PU_CF27_3(1),T), timepoint(T).
 initiates(upTriger_PU_CF27_3,monoOutput_PU_CF27_3(1),T)  :-
happens(upTriger_PU_CF27_3,T), holdsAt(inputIs_PU_CF27_3(1),T), timepoint(T).

happens(downTriger_PU_CF27_3,0) :- initially(inputIs_PU_CF27_3(0)).
```

```
happens(downTriger_PU_CF27_3,T1) :- holdsAt(inputIs_PU_CF27_3(1),T1),
holdsAt(inputIs_PU_CF27_3(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF27_3,monoOutput_PU_CF27_3(1),T) :-
happens(downTriger_PU_CF27_3,T), holdsAt(monoOutput_PU_CF27_3(1),T),
timepoint(T).
 initiates(downTriger_PU_CF27_3,monoOutput_PU_CF27_3(0),T)  :-
happens(downTriger_PU_CF27_3,T), holdsAt(monoOutput_PU_CF27_3(1),T),
timepoint(T).

%============ Pick-up timer Circuito F27_4 ============
initially(monoOutput_PU_CF27_4(0)).
happens(upTriger_PU_CF27_4,T3) :- holdsAt(inputIs_PU_CF27_4(0),T1),
holdsAt(inputIs_PU_CF27_4(1),T2), T2=T1+1, T3=T1+3, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF27_4,monoOutput_PU_CF27_4(0),T) :-
happens(upTriger_PU_CF27_4,T), holdsAt(inputIs_PU_CF27_4(1),T), timepoint(T).
 initiates(upTriger_PU_CF27_4,monoOutput_PU_CF27_4(1),T)  :-
happens(upTriger_PU_CF27_4,T), holdsAt(inputIs_PU_CF27_4(1),T), timepoint(T).

happens(downTriger_PU_CF27_4,0) :- initially(inputIs_PU_CF27_4(0)).
happens(downTriger_PU_CF27_4,T1) :- holdsAt(inputIs_PU_CF27_4(1),T1),
holdsAt(inputIs_PU_CF27_4(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF27_4,monoOutput_PU_CF27_4(1),T) :-
happens(downTriger_PU_CF27_4,T), holdsAt(monoOutput_PU_CF27_4(1),T),
timepoint(T).
 initiates(downTriger_PU_CF27_4,monoOutput_PU_CF27_4(0),T)  :-
happens(downTriger_PU_CF27_4,T), holdsAt(monoOutput_PU_CF27_4(1),T),
timepoint(T).

%============ Pick-up timer Circuito F27_5 ============
initially(monoOutput_PU_CF27_5(0)).
happens(upTriger_PU_CF27_5,T3) :- holdsAt(inputIs_PU_CF27_5(0),T1),
holdsAt(inputIs_PU_CF27_5(1),T2), T2=T1+1, T3=T1+120, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF27_5,monoOutput_PU_CF27_5(0),T) :-
happens(upTriger_PU_CF27_5,T), holdsAt(inputIs_PU_CF27_5(1),T), timepoint(T).
 initiates(upTriger_PU_CF27_5,monoOutput_PU_CF27_5(1),T)  :-
happens(upTriger_PU_CF27_5,T), holdsAt(inputIs_PU_CF27_5(1),T), timepoint(T).

happens(downTriger_PU_CF27_5,0) :- initially(inputIs_PU_CF27_5(0)).
happens(downTriger_PU_CF27_5,T1) :- holdsAt(inputIs_PU_CF27_5(1),T1),
holdsAt(inputIs_PU_CF27_5(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF27_5,monoOutput_PU_CF27_5(1),T) :-
happens(downTriger_PU_CF27_5,T), holdsAt(monoOutput_PU_CF27_5(1),T),
timepoint(T).
 initiates(downTriger_PU_CF27_5,monoOutput_PU_CF27_5(0),T)  :-
happens(downTriger_PU_CF27_5,T), holdsAt(monoOutput_PU_CF27_5(1),T),
timepoint(T).
```

```
% === SAÍDAS Circuito F_27 ===
% Partes comuns a algumas saídas
holdsAt(inputIs_DO_CF27_1(C),T) :- holdsAt(vI38_24(C),T), timepoint(T).
holdsAt(cA_27(CA),T) :- and(BA,D,CA), or(NA,AA,BA), and(B,AB,AA), inv(A,NA),
holdsAt(vI62_28(A),T), holdsAt(vI63_28(B),T),
holdsAt(monoOutput_DO_CF27_1(AB),T), holdsAt(vI32_27(D),T), nivelLogico(CA),
nivelLogico(BA), nivelLogico(AA), nivelLogico(NA), timepoint(T).
holdsAt(bB_27(BB),T) :- and3(NCA,A,E,BB), inv(CA,NCA), holdsAt(cA_27(CA),T),
holdsAt(vI62_28(A),T), holdsAt(vI10_16(E),T), nivelLogico(BB), nivelLogico(NCA),
timepoint(T).
holdsAt(inputIs_PU_CF27_1(BB),T) :- holdsAt(bB_27(BB),T), timepoint(T).
holdsAt(inputIs_DO_CF27_2(F),T) :- holdsAt(vI35_23(F),T), timepoint(T).
holdsAt(cC_27(CC),T) :- and(D,BC,CC), or(E,AC,BC), holdsAt(vI32_27(D),T),
holdsAt(vI10_16(E),T), holdsAt(monoOutput_DO_CF27_2(AC),T), nivelLogico(CC),
nivelLogico(BC), timepoint(T).
holdsAt(inputIs_PU_CF27_2(AD),T) :- and(NCC,G,AD), inv(CC,NCC),
holdsAt(cC_27(CC),T), holdsAt(vI09_16(G),T), nivelLogico(AD), nivelLogico(NCC),
timepoint(T).
holdsAt(aE_27(AE),T) :- and(H,NI,AE), inv(I,NI), holdsAt(dI11L_409(H),T),
holdsAt(dI12L_409(I),T), nivelLogico(AE), nivelLogico(NI), timepoint(T).
holdsAt(aF_27(AF),T) :- and(I,H,AF), holdsAt(dI12L_409(I),T),
holdsAt(dI11L_409(H),T), nivelLogico(AF), timepoint(T).
holdsAt(inputIs_PU_CF27_3(AJ),T) :- and(NH,NI,AJ), inv(H,NH), inv(I,NI),
holdsAt(dI11L_409(H),T), holdsAt(dI12L_409(I),T), nivelLogico(AJ),
nivelLogico(NH), nivelLogico(NI), timepoint(T).
holdsAt(inputIs_PU_CF27_4(AF),T) :- holdsAt(aF_27(AF),T), timepoint(T).
holdsAt(inputIs_PU_CF27_5(AK),T) :- and(J,K,AK), holdsAt(vI14_17(J),T),
holdsAt(vI18_17(K),T), nivelLogico(AK), timepoint(T).

% vI45_27 - Transferência de Proteção em Automático
holdsAt(vI45_27(D),T) :- holdsAt(vI32_27(D),T), timepoint(T).

% vO3_27 - Virtual Output 3 - Transferência de Proteção em Automático
holdsAt(vO3_27(D),T) :- holdsAt(vI32_27(D),T), timepoint(T).

% vI46_27 - Set da Função "N" (Automática)
holdsAt(vI46_27(CA),T) :- holdsAt(cA_27(CA),T), timepoint(T).

% vI47_27 - Reset da Função "N" (Automática)
holdsAt(vI47_27(DA),T) :- and(CB,D,DA), holdsAt(monoOutput_PU_CF27_1(CB),T),
holdsAt(vI32_27(D),T), nivelLogico(DA), timepoint(T).

% vI48_27 - Set da Função "T" (Automática)
holdsAt(vI48_27(CC),T) :- holdsAt(cC_27(CC),T), timepoint(T).

% vI49_27 - Reset da Função "T" (Automática)
holdsAt(vI49_27(DB),T) :- and(D,CD,DB), holdsAt(vI32_27(D),T),
holdsAt(monoOutput_PU_CF27_2(CD),T), nivelLogico(DB), timepoint(T).
```

```
% vI50_27 - Proteção da Seção AX (N) - Normal
holdsAt(vI50_27(AE),T) :- holdsAt(aE_27(AE),T), timepoint(T).

% vO1_27 - Virtual Output 1 - Proteção da Seção AX (N) - Normal
holdsAt(vO1_27(AE),T) :- holdsAt(aE_27(AE),T), timepoint(T).

% vI51_27 - Proteção da Seção AX (ET) - Em Transferência
holdsAt(vI51_27(AF),T) :- holdsAt(aF_27(AF),T), timepoint(T).

% vO2_27 - Virtual Output 2 - Proteção da Seção AX (ET) - Em Transferência
holdsAt(vO1_27(AF),T) :- holdsAt(aF_27(AF),T), timepoint(T).

% vI52_27 - Proteção da Seção AX (T) - Transferida
holdsAt(vI52_27(AG),T) :- and(NH,I,AG), inv(H,NH), holdsAt(dI11L_409(H),T),
holdsAt(dI12L_409(I),T), nivelLogico(AG), nivelLogico(NH), timepoint(T).

% vI53_27 - Transferência Proteção Relé 43N - Atuado
holdsAt(vI53_27(H),T) :- holdsAt(dI11L_409(H),T), timepoint(T).

% vI54_27 - Transferência Proteção Relé 43T - Atuado
holdsAt(vI54_27(I),T) :- holdsAt(dI12L_409(I),T), timepoint(T).

% vI55_27 - Proteção da Seção AX Indefinida
holdsAt(vI55_27(BD),T) :- holdsAt(monoOutput_PU_CF27_3(BD),T), timepoint(T).

% vI56_27 - Transferência de Proteção Incompleta
holdsAt(vI56_27(BE),T) :- holdsAt(monoOutput_PU_CF27_4(BE),T), timepoint(T).

% vI57_27 - Transferência de Barras Incompleta
holdsAt(vI57_27(BF),T) :- holdsAt(monoOutput_PU_CF27_5(BF),T), timepoint(T).

%= FIM - Figura 24 - Coqueiros Simplificado F_27  ===



% === Figura 25 - Coqueiros Simplificado F_28 ===

% === Entradas Circuito F_28 ===
% vI45_27   (Transferência de Proteção em Automático - Saída Circuito F_27)
% vI50_27   (Proteção da Seção AX - Normal (N) - Saída Circuito F_27)
% vI51_27   (Transferência de Proteção - Em Transferência (ET) - Saída Circuito F_27)
% vI21_18   (Disjuntor 52AX Aberto - Saída Circuito F_18)
% vI06_15   (Secionadora 89AX2 Fechada - Saída Circuito F_15)
% vI14_17   (Secionadora 89AX1 Fechada - Saída Circuito F_15)
% vI18_17   (Secionadora 89AX3 Fechada - Saída Circuito F_17)
% vI64_28   (Disjuntor 52AX Isolado - Saída Circuito F_28)
% vI05_15   (Secionadora 89AX2 Aberta - Saída Circuito F_15)
% vI13_17   (Secionadora 89AX1 Aberta - Saída Circuito F_17)
```

```
% vI17_17   (Secionadora 84AX3 Aberta - Saída Circuito F_17)


% === TPAX (Enrolamento 1X) - Fechado
initially(dI3L(1)).

 terminates(dI3Lup,dI3L(0),T) :- timepoint(T).
   initiates(dI3Lup,dI3L(1),T) :- timepoint(T).
 terminates(dI3Ldw,dI3L(1),T) :- timepoint(T).
   initiates(dI3Ldw,dI3L(0),T) :- timepoint(T).

happens(dI3Lup,1).


% === TPB1 (Enrolamento 1X) - Fechado
initially(dI1L(1)).

 terminates(dI1Lup,dI1L(0),T) :- timepoint(T).
   initiates(dI1Lup,dI1L(1),T) :- timepoint(T).
 terminates(dI1Ldw,dI1L(1),T) :- timepoint(T).
   initiates(dI1Ldw,dI1L(0),T) :- timepoint(T).

happens(dI1Lup,1).


% === TPB2 (Enrolamento 1X) - Fechado
initially(dI2L(1)).

 terminates(dI2Lup,dI2L(0),T) :- timepoint(T).
   initiates(dI2Lup,dI2L(1),T) :- timepoint(T).
 terminates(dI2Ldw,dI2L(1),T) :- timepoint(T).
   initiates(dI2Ldw,dI2L(0),T) :- timepoint(T).

happens(dI2Lup,1).


% === Sincronismo Verificado UCD1
initially(vI65_34(0)).

 terminates(vI65_34up,vI65_34(0),T) :- timepoint(T).
   initiates(vI65_34up,vI65_34(1),T) :- timepoint(T).
 terminates(vI65_34dw,vI65_34(1),T) :- timepoint(T).
   initiates(vI65_34dw,vI65_34(0),T) :- timepoint(T).

happens(vI65_34dw,1).
happens(vI65_34up,15).


%============ Pick-up timer Circuito F28_1 ============
```

```
initially(monoOutput_PU_CF28_1(0)).
happens(upTriger_PU_CF28_1,T3) :- holdsAt(inputIs_PU_CF28_1(0),T1),
holdsAt(inputIs_PU_CF28_1(1),T2), T2=T1+1, T3=T1+1, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF28_1,monoOutput_PU_CF28_1(0),T) :-
happens(upTriger_PU_CF28_1,T), holdsAt(inputIs_PU_CF28_1(1),T), timepoint(T).
 initiates(upTriger_PU_CF28_1,monoOutput_PU_CF28_1(1),T)  :-
happens(upTriger_PU_CF28_1,T), holdsAt(inputIs_PU_CF28_1(1),T), timepoint(T).

happens(downTriger_PU_CF28_1,0) :- initially(inputIs_PU_CF28_1(0)).
happens(downTriger_PU_CF28_1,T1) :- holdsAt(inputIs_PU_CF28_1(1),T1),
holdsAt(inputIs_PU_CF28_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF28_1,monoOutput_PU_CF28_1(1),T) :-
happens(downTriger_PU_CF28_1,T), holdsAt(monoOutput_PU_CF28_1(1),T),
timepoint(T).
 initiates(downTriger_PU_CF28_1,monoOutput_PU_CF28_1(0),T)  :-
happens(downTriger_PU_CF28_1,T), holdsAt(monoOutput_PU_CF28_1(1),T),
timepoint(T).

%============ Pick-up timer Circuito F28_2 ============
initially(monoOutput_PU_CF28_2(0)).
happens(upTriger_PU_CF28_2,T3) :- holdsAt(inputIs_PU_CF28_2(0),T1),
holdsAt(inputIs_PU_CF28_2(1),T2), T2=T1+1, T3=T1+1, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF28_2,monoOutput_PU_CF28_2(0),T) :-
happens(upTriger_PU_CF28_2,T), holdsAt(inputIs_PU_CF28_2(1),T), timepoint(T).
 initiates(upTriger_PU_CF28_2,monoOutput_PU_CF28_2(1),T)  :-
happens(upTriger_PU_CF28_2,T), holdsAt(inputIs_PU_CF28_2(1),T), timepoint(T).

happens(downTriger_PU_CF28_2,0) :- initially(inputIs_PU_CF28_2(0)).
happens(downTriger_PU_CF28_2,T1) :- holdsAt(inputIs_PU_CF28_2(1),T1),
holdsAt(inputIs_PU_CF28_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF28_2,monoOutput_PU_CF28_2(1),T) :-
happens(downTriger_PU_CF28_2,T), holdsAt(monoOutput_PU_CF28_2(1),T),
timepoint(T).
 initiates(downTriger_PU_CF28_2,monoOutput_PU_CF28_2(0),T)  :-
happens(downTriger_PU_CF28_2,T), holdsAt(monoOutput_PU_CF28_2(1),T),
timepoint(T).

%============ Pick-up timer Circuito F28_3 ============
initially(monoOutput_PU_CF28_3(0)).
happens(upTriger_PU_CF28_3,T3) :- holdsAt(inputIs_PU_CF28_3(0),T1),
holdsAt(inputIs_PU_CF28_3(1),T2), T2=T1+1, T3=T1+2, timepoint(T1),
timepoint(T2), timepoint(T3).
terminates(upTriger_PU_CF28_3,monoOutput_PU_CF28_3(0),T) :-
happens(upTriger_PU_CF28_3,T), holdsAt(inputIs_PU_CF28_3(1),T), timepoint(T).
 initiates(upTriger_PU_CF28_3,monoOutput_PU_CF28_3(1),T)  :-
happens(upTriger_PU_CF28_3,T), holdsAt(inputIs_PU_CF28_3(1),T), timepoint(T).
```

```
happens(downTriger_PU_CF28_3,0) :- initially(inputIs_PU_CF28_3(0)).
happens(downTriger_PU_CF28_3,T1) :- holdsAt(inputIs_PU_CF28_3(1),T1),
holdsAt(inputIs_PU_CF28_3(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(downTriger_PU_CF28_3,monoOutput_PU_CF28_3(1),T) :-
happens(downTriger_PU_CF28_3,T), holdsAt(monoOutput_PU_CF28_3(1),T),
timepoint(T).
 initiates(downTriger_PU_CF28_3,monoOutput_PU_CF28_3(0),T)  :-
happens(downTriger_PU_CF28_3,T), holdsAt(monoOutput_PU_CF28_3(1),T),
timepoint(T).


% === SAÍDAS Circuito F_28 ===
% Partes comuns a algumas saídas
holdsAt(aB_28(AB),T) :- and(N,O,AB), holdsAt(vI13_17(N),T),
holdsAt(vI17_17(O),T), nivelLogico(AB), timepoint(T).
holdsAt(bA_28(BA),T) :- and4(AA,D,E,F,BA), or3(A,B,C,AA),
holdsAt(vI45_27(A),T), holdsAt(vI50_27(B),T), holdsAt(vI51_27(C),T),
holdsAt(dI3L(D),T), holdsAt(vI21_18(E),T), holdsAt(vI02_15(F),T), nivelLogico(BA),
nivelLogico(AA), timepoint(T).
holdsAt(bB_28(BB),T) :- or(M,AB,BB), holdsAt(vI05_15(M),T),
holdsAt(aB_28(AB),T), nivelLogico(BB), timepoint(T).
holdsAt(cA_28(CA),T) :- and3(BA,G,H,CA), holdsAt(bA_28(BA),T),
holdsAt(dI1L(G),T), holdsAt(vI14_17(H),T), nivelLogico(CA), timepoint(T).
holdsAt(inputIs_PU_CF28_1(CA),T) :- holdsAt(cA_28(CA),T), timepoint(T).
holdsAt(dA_28(DA),T) :- holdsAt(monoOutput_PU_CF28_1(DA),T), timepoint(T).
holdsAt(inputIs_PU_CF28_2(CB),T) :- and4(NCA,BA,I,J,CB), inv(CA,NCA),
holdsAt(cA_28(CA),T), holdsAt(bA_28(BA),T), holdsAt(dI2L(I),T),
holdsAt(vI18_17(J),T), nivelLogico(CB), nivelLogico(NCA), timepoint(T).
holdsAt(dB_28(DB),T) :- holdsAt(monoOutput_PU_CF28_2(DB),T), timepoint(T).
holdsAt(eA_28(EA),T) :- or(DA,DB,EA), holdsAt(dA_28(DA),T),
holdsAt(dB_28(DB),T), nivelLogico(EA), timepoint(T).
holdsAt(fB_28(FB),T) :- or(EA,L,FB), holdsAt(eA_28(EA),T), holdsAt(vI64_28(L),T),
nivelLogico(FB), timepoint(T).
holdsAt(inputIs_PU_CF28_3(GB),T) :- and(E,NFB,GB), inv(FB,NFB),
holdsAt(vI21_18(E),T), holdsAt(fB_28(FB),T), nivelLogico(GB), nivelLogico(NFB),
timepoint(T).

% vI58_28 - TPAX (Enrolamento 1X) - Aberto
holdsAt(vI58_28(D),T) :- holdsAt(dI3L(D),T), timepoint(T).

% dO9E - Seleção de Tensão Barra 1
holdsAt(dO9E(DA),T) :- holdsAt(dA_28(DA),T), timepoint(T).

% dO10E - Seleção de Tensão Barra 2
holdsAt(dO10E(DB),T) :- holdsAt(dB_28(DB),T), timepoint(T).

% vI59_28 - Permissão de Fechamento Disjuntor 52AX Sincronismo Verificado ou
Disjuntor Isolado
```

```
holdsAt(vI59_28(HA),T) :- and(GA,E,HA), or(FA,L,GA), and(EA,K,FA),
holdsAt(eA_28(EA),T), holdsAt(vI65_34(K),T), holdsAt(vI64_28(L),T),
holdsAt(vI21_18(E),T), nivelLogico(HA), nivelLogico(GA), nivelLogico(FA),
timepoint(T).

% vI60_28 - Disjuntor 52AX - Seleção de Potencial Normal
holdsAt(vI60_28(HB),T) :- and(E,FB,HB), holdsAt(vI21_18(E),T),
holdsAt(fB_28(FB),T), nivelLogico(HB), timepoint(T).

% vI61_28 - Disjuntor 52AX Seleção de Potencial Falha - Alarme
holdsAt(vI61_28(HC),T) :- holdsAt(monoOutput_PU_CF28_3(HC),T), timepoint(T).

% vI62_28 - Seção AX Aberta
holdsAt(vI62_28(CC),T) :- or(E,BB,CC), holdsAt(vI21_18(E),T),
holdsAt(bB_28(BB),T), nivelLogico(CC), timepoint(T).

% vI63_28 - Disjuntor 52AX Pronto para Manobra
holdsAt(vI63_28(CD),T) :- inv(BB,CD), holdsAt(bB_28(BB),T), nivelLogico(CD),
timepoint(T).

% vI64_28 - Disjuntor 52AX Isolado
holdsAt(vI64_28(BC),T) :- and(M,AB,BC), holdsAt(vI05_15(M),T),
holdsAt(aB_28(AB),T), nivelLogico(BC), timepoint(T).

%= FIM - Figura 25 - Coqueiros Simplificado F_28  ===



% === Figura 26 - Coqueiros Simplificado F_29 ===

% === Entradas Circuito F_29 ===
% eP01_32   (Seleção Local/Remoto-Remoto - Declarada no Circuito F_19)
% vI39_24   (Cartão Amarelo colocado Disjuntor 52AX - Saída Circuito F_24)
% vI32_88   (Ativar Religamento Monopolar Disjuntor 52AX - Saída Circuito F_88)
% vI33_88   (Ativar Religamento Tripolar Disjuntor 52AX - Saída Circuito F_88)
% vI34_88   (Religamento Disjuntor 52AX Desligado - Saída Circuito F_88)


% === (IHM-UCD1) Seleção Monopolar Disjuntor 52AX
initially(eP19_32(0)).

 terminates(eP19_32up,eP19_32(0),T) :- timepoint(T).
   initiates(eP19_32up,eP19_32(1),T) :- timepoint(T).
 terminates(eP19_32dw,eP19_32(1),T) :- timepoint(T).
   initiates(eP19_32dw,eP19_32(0),T) :- timepoint(T).

happens(eP19_32dw,1).
```

```
% === (IHM-UCD1) Seleção Religamento Tripolar Disjuntor 52AX
initially(eP20_32(0)).

 terminates(eP20_32up,eP20_32(0),T) :- timepoint(T).
   initiates(eP20_32up,eP20_32(1),T) :- timepoint(T).
 terminates(eP20_32dw,eP20_32(1),T) :- timepoint(T).
   initiates(eP20_32dw,eP20_32(0),T) :- timepoint(T).

happens(eP20_32dw,1).


% === (IHM-UCD1) Religamento Disjuntor 52AX Desligado
initially(eP21_32(0)).

 terminates(eP21_32up,eP21_32(0),T) :- timepoint(T).
   initiates(eP21_32up,eP21_32(1),T) :- timepoint(T).
 terminates(eP21_32dw,eP21_32(1),T) :- timepoint(T).
   initiates(eP21_32dw,eP21_32(0),T) :- timepoint(T).

happens(eP21_32dw,1).


%============ Flip-flop SR Circuito F29_1 ============
initially(qout_CF29_1(0)).
happens(i2_CF29_1_sup,T1) :- holdsAt(sin_CF29_1(0),T1),
holdsAt(sin_CF29_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_1_sdw,T1) :- holdsAt(sin_CF29_1(1),T1),
holdsAt(sin_CF29_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_1_rup,T1) :- holdsAt(rin_CF29_1(0),T1),
holdsAt(rin_CF29_1(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_1_rdw,T1) :- holdsAt(rin_CF29_1(1),T1),
holdsAt(rin_CF29_1(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF29_1_sup,qout_CF29_1(0),T) :- holdsAt(rin_CF29_1(0),T),
holdsAt(qout_CF29_1(0),T), timepoint(T).
 initiates(i2_CF29_1_sup,qout_CF29_1(1),T) :- holdsAt(rin_CF29_1(0),T),
holdsAt(qout_CF29_1(0),T), timepoint(T).
terminates(i2_CF29_1_rup,qout_CF29_1(1),T) :- holdsAt(sin_CF29_1(0),T),
holdsAt(qout_CF29_1(1),T), timepoint(T).
 initiates(i2_CF29_1_rup,qout_CF29_1(0),T) :- holdsAt(sin_CF29_1(0),T),
holdsAt(qout_CF29_1(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF29_1_rup,qout_CF29_1(1),T) :- holdsAt(sin_CF29_1(1),T),
holdsAt(qout_CF29_1(1),T), timepoint(T).
 initiates(i2_CF29_1_rup,qout_CF29_1(0),T) :- holdsAt(sin_CF29_1(1),T),
holdsAt(qout_CF29_1(1),T), timepoint(T).

%============ Flip-flop SR Circuito F29_2 ============
initially(qout_CF29_2(0)).
```

```
happens(i2_CF29_2_sup,T1) :- holdsAt(sin_CF29_2(0),T1),
holdsAt(sin_CF29_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_2_sdw,T1) :- holdsAt(sin_CF29_2(1),T1),
holdsAt(sin_CF29_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_2_rup,T1) :- holdsAt(rin_CF29_2(0),T1),
holdsAt(rin_CF29_2(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_2_rdw,T1) :- holdsAt(rin_CF29_2(1),T1),
holdsAt(rin_CF29_2(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF29_2_sup,qout_CF29_2(0),T) :- holdsAt(rin_CF29_2(0),T),
holdsAt(qout_CF29_2(0),T), timepoint(T).
 initiates(i2_CF29_2_sup,qout_CF29_2(1),T) :- holdsAt(rin_CF29_2(0),T),
holdsAt(qout_CF29_2(0),T), timepoint(T).
terminates(i2_CF29_2_rup,qout_CF29_2(1),T) :- holdsAt(sin_CF29_2(0),T),
holdsAt(qout_CF29_2(1),T), timepoint(T).
 initiates(i2_CF29_2_rup,qout_CF29_2(0),T) :- holdsAt(sin_CF29_2(0),T),
holdsAt(qout_CF29_2(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF29_2_rup,qout_CF29_2(1),T) :- holdsAt(sin_CF29_2(1),T),
holdsAt(qout_CF29_2(1),T), timepoint(T).
 initiates(i2_CF29_2_rup,qout_CF29_2(0),T) :- holdsAt(sin_CF29_2(1),T),
holdsAt(qout_CF29_2(1),T), timepoint(T).

%============ Flip-flop SR Circuito F29_3 ============
initially(qout_CF29_3(0)).
happens(i2_CF29_3_sup,T1) :- holdsAt(sin_CF29_3(0),T1),
holdsAt(sin_CF29_3(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_3_sdw,T1) :- holdsAt(sin_CF29_3(1),T1),
holdsAt(sin_CF29_3(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_3_rup,T1) :- holdsAt(rin_CF29_3(0),T1),
holdsAt(rin_CF29_3(1),T2), T2=T1+1, timepoint(T1), timepoint(T2).
happens(i2_CF29_3_rdw,T1) :- holdsAt(rin_CF29_3(1),T1),
holdsAt(rin_CF29_3(0),T2), T2=T1+1, timepoint(T1), timepoint(T2).
terminates(i2_CF29_3_sup,qout_CF29_3(0),T) :- holdsAt(rin_CF29_3(0),T),
holdsAt(qout_CF29_3(0),T), timepoint(T).
 initiates(i2_CF29_3_sup,qout_CF29_3(1),T) :- holdsAt(rin_CF29_3(0),T),
holdsAt(qout_CF29_3(0),T), timepoint(T).
terminates(i2_CF29_3_rup,qout_CF29_3(1),T) :- holdsAt(sin_CF29_3(0),T),
holdsAt(qout_CF29_3(1),T), timepoint(T).
 initiates(i2_CF29_3_rup,qout_CF29_3(0),T) :- holdsAt(sin_CF29_3(0),T),
holdsAt(qout_CF29_3(1),T), timepoint(T).
% OBS.: Adicionado para compatibilizar com a lógica do esquema da AREVA
terminates(i2_CF29_3_rup,qout_CF29_3(1),T) :- holdsAt(sin_CF29_3(1),T),
holdsAt(qout_CF29_3(1),T), timepoint(T).
 initiates(i2_CF29_3_rup,qout_CF29_3(0),T) :- holdsAt(sin_CF29_3(1),T),
holdsAt(qout_CF29_3(1),T), timepoint(T).


% === SAÍDAS Circuito F_29 ===
% Partes comuns a algumas saídas
```

```
holdsAt(bA_29(BA),T) :- or(AA,AB,BA), and3(NA,NB,C,AA), and3(A,NB,D,AB),
inv(A,NA), inv(B,NB), holdsAt(eP01_32(A),T), holdsAt(vI39_24(B),T),
holdsAt(eP19_32(C),T), holdsAt(vI32_88(C),T), nivelLogico(BA), nivelLogico(AA),
nivelLogico(AB), nivelLogico(NA), nivelLogico(NB), timepoint(T).
holdsAt(bB_29(BB),T) :- or(AC,AD,BB), and3(NA,NB,E,AC), and3(A,NB,F,AD),
inv(A,NA), inv(B,NB), holdsAt(eP01_32(A),T), holdsAt(vI39_24(B),T),
holdsAt(eP20_32(E),T), holdsAt(vI33_88(F),T), nivelLogico(BB), nivelLogico(AC),
nivelLogico(AD), nivelLogico(NA), nivelLogico(NB), timepoint(T).
holdsAt(bC_29(BC),T) :- or(AE,AF,BC), and(NA,G,AE), and(A,H,AF), inv(A,NA),
holdsAt(eP01_32(A),T), holdsAt(eP21_32(G),T), holdsAt(vI34_88(H),T),
nivelLogico(BC), nivelLogico(AE), nivelLogico(AF), nivelLogico(NA), timepoint(T).
holdsAt(sin_CF29_1(BA),T) :- holdsAt(bA_29(BA),T), timepoint(T).
holdsAt(rin_CF29_1(CA),T) :- or(BB,BC,CA), holdsAt(bB_29(BB),T),
holdsAt(bC_29(BC),T), nivelLogico(CA), timepoint(T).
holdsAt(sin_CF29_2(BB),T) :- holdsAt(bB_29(BB),T), timepoint(T).
holdsAt(rin_CF29_2(CB),T) :- or(BA,BC,CA), holdsAt(bA_29(BA),T),
holdsAt(bC_29(BC),T), nivelLogico(CB), timepoint(T).
holdsAt(sin_CF29_3(BC),T) :- holdsAt(bC_29(BC),T), timepoint(T).
holdsAt(rin_CF29_3(CC),T) :- or(BA,BB,CC), holdsAt(bA_29(BA),T),
holdsAt(bB_29(BB),T), nivelLogico(CC), timepoint(T).
holdsAt(dA_29(DA),T) :- holdsAt(qout_CF29_1(DA),T), timepoint(T).
holdsAt(dB_29(DB),T) :- holdsAt(qout_CF29_2(DB),T), timepoint(T).
holdsAt(dC_29(DC),T) :- holdsAt(qout_CF29_3(DC),T), timepoint(T).

% vI66_29 - Seleção Religamento Monopolar
holdsAt(vI66_29(DA),T) :- holdsAt(dA_29(DA),T), timepoint(T).

% vI67_29 - Seleção Religamento Tripolar
holdsAt(vI67_29(DB),T) :- holdsAt(dB_29(DB),T), timepoint(T).

% vI68_29 - Religamento Disjuntor 52AX Ligado
holdsAt(vI68_29(EA),T) :- or(DB,DA,EA), holdsAt(dB_29(DB),T),
holdsAt(dA_29(DA),T), nivelLogico(EA), timepoint(T).

% vI69_29 - Religamento Disjuntor 52AX Desligado
holdsAt(vI69_29(DC),T) :- holdsAt(dC_29(DC),T), timepoint(T).

%= FIM - Figura 26 - Coqueiros Simplificado F_29  ===



%=======================================================

#hide.
#show holdsAt/2.
#show happens/2.
```

Anexo B

Timelime dos Eventos do Modelo

Tabela 5 - Timeline da Falha da Abertura de disjuntor em Barra dos “Coqueiros Simplificada”

Tempo	Evento	Descrição
00s	dI1P(1) e dI2P(0)	Secionadora_57AX_Aberta
	dI3P(0) e dI4P(1)	Secionadora_89AX2_Fechada
	dI5P(1) e dI6P(0)	Secionadora_89AX4_Aberta
	dI9P(1) e dI10P(0)	Secionadora_89AX3_Aberta
	dI7P(0) e dI8P(1)	Secionadora_89AX1_Fechada
	dI11P(0) e dI12P(1)	Disjuntor_52AX_Fechado
	eP01_90(1) = vI01_80(1)	Seleção de controle Local/Remoto - Setado Local
	eP02_90(0), iECA(0),	
	eP03_90(0), iECB(0),	
	eP04_90(0), iECC(0),	
	eP05_90(0), iECD(0)	Sem comandos para Secionadora 89AX1
	eP06_90(1), iECE(0),	
	eP07_90(0), iECF(0),	
	eP08_90(0), iECG(0),	
	eP09_90(0), iECH(0)	Sem comandos para Secionadora 89AX3
	eP10_90(1), iECI(0),	
	eP11_90(0), iECJ(0),	
	eP12_90(0), iECK(0),	
	eP13_90(0), iECL(0)	Sem comandos para Secionadora 89AX2
	eP14_90(1), iECM(0),	
	eP15_90(0), iECN(0),	
	eP16_90(0), iECO(0),	
	eP17_90(0), iECP(0)	Sem comandos para Secionadora 57AX
	eP18_90(1), iECQ(0),	
	eP19_90(0), iECR(0),	
	eP20_90(0), iECS(0),	
	eP21_90(0), iECT(0)	Sem comandos para Secionadora 89AX4
	eP22_90(0), iECU(0),	
	eP23_90(0), iECV(0),	
	eP24_90(0), iECW(0),	
	eP25_90(0), iECX(0),	
	eP26_90(0), iECY(0),	
	eP27_90(0), iECZ(0)	Sem comandos para Disjuntor 52AX
	eP28_90(0), iECAA(0)	Disjuntor 52AX Desbloqueado
	ePN_90(0), iECAB(1),	
	ePET_90(0), iECAC(0),	
	ePT_90(0), iECAD(0)	Seleção Transferência em Proteção Normal "N"

		(setado pelo Local)
	eP32_90(0), iECAE(0),	
	eP33_90(0), iECAF(1),	
	eP34_90(0), iECAG(0)	Religamento Tripolar Disjuntor 52AX Selecionado
	dDB691(0), dDB695(1),	
	dDB99(0), oPTO12(1)	TDD Mantido, Sem Ocorrer Proteção de Falha Disjuntor 52AX
	eP01_32(1)	Seleção de controle Local/Remoto - Setado Local
	eP02_32(0)	Sem comando de Fechamento remoto Secionadora 89AX1
	eP03_32(0)	Sem cartões inseridos
	dI14O(1)	Secionadora 89AX1 Chave 43 L/R-Remoto
	eP04_32(0)	Sem comando de Abertura remota Secionadora 89AX1
	dI12N_405(0)	Sem indicação de baixão pressão SF6 2o Estágio Disj. 52AX
	eP05_32(0)	Sem comando de Fechamento remoto Secionadora 89AX3
	dI2N(1)	Secionadora 89AX3 Chave 43 L/R-Remoto
	eP06_32(0)	Sem comando de Abertura remota Secionadora 89AX3
	eP07_32(0)	Sem comando de Fechamento remoto Secionadora 89AX2
	dI2O(1)	Secionadora 89AX2 Chave 43 L/R-Remoto
	eP08_32(0)	Sem comando de Abertura remota Secionadora 89AX2
	dI9M(0)	Bloco de Teste não está em serviço
	vI70_34(0)	Sem tensão na LT Fases A, B, V
	eP11_32(0)	Sem comando de Fechamento remoto Secionadora 89AX4
	dI10O(1)	Secionadora 89AX4 Chave 43 L/R-Remoto
	eP12_32(0)	Sem comando de Abertura remota Secionadora 89AX4
	eP11_32(0)	Sem comando de Fechamento remoto Disjuntor 52AX
	vI14_405(0)	Mola Carregada Disjuntor 52AX
	dI7M_406(0)	Não há falta de VCC no circuitor de Fechamento Disjuntor 52AX
	dI13N_405(0)	Não há discordância de polos no Disjuntor 52AX
	dI14N_406(0)	Não falta VCC no Circuito de Comando Disjuntor 52AX
	dI17N_406(0)	Não há Falha Abertura 1
	dI18N_406(0)	Não há Falha Abertura 2
	dI15N(1)	Disjuntor 52AX Chave 43 L/R-Remoto
	eP14_32(0) 52AX	Sem comando de Abertura remota Secionadora
	dI4L(0)	Sem Falha Disjuntor 52AX
	dI14N_405(0)	Mola de Fechamento Carregada Disjuntor 52AX

	eP15_32dw(0)	Disjuntor 52AX Desbloqueado
	eP16_32dw(0)	Seleção Transferência Proteção Normal "N" (Não setado pelo Remoto)
	eP17_32dw(0)	Seleção Transferência Proteção Em Transferência "ET" (Não setado pelo Remoto)
	eP18_32dw(0)	Seleção Transferência Proteção Transferida "T" (Não setado pelo Remoto)
	vI33_26(0)	Nenhum Relé 43T atuado
	vI35_26(0)	Nenhuma das Secionadoras BY PASS Adjacentes Seção AX Abertas
	vI34_26(0)	Nenhuma das Secionadoras BARRA 2 Adjacentes Seção AX Abertas
	vI31_26(0)	Nenhuma Transferência de Proteção em Manual
	vI32_27(1)	Transferência de Proteção em Automático
	dI11L_409(0)	Transferência de Proteção Relé 43N - Não Atuado
	dI12L_409(0)	Transferência de Proteção Relé 43T - Não Atuado
	dI3L(1)	TPAX (Enrolamento 1X) - Fechado
	dI1L(1)	TPB1 (Enrolamento 1X) - Fechado
	dI2L(2)	TPB2 (Enrolamento 1X) - Fechado
	vI65_34(0)	Sincronismo Não Verificado UCD1 (P/ Fechamento Disjuntor 52AX)
	eP19_32(0)	Seleção Monopolar Disjuntor 52AX (Não setado pelo Remoto)
	eP20_32(0)	Seleção Tripolar Disjuntor 52AX (Não setado pelo Remoto)
	eP21_32(0)	Religamento Disjuntor 52AX Desligado (Não setado pelo Remoto)
01s		
02s		
03s		
04s		
05s		
06s	iECV(1)	SECOQ_CMD_AD52AX - CMD/PERM Abrir Disjuntor 52AX
07s	dI12P(0)	
08s	dI11P(1) iECV(0)	SECOQ_INFOR_AD52AX - Abertura DISJUNTOR 52AX
09s	iECB(1)	SECOQ_CMD_ASEC89AX1 - Abrir SECIONADORA 89AX1
10s		
11s	dI8P(0)	
12s	dI7P(1) iECB(0)	SECOQ_INFOR_ASEC89AX1 - Abertura SECIONADORA 89AX1
13s	iECE(1)	SECOQ_CMD_FSEC89AX3 - Fechar SECIONADORA 89AX3
14s	dI9P(0)	
15s	dI10P(1) iECE(0)	SECOQ_INFOR_FSEC89AX3 - Fechamento SECIONADORA 89AX3

	vI65_34(1)	Sincronismo Verificado UCD1
16s	iECU(1)	SECOQ_CMD_FD52AX - CMD/PERM Fechar Disjuntor 52AX
17s	dI11L_409(1)	Transferência de Proteção Relé 43N - Atuado
	dDB691(1) e	Operou Proteção Falha Disjuntor 52AX / PROTEÇÃO EFP
	oPTO12(1)	
	vI14_405(1)	Mola Descarregada Disjuntor 52AX
	dI7M_406(1)	Falta VCC no Circuito de Fechamento Disjuntor 52AX
	dI4L(1)	SECOQ_EMER_FD52AX - Falha DISJUNTOR 52AX
	dI14N_405(1)	Mola de Fechamento Descarregada Disjuntor 52AX
18s		
19s	iECJ(1) iECU(0)	SECOQ_CMD_ASEC89AX2 - Abrir SECIONADORA 89AX2
20s	dI4P(0)	
21s	dI3P(1) iECJ(0)	SECOQ_INFOR_ASEC89AX2 - Abertura SECIONADORA 89AX2
22s	iECF(1)	SECOQ_CMD_ASEC89AX3 - Abrir SECIONADORA 89AX3
23s	dI10P(0)	
24s	dI9P(1) iECF(0)	SECOQ_INFOR_ASEC89AX3 - Abertura SECIONADORA 89AX3

Anexo C

Exemplo de LOG de um Sistema SAGE

153245301700:MGPSR_LTI_85BLR-A:85A - Teleproteçãp Blocking
153245302900:SECOQ_CMD_SUPERV:CMD Supervisão
153245303300:SECOQ_SA_57AX:27P - Subtensão Linha A
153245305000:LTTES_DJ9056_MD:74 - Mola Descarregada DJ9056
153245308000:SECOQ_CMD_AD52AX:CMD/PERM Abertura Disjuntor 52AX
153245308200:SECOQ_INFO_57AX:Operação Normal
153245308400:CTEIG_LTMGP_T:Linha FRC sem Tensão
153245309200:LTTES_DJ9056_POS:Posição Disjuntor DJ9056
153245309900:SECOQ_INFOR_AD52AX:Abertura DISJUNTOR 52AX
153245310100:LTTES_LTJA-PCTE_PROTP:PCTE - Atuação Prot Principal LT Jaguara
153245310500:SECOQ_INFO_57AX:EM PRONTIDÃO
153245311300:PTEVP_SC8023_FTM:Falta Tensão no Motor
153245311400:SECOQ_CMD_ASEC89AX1:Abrir SECIONADORA 89AX1
153245311600:LTTS_DJ9046_94P:94P - Relé de DISPARO DJ9046 Atuado
153245313400:SECOQ_EMER_BA:SubTensão Barramento A
153245314400:SECOQ_INFOR_ASEC89AX1:Abertura SECIONADORA 89AX1
153245314900:SECOQ_INFO_PRAX:EM PRONTIDÃO
153245315700:SECOQ_INFO_BA:Tensão Barramento A normal
153245315700:LTTES_SA_QD1_27B:27 - Subtenção Barramento QD1
153245315700:LTTNP_LTES_50EN-A:50ENA - Sobrecorrente Emergência de Neutro
153245315700:SECOQ_INFO_BA:Sobretensão Barramento A
153245316000:LTTSG_SA_QD2_27B:27 - Subtensão Barramento QD2
153245316300:SECOQ_CMD_FSEC89AX3:Fechar SECIONADORA 89AX3
153245317800:PTEVP_SC6123_FTM:Falta Tensão no motor
153245318800:PTEVP_SA_QS2_27B:27 Subtensão Barramento QS2
153245319200:SECOQ_INFOR_FSEC89AX3:Fechamento SECIONADORA 89AX3
153245319500:LTTES_LTJA-PCTE-PROTA:PCTE - Atuação Prot Alternada LT Jaguara
153245320600:LTTES_LTES_50EN-P:50ENP - Sobrecorrente Emergência de neutro
153245321200:LTTES_8BB_8UCXT2_KVAB:Tensão Barra B - 8UCXT2
153245321400:SECOQ_CMD_FD52AX:CMD/PERM Fechar DISJUNTOR 52AX
153245321500:SECOQ_INFOR_BB:Tensão Barramento B normal
153245321700:LTTIM_LTNP_67R-A:67RA - Sobrecorrente Reversa
153245322000:LTTIM_DJ9756_FTM:27 Falta Tensão Motor do DJ9756
153245322300:LTTIM_LYNP_60_P:60P - Falha Fusível
153245322800:SECOQ_EMER_FD52AX:Falha DISJUNTOR 52AX
153245324900:VCTVC_AGROP_RET_BD:Retificador Bateria em Descarga
153245325000:LTCAN_7RTCO2_71LBT:71 Nível Baixo Óleo Reator
153245325100:SECOQ_CMD_ASEC89AX2:Abrir SECIONADORA 89AX2
153245325900:CTESS_DJ17U4_FTM:27 - Falta Tensão Motor
153245326500:CTEIG_9AT01_1VFG1FA:Ventilação Forçada Grupo 1 Fase A

153245327600:ETIG_9AT01_VFG1FC:Ventilação Forçada Grupo 1 Fase C
153245328200:SECOQ_INFOR_ASEC89AX2:Abertura SECIONADORA 89AX2
153245329800:LTTES_9BA_UCX_KVAB:Tensão Barra A - UCX
153245330200:SECOQ_CMD_ASEC89AX3:Abrir SECIONADORA 89AX3
153245331200:CC1SR_7TR02_VF_LIG:Ventilação Forçada 7TR02 Ligada
153245332200:SECOQ_INFOR_ASEC89AX3:Abertura SECIONADORA 89AX3
153245336600:SECOQ_INFO_BB:Tensão Barramento B Normal
153245338900:LTTES_SA_QD3_27B:27 Subtensão Barramento QD3
153245339000:null:null

Anexo D

Programa Java para teste de Reconhecimento de Crônicas (MainFrame.java)

```
// ---------------------------------------------------------------------
// MainFrame.java
//
// Programa Java para teste de Reconhecimento de Crônicas
//
// Autores: Jorge Lopes de Souza Leão <leao@lemt.ufrj.br> e
//          Rafael Jorge Csura Szendrodi <szendro@lemt.ufrj.br>
//
// Última Modificação: 12/11/2013
//
package cronicas01;

import java.io.BufferedReader;
import java.io.FileReader;
import java.text.DecimalFormat;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.GregorianCalendar;
import java.util.concurrent.CopyOnWriteArrayList;

public class MainFrame extends javax.swing.JFrame {
    public static MainFrame mainFrame;
    CopyOnWriteArrayList<Cronica> listaDeInstancias;
    CopyOnWriteArrayList<Cronica> listaDeModelos;
    CopyOnWriteArrayList<Cronica> cronicasReconhecidas;

//---------------------------------------------------------------------
----------
    public MainFrame() {
        initComponents();
        listaDeInstancias = new CopyOnWriteArrayList<>();
        listaDeModelos = new CopyOnWriteArrayList<>();
        cronicasReconhecidas = new CopyOnWriteArrayList<>();
//        (new ThreadDeSaida()).start();
//        System.out.println("===> "+getCurrentTimeDate());
//        System.out.println("===>
"+convertMillisecondsToISO8601(1379895601123L));
//        System.out.println("===>
"+converterISO8601paraMilisegundos("2013-08-22 21:20:00.000"));
    }
//---------------------------------------------------------------------
----------
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {
```

```
        botaoTesteSimples = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        areaDeTexto = new javax.swing.JTextArea();
        botaoTesteLog = new javax.swing.JButton();
        botaoTesteFila = new javax.swing.JButton();
        botaoTesteVarias = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setMinimumSize(new java.awt.Dimension(1000, 600));
        getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());

        botaoTesteSimples.setFont(new java.awt.Font("Tahoma", 0, 18));
// NOI18N
        botaoTesteSimples.setText("TESTE SIMPLES");
        botaoTesteSimples.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                botaoTesteSimplesActionPerformed(evt);
            }
        });
        getContentPane().add(botaoTesteSimples, new
org.netbeans.lib.awtextra.AbsoluteConstraints(30, 30, 200, -1));

        areaDeTexto.setColumns(20);
        areaDeTexto.setRows(5);
        jScrollPane1.setViewportView(areaDeTexto);

        getContentPane().add(jScrollPane1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(30, 80, 840, 230));

        botaoTesteLog.setFont(new java.awt.Font("Tahoma", 0, 18)); //
NOI18N
        botaoTesteLog.setText("TESTE LOG");
        botaoTesteLog.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                botaoTesteLogActionPerformed(evt);
            }
        });
        getContentPane().add(botaoTesteLog, new
org.netbeans.lib.awtextra.AbsoluteConstraints(250, 30, 200, -1));

        botaoTesteFila.setFont(new java.awt.Font("Tahoma", 0, 18)); //
NOI18N
        botaoTesteFila.setText("TESTE FILA");
        botaoTesteFila.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                botaoTesteFilaActionPerformed(evt);
            }
        });
        getContentPane().add(botaoTesteFila, new
org.netbeans.lib.awtextra.AbsoluteConstraints(460, 30, 200, -1));
```

```
        botaoTesteVarias.setFont(new java.awt.Font("Tahoma", 0, 18));
// NOI18N
        botaoTesteVarias.setText("TESTE VÃ        RIAS");
        botaoTesteVarias.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                botaoTesteVariasActionPerformed(evt);
            }
        });
        getContentPane().add(botaoTesteVarias, new
org.netbeans.lib.awtextra.AbsoluteConstraints(670, 30, 200, -1));

        pack();
    }// </editor-fold>//GEN-END:initComponents
//-------------------------------------------------------------------
----------
    Cronica lerModeloDeArquivo(String filename){
        Cronica cronica = null;
        try{
            BufferedReader bf = new BufferedReader(new
FileReader("./"+filename+".cro"));
            cronica = new Cronica(filename,new
Integer(bf.readLine()).intValue());
            for(int nroEv=0;nroEv<cronica.nroEventos;nroEv++){
                cronica.ev[nroEv] = bf.readLine();
                cronica.min[nroEv] = new
Long(bf.readLine()).longValue();
                cronica.max[nroEv] = new
Long(bf.readLine()).longValue();
            }
        }catch(Exception e){e.printStackTrace();}
        return cronica;
    }
//-------------------------------------------------------------------
----------
    private void
botaoTesteSimplesActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_botaoTesteSimplesActionPerformed
        Cronica cronicaBase = lerModeloDeArquivo("cronica01");
        Cronica ccronica = cronicaBase.testarPrimeiroEvento("e95",
600);
        if(ccronica != null &&
           ccronica.testarEAtualizarTimestamp("e41", 800)==
                ResultadoTesteInstancia.RECONHECEU_EVENTO &&
           ccronica.testarEAtualizarTimestamp("e64", 1200)==
                ResultadoTesteInstancia.RECONHECEU_EVENTO     ){
         areaDeTexto.append("RECONHECEU OS EVENTOS");
        }else{
            areaDeTexto.append("NÃƒO RECONHECEU OS EVENTOS!");
        }
    }//GEN-LAST:event_botaoTesteSimplesActionPerformed
//-------------------------------------------------------------------
----------
    private void
botaoTesteLogActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_botaoTesteLogActionPerformed
        int posSeparador;
        String linha;
        String nomeDoEvento;
```

```
        String dateTimeDoEvento;
        try{
            BufferedReader br = new BufferedReader(new
FileReader("./logDeEventos10.log"));

            linha = br.readLine();
            posSeparador = linha.indexOf(':');
            nomeDoEvento = linha.substring(0, posSeparador);
            dateTimeDoEvento =
linha.substring(posSeparador+1,linha.length());
            Cronica cronicaBase = lerModeloDeArquivo("cronica01");
            Cronica ccronica = cronicaBase.testarPrimeiroEvento(
                    nomeDoEvento,
                    new Long(dateTimeDoEvento).longValue());
            System.out.println(
                    (ccronica!=null)+"      "+
                    linha           +"      "+
                    new Long(dateTimeDoEvento).longValue());

            ResultadoTesteInstancia result;
            while(br.ready()){
                linha = br.readLine();
                posSeparador = linha.indexOf(':');
                nomeDoEvento = linha.substring(0, posSeparador);
                dateTimeDoEvento =
linha.substring(posSeparador+1,linha.length());
//System.out.println(linha+" = "+nomeDoEvento+" + "+dateTimeDoEvento);
                result = ccronica.testarEAtualizarTimestamp(
                        nomeDoEvento,
                        new Long(dateTimeDoEvento).longValue());
                System.out.println(
                        result.toString()+"      "+
                        linha            +"      "+
                        new Long(dateTimeDoEvento).longValue());
            }
            br.close();
        }catch(Exception e){e.printStackTrace();}
    }//GEN-LAST:event_botaoTesteLogActionPerformed
//-------------------------------------------------------------------
----------
    private void
botaoTesteFilaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_botaoTesteFilaActionPerformed
        Cronica cronica1 = lerModeloDeArquivo("cronica1");
        listaDeInstancias.add(cronica1);

        Cronica cronica2 = lerModeloDeArquivo("cronica2");
        listaDeInstancias.add(cronica2);

        Cronica cronica3 = lerModeloDeArquivo("cronica3");
        listaDeInstancias.add(cronica3);

        Collections.sort(listaDeInstancias);
        System.out.println("=== Ordenadas:");
        for(Cronica c : listaDeInstancias){
            System.out.println(c.ev[0]+"  "+c.min[0]);
        }
        System.out.println("=== Ordenadas 2:");
        for(int i=0;i<3;i++){
            Cronica c = listaDeInstancias.get(i);
```

```
            System.out.println(c.ev[0]+"  "+c.min[0]);
        }
    }//GEN-LAST:event_botaoTesteFilaActionPerformed
//-------------------------------------------------------------------------
----------
    void imprimirLista(CopyOnWriteArrayList<Cronica> lista){
        System.out.print("=== Lista:");
        for(Cronica c : lista){
            System.out.print("  "+c.nome);
        }
        System.out.println(".");
    }
//-------------------------------------------------------------------------
----------
    private void
botaoTesteVariasActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_botaoTesteVariasActionPerformed
        int posSeparador;
        String linha;
        String dateTimeDoEvento;
        String nomeDoEvento;
        String descricaoDoEvento;
        try{
            //--- Ler modelos de cronicas
            Cronica modelo;

            listaDeModelos.add(lerModeloDeArquivo("cronica01"));
            listaDeModelos.add(lerModeloDeArquivo("cronica02"));
            listaDeModelos.add(lerModeloDeArquivo("cronica03"));

            //--- Ler Log de entrada e procurar identificar cronicas
            BufferedReader br = new BufferedReader(new
FileReader("./logDeEventos30.log"));
            int nroLinha = 0;
            while(br.ready()){
                linha = br.readLine();
                nroLinha++;
                //posSeparador = linha.indexOf(':');
                //dateTimeDoEvento = linha.substring(0, posSeparador);
                //nomeDoEvento =
linha.substring(posSeparador+1,linha.length());
                String[] tokens = linha.split(":");
                dateTimeDoEvento = tokens[0];
                nomeDoEvento = tokens[1];
                descricaoDoEvento = tokens[2];

                for(Cronica instancia : listaDeInstancias){
                    switch(instancia.testarEAtualizarTimestamp(
                                        nomeDoEvento,
                                        new
Long(dateTimeDoEvento).longValue())){
                        case NAO_SE_APLICA:{
//                            System.out.println("NAO_SE_APLICA
"+instancia.nome+" - "+instancia.lastTriedEvent);
                            break;
                        }
                        case MENOR_QUE_MINIMO:{
//                            System.out.println("MENOR_QUE_MINIMO
"+instancia.nome+" - "+instancia.lastTriedEvent);
                            break;
```

```
                        }
                        case RECONHECEU_EVENTO:{
//                            System.out.println("RECONHECEU_EVENTO
"+instancia.nome+" - "+instancia.lastTriedEvent);
                            break;
                        }
                        case MAIOR_QUE_MAXIMO:{
//                            System.out.println("MAIOR_QUE_MAXIMO
"+instancia.nome+" - "+instancia.lastTriedEvent);
                            // kill instance
                            areaDeTexto.append("\n Descartou
instÃ¢ncia de cronica: "+instancia.nome+
                                                "
Serial:"+instancia.serial+"   nroLinha: "+nroLinha);
                            listaDeInstancias.remove(instancia);
                            break;
                        }
                        case RECONHECEU_CRONICA:{
//                            System.out.println("RECONHECEU_CRONICA
"+instancia.nome+" - "+instancia.lastTriedEvent);
                            // retirar e colocar na fila produtor-
consumidor
//
cronicasReconhecidas.add(listaDeInstancias.get(listaDeInstancias.index
Of(instancia)));
                            areaDeTexto.append("\n Reconheceu
instÃ¢ncia de cronica: "+instancia.nome+
                                                "
Serial:"+instancia.serial+"   nroLinha: "+nroLinha);
                            listaDeInstancias.remove(instancia);
                            break;
                        }
                    }
                }

                Cronica instancia;
                for(Cronica mod : listaDeModelos){
                    instancia = mod.testarPrimeiroEvento(
                            nomeDoEvento,
                            new Long(dateTimeDoEvento).longValue());
                    if(instancia!=null){
                        listaDeInstancias.add(instancia);
                        areaDeTexto.append("\n Criou instÃ¢ncia de
cronica: "+instancia.nome+
                                                "
Serial:"+instancia.serial+"   nroLinha: "+nroLinha);
//                        System.out.println("CRIOU NOVA INSTANCIA
"+instancia.nome+" - "+instancia.lastTriedEvent);
                    }
                }

//                imprimirLista(listaDeInstancias);

            }
            imprimirLista(listaDeInstancias);
//            System.out.println("=== TERMINOU ===");
        }catch(Exception e){e.printStackTrace();}
    }//GEN-LAST:event_botaoTesteVariasActionPerformed
//-----------------------------------------------------------------
----------
```

```
    public static void main(String args[]) {
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.ht
ml
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java
.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                mainFrame = new MainFrame();
                mainFrame.setVisible(true);
            }
        });
    }
//------------------------------------------------------------------
----------
    long converterISO8601paraMilisegundos(String dateTime){
        long millis = new Long(dateTime.substring(20,23)).longValue();
        long timestamp = new GregorianCalendar(
                                    new
Integer(dateTime.substring(0,4)).intValue(),//4-traÃ§o
                                    new
Integer(dateTime.substring(5,7)).intValue(),//7-traÃ§o
                                    new
Integer(dateTime.substring(8,10)).intValue(),//10-traÃ§o
                                    new
Integer(dateTime.substring(11,13)).intValue(),//13-dois pontos
                                    new
Integer(dateTime.substring(14,16)).intValue(),//16-dois pontos
```

```
                                new
Integer(dateTime.substring(17,19)).intValue()//19-ponto decimal
                               ).getTimeInMillis();
        return (timestamp + millis);
    }
//----------------------------------------------------------------------
----------
    public String getCurrentTimeDate() {
        DecimalFormat df = new DecimalFormat("00");
        Calendar calendar = GregorianCalendar.getInstance();
        int hour = calendar.get(Calendar.AM_PM) == Calendar.AM ?
                                        calendar.get(Calendar.HOUR) :
                                        calendar.get(Calendar.HOUR) +
12;
        return (new Integer(calendar.get(Calendar.YEAR))).toString() +
                "-" + df.format(calendar.get(Calendar.MONTH) + 1) +
                "-" + df.format(calendar.get(Calendar.DATE)) +
                " " + df.format(hour) +
                ":" + df.format(calendar.get(Calendar.MINUTE)) +
                ":" + df.format(calendar.get(Calendar.SECOND));
    }
//----------------------------------------------------------------------
----------
    public String convertMillisecondsToISO8601(long milli) {
        DecimalFormat df = new DecimalFormat("00");
        DecimalFormat dfmilli = new DecimalFormat("000");
        Calendar calendar = GregorianCalendar.getInstance();
        calendar.setTimeInMillis(milli);
        int hour = calendar.get(Calendar.AM_PM) == Calendar.AM ?
                                        calendar.get(Calendar.HOUR) :
                                        calendar.get(Calendar.HOUR) +
12;
        String time = (new
Integer(calendar.get(Calendar.YEAR))).toString() +
                "-" + df.format(calendar.get(Calendar.MONTH)) +
                "-" + df.format(calendar.get(Calendar.DATE)) +
                " " + df.format(hour) +
                ":" + df.format(calendar.get(Calendar.MINUTE)) +
                ":" + df.format(calendar.get(Calendar.SECOND));
        long tlong = milli-
converterISO8601paraMilisegundos(time+".000");
        time = time + "." + dfmilli.format(tlong);
        return time;
    }
//----------------------------------------------------------------------
----------
    // Variables declaration - do not modify//GEN-BEGIN:variables
    javax.swing.JTextArea areaDeTexto;
    javax.swing.JButton botaoTesteFila;
    javax.swing.JButton botaoTesteLog;
    javax.swing.JButton botaoTesteSimples;
    javax.swing.JButton botaoTesteVarias;
    javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration//GEN-END:variables
//----------------------------------------------------------------------
----------
}
```

Anexo E

Programa Java para teste de Reconhecimento de Crônicas (Cronica.java)

```
// ---------------------------------------------------------------
// Cronica.java
//
// Programa Java para teste de Reconhecimento de Crônicas
// (Modulo Cronica.java)
//
// Autores: Jorge Lopes de Souza Leão <leao@lemt.ufrj.br> e
//          Rafael Jorge Csura Szendrodi <szendro@lemt.ufrj.br>
//
// Última Modficação: 12/11/2013
//
package cronicas01;

public class Cronica implements Cloneable, Comparable<Cronica>{
    String nome;
    static int serialBase;
    int serial;
    long eventTime;
    String[] ev;
    long[] timestamp;
    long[] min;
    long[] max;
    int nroEventos;
    int openEvent = 0;
    int lastAcceptedEvent = 0;
    String lastTriedEvent = "";

//-----------------------------------------------------------------
----------
    @Override
    public int compareTo(Cronica c) {
        long dif = this.min[openEvent]-c.min[openEvent];
        if(dif<0)return -1;
        else if(dif==0)return 0;
        else return 1;
    }
//-----------------------------------------------------------------
----------
    public Cronica(String nome, int nroEventos){
        this.serial = serialBase++;
        this.nome = nome;
        this.nroEventos = nroEventos;
        ev = new String[nroEventos];
        timestamp = new long[nroEventos];
        min = new long[nroEventos];
        max = new long[nroEventos];
    }
```

```
//-------------------------------------------------------------------
----------
    public void printCronica(){
        System.out.println("nroEventos: "+nroEventos);
        System.out.println("openEvent: "+openEvent);
        System.out.println("lastAcceptedEvent:"+lastAcceptedEvent);
        for(int i=0;i<nroEventos;i++)System.out.println("ev["+i+"]:
"+ev[i]);
        for(int
i=0;i<nroEventos;i++)System.out.println("timestamp["+i+"]:
"+timestamp[i]);
        for(int i=0;i<nroEventos;i++)System.out.println("min["+i+"]:
"+min[i]);
        for(int i=0;i<nroEventos;i++)System.out.println("max["+i+"]:
"+max[i]);
    }
//-------------------------------------------------------------------
----------
    @Override
    public Cronica clone(){
        Cronica cronica = new Cronica(nome, nroEventos);

        cronica.ev = new String[nroEventos];
        System.arraycopy(this.ev, 0, cronica.ev, 0, nroEventos);

        cronica.timestamp = new long[nroEventos];
        System.arraycopy(this.timestamp, 0, cronica.timestamp, 0,
nroEventos);

        cronica.min = new long[nroEventos];
        System.arraycopy(this.min, 0, cronica.min, 0, nroEventos);

        cronica.max = new long[nroEventos];
        System.arraycopy(this.max, 0, cronica.max, 0, nroEventos);

        cronica.timestamp[0] = this.eventTime;
        cronica.min[0] += this.eventTime;
        cronica.max[0] += this.eventTime;
        cronica.openEvent = 1;
        cronica.lastAcceptedEvent = 0;
        cronica.lastTriedEvent = this.lastTriedEvent;
//cronica.printCronica();
        return cronica;
    }
//-------------------------------------------------------------------
----------
    public Cronica testarPrimeiroEvento(String event, long eventTime){
        this.lastTriedEvent = event;
        if(event.equals(ev[0])){
            this.eventTime = eventTime;
            return this.clone();
        }else{
            return null;
        }
    }
//-------------------------------------------------------------------
----------
    public ResultadoTesteInstancia testarEAtualizarTimestamp(String
event, long eventTime){
  //this.printCronica();
```

```
        this.lastTriedEvent = event;
        if(event.equals(ev[openEvent])){
            if( (eventTime>=min[lastAcceptedEvent]) &&
                (eventTime<=max[lastAcceptedEvent])     ){

                timestamp[openEvent] = eventTime;
                min[openEvent] += eventTime;
                max[openEvent] += eventTime;
                lastAcceptedEvent = openEvent;
                openEvent++;
                if(openEvent==nroEventos){
                    return ResultadoTesteInstancia.RECONHECEU_CRONICA;
                }
                return ResultadoTesteInstancia.RECONHECEU_EVENTO;
            }else if(eventTime>max[lastAcceptedEvent]){
                //kill instance
                return ResultadoTesteInstancia.MAIOR_QUE_MAXIMO;
            }else{
                return ResultadoTesteInstancia.MENOR_QUE_MINIMO;
            }
        }else{
            // event not recognized by this chronicle instance...
            return ResultadoTesteInstancia.NAO_SE_APLICA;
        }
    }
//-------------------------------------------------------------------------
----------
}
```

Anexo F

Crônicas (Cronica01.cro, Cronica02.cro e Cronica03.cro) para uso no programa de teste de Reconhecimento de Crônicas

// Formato dos arquivos:
// 1ª Linha: Número de Eventos
// Demais linhas (cada 3 linhas): Evento
// Limite Mínimo
// Limite Máximo
// Obs.: Último evento da crônica deve ter os tempos limites zerados!

Arquivo: Cronica01.cro

12
SECOQ_CMD_AD52AX
1700
2200
SECOQ_INFOR_AD52AX
800
1800
SECOQ_CMD_ASEC89AX1
1500
3200
SECOQ_INFOR_ASEC89AX1
600
2400
SECOQ_CMD_FSEC89AX3
1500
3200
SECOQ_INFOR_FSEC89AX3
600
2400
SECOQ_CMD_FD52AX
500
1500
SECOQ_EMER_FD52AX
1700
2400
SECOQ_CMD_ASEC89AX2
1500

3200
SECOQ_INFOR_ASEC89AX2
1700
2400
SECOQ_CMD_ASEC89AX3
1500
3200
SECOQ_INFOR_ASEC89AX3
0
0

Arquivo: Cronica02.cro

11
SECOQ_CMD_AD52AX
1700
2200
SECOQ_INFOR_AD52AX
800
1800
SECOQ_CMD_ASEC89AX1
1500
3200
SECOQ_INFOR_ASEC89AX1
600
2400
SECOQ_CMD_FSEC89AX3
1500
3200
SECOQ_INFOR_FSEC89AX3
600
2400
SECOQ_CMD_FD52AX
2200
3900
SECOQ_CMD_ASEC89AX2
1500
3200
SECOQ_INFOR_ASEC89AX2
1700
2400
SECOQ_CMD_ASEC89AX3
1500
3200
SECOQ_INFOR_ASEC89AX3
0
0

Arquivo: Cronica03.cro

9
SECOQ_CMD_AD52AX
1700
2200
SECOQ_INFOR_AD52AX
800
1800
SECOQ_EMER_FD52AX
1500
3200
SECOQ_CMD_ASEC89AX1
600
800
SECOQ_CMD_ASEC89AX2
1500
3200
SECOQ_INFOR_ASEC89AX1
600
2400
SECOQ_INFOR_ASEC89AX2
500
1500
SECOQ_CMD_FSEC89AX4
1500
3200
SECOQ_INFOR_FSEC89AX4
0
0

Anexo G

Histórico de Eventos (logDeEventos30.log) utilizado para uso no programa de teste de Reconhecimento de Crônicas

// Formato do arquivo:
// Todas as linhas do arquivo deve ser da forma:
// Timestamp: Evento:Descrição do Evento
// A ultima linha deve ter “Evento” e “Descrição do Evento” setadas como null

153245301700:MGPSR_LTI_85BLR-A:85A - Teleproteçãp Blocking
153245302900:SECOQ_CMD_SUPERV:CMD Supervisão
153245303300:SECOQ_SA_57AX:27P - Subtensão Linha A
153245305000:LTTES_DJ9056_MD:74 - Mola Descarregada DJ9056
153245308000:SECOQ_CMD_AD52AX:CMD/PERM Abertura Disjuntor 52AX
153245308200:SECOQ_INFO_57AX:Operação Normal
153245308400:CTEIG_LTMGP_T:Linha FRC sem Tensão
153245309200:LTTES_DJ9056_POS:Posição Disjuntor DJ9056
153245309900:SECOQ_INFOR_AD52AX:Abertura DISJUNTOR 52AX
153245310100:LTTES_LTJA-PCTE_PROTP:PCTE - Atuação Prot Principal LT Jaguara
153245310500:SECOQ_INFO_57AX:EM PRONTIDÃO
153245311300:PTEVP_SC8023_FTM:Falta Tensão no Motor
153245311400:SECOQ_CMD_SEC89AX1:Abrir SECIONADORA 89AX1
153245311600:LTTS_DJ9046_94P:94P - Relé de DISPARO DJ9046 Atuado
153245313400:SECOQ_EMER_BA:SubTensão Barramento A
153245314400:SECOQ_INFOR_ASEC89AX1:Abertura SECIONADORA 89AX1
153245314900:SECOQ_INFO_PRAX:EM PRONTIDÃO
153245315700:SECOQ_INFO_BA:Tensão Barramento A normal
153245315700:LTTES_SA_QD1_27B:27 - Subtenção Barramento QD1
153245315700:LTTNP_LTES_50EN-A:50ENA - Sobrecorrente Emergência de Neutro
153245315700:SECOQ_INFO_BA:Sobretensão Barramento A
153245316000:LTTSG_SA_QD2_27B:27 - Subtensão Barramento QD2
153245316300:SECOQ_CMD_FSEC89AX3:Fechar SECIONADORA 89AX3
153245317800:PTEVP_SC6123_FTM:Falta Tensão no motor
153245318800:PTEVP_SA_QS2_27B:27 Subtensão Barramento QS2
153245319200:SECOQ_INFOR_FSEC89AX3:Fechamento SECIONADORA 89AX3
153245319500:LTTES_LTJA-PCTE-PROTA:PCTE - Atuação Prot Alternada LT Jaguara
153245320600:LTTES_LTES_50EN-P:50ENP - Sobrecorrente Emergência de neutro
153245321200:LTTES_8BB_8UCXT2_KVAB:Tensão Barra B - 8UCXT2
153245321400:SECOQ_CMD_FD52AX:CMD/PERM Fechar DISJUNTOR 52AX

153245321500:SECOQ_INFOR_BB:Tensão Barramento B normal
153245321700:LTTIM_LTNP_67R-A:67RA - Sobrecorrente Reversa
153245322000:LTTIM_DJ9756_FTM:27 Falta Tensão Motor do DJ9756
153245322300:LTTIM_LYNP_60_P:60P - Falha Fusível
153245322800:SECOQ_EMER_FD52AX:Falha DISJUNTOR 52AX
153245324900:VCTVC_AGROP_RET_BD:Retificador Bateria em Descarga
153245325000:LTCAN_7RTCO2_71LBT:71 Nível Baixo Óleo Reator
153245325100:SECOQ_CMD_SEC89AX2:Abrir SECIONADORA 89AX2
153245325900:CTESS_DJ17U4_FTM:27 - Falta Tensão Motor
153245326500:CTEIG_9AT01_1VFG1FA:Ventilação Forçada Grupo 1 Fase A
153245327600:ETIG_9AT01_VFG1FC:Ventilação Forçada Grupo 1 Fase C
153245328200:SECOQ_INFOR_ASEC89AX2:Abertura SECIONADORA 89AX2
153245329800:LTTES_9BA_UCX_KVAB:Tensão Barra A - UCX
153245330200:SECOQ_CMD_SEC89AX3:Abrir SECIONADORA 89AX3
153245331200:CC1SR_7TR02_VF_LIG:Ventilação Forçada 7TR02 Ligada
153245332200:SECOQ_INFOR_ASEC89AX3:Abertura SECIONADORA 89AX3
153245336600:SECOQ_INFO_BB:Tensão Barramento B Normal
153245338900:LTTES_SA_QD3_27B:27 Subtensão Barramento QD3
153245339000:null:null

Anexo H

Exemplo de modularização (macro) em ASP/SEC

```
%
%  Autores: Jorge Lopes de Souza Leão <leao@lemt.ufrj.br> e
%               Rafael Jorge Csura Szendrodi <szendro@lemt.ufrj.br>
%================== Module declarations ==================
      module GeradorDeSinalSR{
         outputFluents: s, r;
         internalEvents: sup, sdw, rup, rdw;

         initially(s(0)).
         initially(r(0)).
         terminates(sup,s(0),T) :- timepoint(T).
          initiates(sup,s(1),T) :- timepoint(T).
             terminates(sdw,s(1),T) :- timepoint(T).
          initiates(sdw,s(0),T) :- timepoint(T).
             terminates(rup,r(0),T) :- timepoint(T).
          initiates(rup,r(1),T) :- timepoint(T).
             terminates(rdw,r(1),T) :- timepoint(T).
          initiates(rdw,r(0),T) :- timepoint(T).
         happens(sup,0).
         happens(sdw,50).
         happens(rup,100).
         happens(rdw,150).
         happens(sup,200).
         happens(sdw,250).
         happens(rup,300).
         happens(rdw,350).
      }

%=========================================================
      module FlipFlopSR{
```

```
        inputFluents: s, r;
        outputFluents: q;
        internalEvents: sup, sdw, rup, rdw;


        initially(q(0)).
        happens(sup,T1) :- holdsAt(s(0),T1), holdsAt(s(1),T2), T2=T1+1,
timepoint(T1), timepoint(T2).
        happens(sdw,T1) :- holdsAt(s(1),T1), holdsAt(s(0),T2), T2=T1+1,
timepoint(T1), timepoint(T2).
        happens(rup,T1) :- holdsAt(r(0),T1), holdsAt(r(1),T2), T2=T1+1, timepoint(T1),
timepoint(T2).
        happens(rdw,T1) :- holdsAt(r(1),T1), holdsAt(r(0),T2), T2=T1+1,
timepoint(T1), timepoint(T2).
        terminates(sup,q(0),T) :- holdsAt(r(0),T), holdsAt(q(0),T), timepoint(T).
         initiates(sup,q(1),T) :- holdsAt(r(0),T), holdsAt(q(0),T), timepoint(T).
        terminates(rup,q(1),T) :- holdsAt(s(0),T), holdsAt(q(1),T), timepoint(T).
         initiates(rup,q(0),T) :- holdsAt(s(0),T), holdsAt(q(1),T), timepoint(T).
    }

%================= Module instantiation ================


    gen01 = new GeradorDeSinalSR{
            sin <- s;
            rin <- r;
    }


    ff01 = new FlipFlopSR{
            s <- sin;
            r <- rin;
            qout <- q;
    }
```