VOLUME 35 / NUMBER 3 / 2007

CONTROL AND INTELLIGENT SYSTEMS

(Continued on Back Cover)

ISSN 1480-1752

Published by ACTA PRESS

Control and Intelligent Systems, Vol. 35, No. 3, 2007

DECISION SUPPORT SYSTEM WITH INCOMPLETE AND DOMAIN INCOHERENT INFORMATION MANAGEMENT

N. Malheiro,* Z. Vale,** C. Ramos,* M. Cordeiro,*** A. Marques,**** and V. Couto****

Abstract

Many of the most common human functions such as temporal and non-monotonic reasoning have not yet been fully mapped in developed systems, even though some theoretical breakthroughs have already been accomplished. This is mainly due to the inherent computational complexity of the theoretical approaches.

In the particular area of fault diagnosis in power systems however, some systems which tried to solve the problem, have been deployed using methodologies such as production rule based expert systems, neural networks, recognition of chronicles, fuzzy expert systems, etc.

SPARSE (from the Portuguese acronym, which means expert system for incident analysis and restoration support) was one of the developed systems and, in the sequence of its development, came the need to cope with incomplete and/or incorrect information as well as the traditional problems for power systems fault diagnosis based on SCADA (supervisory control and data acquisition) information retrieval, namely real-time operation, huge amounts of information, etc.

This paper presents an architecture for a decision support system, which can solve the presented problems, using a symbiosis of the event calculus and the default reasoning rule based system paradigms, insuring soft real-time operation with incomplete, incorrect or domain incoherent information handling ability. A prototype implementation of this system is already at work in the control centre of the Portuguese Transmission Network.

Key Words

Temporal reasoning, power systems, non-monotonic reasoning, diagnosis, event calculus, default logic, power systems

* Department of Informatics Engineering, Polytechnic Institute of Porto, R. Dr. António Bernardino de Almeida, 4200-072 Porto, Portugal; e-mail: {ntm, csr}@dei.isep.ipp.pt

** Department of Electrical Engineering, Polytechnic Institute of Porto, R. Dr. António Bernardino de Almeida, 4200-072 Porto, Portugal; e-mail: zav@dee.isep.ipp.pt

*** Engineering Section, University of Trás-os-Montes e Alto Douro, Engenharias II, 5000-911 Vila Real, Portugal; e-mail: cordeiro@utad.pt

**** REN – National Electrical Network, S.A. (EDP Group), Apartado 3-4471 Maia Codex; e-mail: {albinomarques, vieira-couto}@ren.pt

Recommended by Prof. Ruxu Du
(paper no. 201-1570)

1. Introduction

Modern power system control centres (CC) enable the operation and control of electrical networks using real-time information acquired through a complex SCADA (supervisory control and data acquisition) system. The SCADA system provides information about devices, alarms, measures, states, etc. in the electrical network and presents this information in the form of event lists, graphics and diagrams. The CC operators use this information to perform the actual real-time network operation and control.

CC operators are skilled professionals who receive the SCADA information, and, based on it, on their knowledge of electrical power systems and on their inherent temporal and non-monotonic reasoning abilities as human beings, assess the network's state. However, these operators have some human setbacks which are the difficulty of acting in extremely stressful situations and the inability to cope with very large amounts of information.

In a recent past, a new generation of computer applications started to be included in the CCs [1]. This new generation, based on artificial intelligence paradigms and techniques, aimed at complete decision-making support, promising full fault diagnosis and accurate restoration aid. These systems, implemented using techniques like expert systems (ES), artificial neural networks or fuzzy expert systems [2, 3], seemed adequate to solve the problems arising in the CCs. In fact, these applications could solve several of their predecessors' problems, but they were still unable to handle with both incompleteness and incoherence of information in the domain.

This paper presents an architecture, which tries to solve the presented problems, based on the knowledge (but not on the techniques) acquired during the development of SPARSE (from the Portuguese acronym, which means Expert System for Incident Analysis and Restoration Support) [4, 5]. This new architecture is based on a symbiotic approach using two paradigms, which are based on solid theoretical backgrounds: the Event Calculus (EC) and the Default Logic (DL). On one hand, such a system is required to take actions in the absence of some information or when incoherence is detected, being the DL paradigm appropriate to map these requirements. On the other hand,

temporal reasoning and non-monotonic behaviour are also important issues which are well dealt with using the EC. The architecture presented in this paper links these two techniques together, producing a knowledge-based intelligent fault diagnosis system, robust to both information incompleteness and domain incoherence.

2. Event Calculus

The EC formalism exists for some time now and was firstly introduced by Kowalsky and Sergot in 1986 [6]. It is basically some sort of logical machinery which, given the narrative of events or "what happens and when" along with our world's mechanics or "what actions do", can properly assess "what is true and when" [7]. This logical machinery is based on first-order predicate calculus, using some special predicates for the representation of events and their consequences in the temporal dimension. The EC comprises events (or actions), fluents and time points, being a fluent some entity, whose value changes its value over time.

The choice of the EC over the situation calculus was due to the time being explicitly represented in the former. As it is known, one formalism implies the other [8]. As pointed out in [9], the full EC can be computationally very heavy and a simpler variant, the simple event calculus (SEC), has proved to be more useful in practice, because it is of a polynomial order of computational complexity [10]. Nevertheless, the use of this variant loses a relevant feature which is the representation of the flow of time between events. In the SEC, reasoning is made only through the events, and hence the flow of time does not affect the reasoning. In SEC the statement "If 3 minutes have elapsed since event E, then ..." cannot be made. To maintain a simple complexity and represent the flow of time, a new variant of the SEC was created: the Simple Event Calculus with Timeouts (SECT).

2.1 Simple Event Calculus with Timeouts

In this paper, the SEC is extended, incorporating special events: the timeout events. The extended paradigm was named SECT. The SECT timeout events are added to the existing fact base whenever a new event arrives. The pair $(E,timeout(E))$, in which E is an event and $timeout(E)$ its corresponding timeout, captures the time window in which E is relevant for reasoning. This time window can be represented by a fluent, dependent from its event and corresponding timeout. The alarm correlation will be based on the intersection of these fluents, thus increasing chronology robustness. Additionally, the words "event" and "action" are used henceforth interchangeably in this paper (as well as in some of the literature), to represent an instantaneous event or action.

SECT has four axioms (SECT1 to 4), whose meaning will be clarified. SECT1 is used to prove that a certain property holds at a given time point if it is initially held and no event has interrupted it since then. SECT2 states that a certain property holds at a given time point if it has been held due to action A and has not been interrupted since that action occurred. The interruption includes both the termination due to another action or the timeout of the initiating action. SECT3 is a representation of the termination of a fluent due to a termination action, meaning that a fluent has (or has not) been clipped between time points *T1* and *T2*. A fluent is interrupted if some action existed, which terminated it. SECT4 represents the timeout of an initiating action. If one action A initiated a fluent and this fluent can be timed out by the initiating action A, the fluent will be terminated by the event $timeout(A)$, which is added to the knowledge base at the same time of the addition of A, to represent the limited temporal persistence of the action.

holdsAt(F,T) ← initiallyP(F) ∧ ¬clipped(0,F,T) (SECT1)

holdsAt(F,T2) ← happens(A,T1) ∧ initiates(A,F,T1) ∧ T1 < T2 ∧ ¬clipped(T1,F,T2) ∧ ¬timedout(T1,A,F,T2) (SECT2)

clipped(T1,F,T2) ↔ ∃ A,T [happens(A,T) ∧ T1 < T < T2 ∧ terminates(A,F,T)] (SECT3)

timedout(T1,A,F,T2) ↔ isTimedout(A,F) ∧ ∃T [happens(timeout(A),T) ∧ T1 < T < T2] (SECT4)

The ontology of the items in the SECT is:

- **happens(A,T)** means that action A happens at time point T;
- **initiallyP(F)** means that fluent F is valid (true) from the beginning of time;
- **holdsAt(F,T)** means that fluent F is valid at time point T;
- **initiates(A,F,T)** means that action A initiates the validity of fluent F at time point T;
- **terminates(A,F,T)** means that action A terminates the validity of fluent F at time point T;
- **clipped(T1,F,T2)** means that fluent F has become invalid (false) between time points *T1* and *T2*;
- **timeout(A)** represents the action added to the knowledge base together with action A, to represent its limited temporal persistence;
- **isTimedout(A,F)** means that action A is able to time out fluent F;
- **timedout(T1,A,F,T2)** means that action A times out fluent F between time points *T1* and *T2*.

3. Default Logic

The DL was initially introduced by Reiter [11] and is considered by some authors [12] to be one of the main approaches to non-monotonic default reasoning to be used in real applications.

DL theory is based on two sets, one of predicate logic formulae and another of default rules (or rules of thumb). The set of predicate logic formulae represents certain information about the world (W). Usually the known information is incomplete. The set of default rules (D) maps plausible assumptions, which aren't necessarily true. A default theory T is normally represented as a pair of the two described sets $T = (W,D)$.

A default rule is composed of three parts: the prerequisite, the justifications and the consequent. A default rule is graphically depicted as in *Default 1*, where φ is the prerequisite, $\psi_1, \ldots, \psi_n$ are the justifications and χ is the consequent of the default rule. Any default rule can be read as "if prerequisite φ is known (φ is a consequent of the current knowledge or assumptions), and it's consistent to assume justifications $\psi_1, \psi_2, \ldots, \psi_n$ (this consistency is based on the inability to prove the negation of each of the justifications by the predicate formulae joined with the consequents of other default rules previously assumed to be true) then the consequent χ can be concluded".

$$\frac{\varphi : \psi_1, \ldots, \psi_n}{\chi}$$

Default 1

The objective of the DL is not only to draw the fully derivable conclusions from the knowledge representation but also to expand these conclusions with plausible assumptions given by rules of thumb. Notice however that two (sets of) default rules can be inconsistent with each other. This will lead to two or more possible sets of assumable conclusions (usually called expansions). These expansions represent different conjectures based on the available knowledge and default rules. As an example, assume a theory $T = (\{animal(pat),\ bird(pat)\},\ \{Default\ 2,\ Default\ 3\})$, which presents *pat*. *pat* is both an animal and a bird. *Default 2* intends to present the default rule: "if X is an animal and it can be assumed that it does not fly, then we can assume that it really does not fly". In other words "animals do not usually fly". *Default 3* shows the default rule: "if X is a bird and it can be assumed that it flies, we can assume that it really flies". In other words, *Default 3* means "birds usually fly".

$$\frac{animal(X) : \neg flies(X)}{\neg flies(X)}$$

Default 2

$$\frac{bird(X) : flies(X)}{flies(X)}$$

Default 3

Notice that this default theory has two extensions, one including *{animal(pat), bird(pat), ¬flies(pat)}* and the other being *{animal(pat), bird(pat), flies(pat)}*. The application of one of the default rules inhibits the application of the other and therefore the initially considered rule prevails. The calculus of the complete set of extensions involves the application of all the permutations of the default rules. The calculus of the complete set of extensions has a formal definition, which can be found in [12]. The presentation of the intrinsic default reasoning is beyond the scope of this paper.

4. Architecture

The architecture proposed for decision support (see Fig. 1) consists of two interconnected subsystems, each specialized in its own reasoning domain. The use of these two modules, each of which encapsulating its own paradigm, is due to the need of commonsense reasoning over knowledge which include temporal information (for which default reasoning is used) and fast temporal reasoning requirements (for which the EC is used).

On one hand, we have the Event Calculus Temporal Expert (ECTE), whose main goals are to identify the temporal events arriving to the system and reason with them, elevating the abstraction level on the knowledge and filtering both information incompleteness and lack of chronology. The ECTE reasons based on fluents, which are entities that change their value over time. These fluents' initiation and termination are triggered by events which occur in defined states. The knowledge base is hierarchic in the level of abstraction. The increase in the abstraction level is performed by progressively giving access only to fluents that contain some event correlation of lower hierarchy. This component performs the same inference as the SPARSE original project but the change of paradigm improves the temporal and non-monotonic abilities of the production rule based ES. In the cases where information incompleteness or inaccuracy does not result

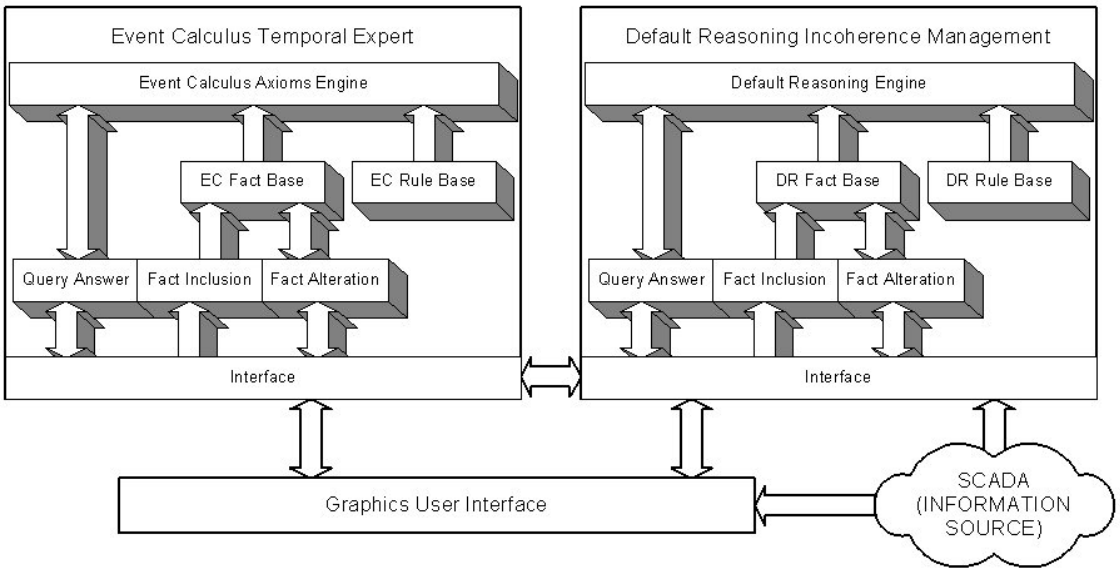


Figure 1. Architecture of the decision support system.

in information incoherence the ECTE will ensure correct decision support.

Besides the proactively delivery of inferred conclusions from the ECTE to the user, he can also access the ECTE and query it directly. In this architecture, this module performs temporal and non-monotonic alarm correlation. The main problem of the ECTE is that it does not have any semantic knowledge to handle information incoherence in the domain.

The Default Reasoning Incoherence Management (DRIM) acts as a component using the ECTE services. The DRIMs objective is to feed the ECTE with temporal events and intelligently query it to maintain knowledge of the network in a higher level of abstraction. This higher abstraction level is reached not basing the DRIMs premises directly on alarms, but on temporal ECTE fluents, which already contain some temporal reasoning.

The implementation of this architecture is done in a distributed environment, using a subset of KQML [13] as the means of communication between the intelligent components ECTE and DRIM.

4.1 Event Calculus Temporal Expert

This module is an implementation of the cached event calculus [14], changed to incorporate the SECT instead of SEC. Nevertheless some new knowledge representation issues have arisen. Due to this problem's specific requirement, the events have a limited time window of usefulness, or in other words, a limited temporal persistence. This was a problem because in the SEC only events can cause change and the simple flow of time doesn't cause anything to happen. This posed two options: either time was mapped as an event such as an alarm, which would go off at specified instants, or each event had to have a complementing event, which was its timeout. The later option has been chosen for efficiency and practicality in the rules representation of the fugitive meaning of each event. Each event is now transformed in two events and a fluent, which is true in the interval defined, by the event and its timeout. This is an extension of the EC presented in Section 2.1.

This module on its own can already answer every query needed for decision support, such as what state is a particular device in; if there is an incident; where the incident is located; identification of the incident type, etc. It has already proved by validation that this module can provide more reliable data than its predecessor SPARSE due to its robustness in the presence of chronological disorders and information incompleteness. A temporal flowchart of the ECTE can be found in Fig. 2.

4.2 Domain Incoherence

The DRIM has as a main function the criticism of the ECTE. In fact, it feeds the ECTE with events (this is merely done due to synchronization as both the ECTE and DRIM could be receiving data directly from the SCADA source) and then queries it using metaknowledge about the events it sent (only some specific queries will be demanded, opposed to the inference of all the possible in-

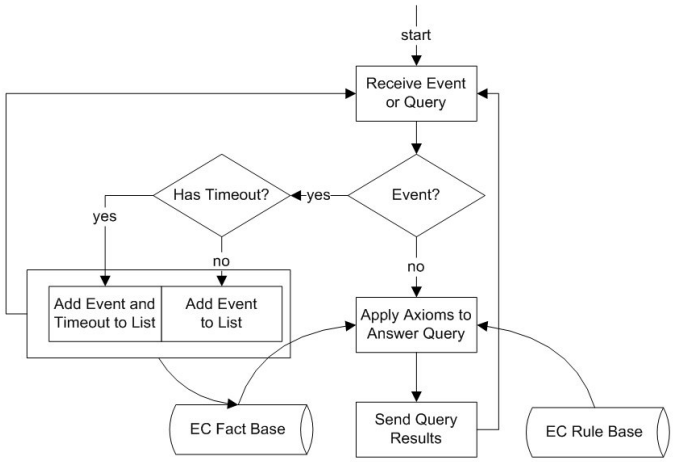


Figure 2. Temporal expert flowchart.

formation, thus limiting the number of applicable default rules). Criticisms come from internal representation of the domain and therefore allow the DRIM to conclude that, even though logical, the ECTEs conclusions can be domain incoherent. The DRIM can also add its own hypotheses (default events) to the list of events it sends to the ECTE. These default events can be added and removed to test "what-if" scenarios.

Some absolutely logic conclusions can be illogically given a domain. If, for instance, a rule says that premise A at time instant $T1$ can derive conclusion C at time instant $T1$, it seems logical that two events A at $T1$ and A at $T2$ will infer conclusion C at instants $T1$ and $T2$. Let us now additionally assume that nothing else was concluded between $T1$ and $T2$. It is logical that the conclusions still hold. But what-if semantics is included, giving premise A the meaning of an event, which informs of a device state change (*e.g.* the opening of a breaker) and giving conclusion C the meaning of internal representation of the device state change. In this case it would be assumed that C is true at $T1$ and $T2$ and that nothing happened in between, so the breaker opened at $T1$ and again changed from open to open at $T2$, which involves some kind of error. This kind of behaviour could be restrained (imposing restrictions on state change), but the information included in it is far more useful. Given that situation, the human reasoner can think that either a malfunction occurred or that information is missing. Notice that this case can also occur with higher levels of abstraction. It is, in fact, desirable that these hypotheses are considered in higher levels of abstraction as the ECTE can better reason in the lower levels (in which time and alarm correlation are prioritary issues), providing higher abstraction level conclusions, which will be the DRIMs base for conclusions and hypotheses.

The DL can handle hypotheses very well and presents non-monotonic behaviour, thus being able to update its beliefs, given additional knowledge. Hypotheses that were once valid can easily be discarded and new ones are generated, ensuring that all the possible knowledge for decision support is available to the user.

To ensure that the reasoning about actions represents

reality, time is an issue and it has to be present in the system as it is definitely present in the electrical network. Therefore, some of the rules of default rules will be dependant on how much time has elapsed, for instance, between the occurrences of events *T1* and *T2* (notice that an event in the DRIM can be a state change in an ECTE fluent). Another example is a default rule which assumes some consequence if an event arrived at instant *T1* and no other event of a particular type have occurred in the last *T2* time units since *T1*. The former introduces the need for temporal constraints, present as the special predicates: *less_than(T1,T2)*; *more_than(T1,T2)*. These predicates test whether *T1* is lesser or greater than *T2* respectively.

4.3 Real-Time Operation

Real-time events arrive from the source (SCADA system) and are handed to the DRIM. The DRIM sends them to the ECTE and, based on the metaknowledge it has about the event, it will find a subset of applicable default rules (this is already an efficiency factor) and creates this set's expansions. These expansions originate on the DRIMs default rules and other predicate logic rules, which can be based on the ECTEs possible conclusions. This allows the DRIM to check domain coherence and possible hypotheses for unusual sequences of conclusions from ECTE, leaving the ECTE with the hard task of temporal correlation between events.

Real-time operation (even soft real-time) must ensure small and limited response times. This has required the introduction of some improvements made in the paradigms, namely the inclusion of metaknowledge in the DRIM and the use of a cache in the ECTE.

The module which presents the most processing time consumption is the DRIM. This module queries the ECTE, when the default knowledge base requires it and these queries are time consuming, so they must be minimal. On the other hand, the expansion generation can also be very time consuming. The expansion generation issue is dealt with metaknowledge, giving each new event a set of meaningful default rules, which is then expanded, keeping default rules, and hence their premises are minimal. The expansion generation is limited through the use of Prioritized Default Logic (PDL) [15]. This is a restriction on top of DL that assigns pairs of default rules with a priority among them, reducing the number of extensions and thus their processing time.

5. Example

5.1 Event Calculus Temporal Expert

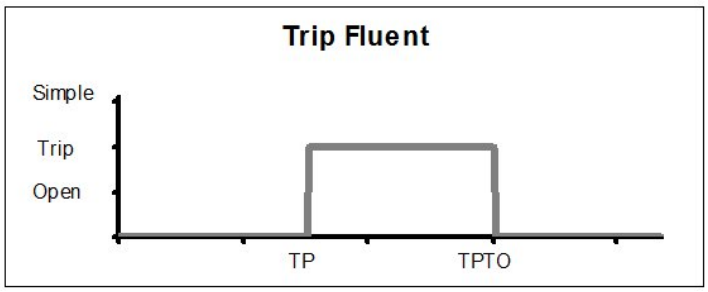
As an example, the identification of the simple tripping will be presented. The simple tripping is composed of a tripping signal, usually emitted by a protection device, and the opening one or more breakers, signalling the actual performance of the tripping order. The necessary rule base for the identification of the (simplified) simple tripping situation is depicted below in ECEX1 through ECEX8. This identification is based on three SCADA message types: device tripping, device opening and device closure. The complete identification would have to contain additionally reclosure and bypass information which, for simplification, will not be included in this example.

This rule base is not an ordinary production rule based, as it has been stressed in the paper, but a collection of PROLOG rules and facts concerning the initiation and termination of fluents. In this case, the incoming events are represented by PROLOG terms which are composed of two arguments: the device identification and the type of alarm. This is obviously a simplification of the real case.

```
initiates(fact(ID,'>>Trip'),                          (ECEX1)
tripFluent(ID), T).

isTimedout(fact(ID,'>>Trip'),                         (ECEX2)
tripFluent(ID)).

initiates(fact(ID,'Open'),                            (ECEX3)
openFluent(ID), T).

isTimedout(fact(ID,'Open'),                           (ECEX4)
openFluent(ID)).

terminates(fact(ID,'Closed'),                         (ECEX5)
openFluent(ID), T).

initiates(fact(ID,'>>Trip'),                          (ECEX6)
simpleTripFluent(ID), T):-
holdsAt(openFluent(ID), T).

initiates(fact(ID,'Open'),                            (ECEX7)
simpleTripFluent(ID), T):-
holdsAt(tripFluent(ID), T).

terminates(fact(ID,'Closed'),                         (ECEX8)
simpleTripFluent(ID,L),T).
```

This particular rule base has three subcomponents, each of which is identified with a particular fluent. The first subcomponent (ECEX1 and 2) identifies the trip fluent. A SCADA message with tripping information initiates the fluent and its respective timeout terminates it. The second subcomponent is related to device opening information (ECEX3, 4 and 5). ECEX3 and 4 are similar to ECEX1 and 2. The only enhancement is that ECEX5 states that the fluent mapping device opening can also be terminated by closure information. Finally, the third subcomponent is related to simple tripping. ECEX6 states that simple tripping occurs if tripping information has arrived and the device is open. ECEX7 maps the situation in which device opening information arrives when the device has recently tripped. The simple trip is, of course, terminated by device closure as depicted in ECEX8. The arrival of opening (OP) and tripping (TP) information activates the simple trip fluent, which is only deactivated by a device closure (in this small knowledge base example).

For a better comprehension an example, an instantiation of the arrival ordered set of events will be presented. The usual order of events is: a TP signal which is due to the detection of some malfunction in a line, associated with its timeout TPTO. Afterwards the device affected by the malfunction should open to insure the isolation of



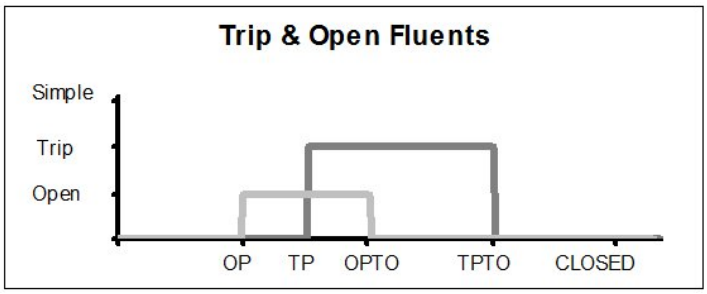


Figure 3. Trip and open fluents.

the line, included with the timeout OPTO. This incident eventually ends when the malfunction is resolved and service can be restored. The arrival information in this order can be shown initially in Fig. 3. Both in Figs. 3 and 4 the ordinate axis represents boolean values for each of the fluents: open, trip and simple trip. These fluents can only be true or false, but the use of only one truth value in the ordinate axis confused the graphs in the authors' opinion. The False value is equal for all of the fluents, coinciding with value zero of the ordinate axis.

The events of Fig. 3 are as follows:

- Initially event TP arrives and is included in the KB with its respective timeout TPTO. This originates the trip fluent by ECEX1 and 2.
- Afterwards event OP arrives and is included in the KB with its respective timeout OPTO. This originates the open fluent by ECEX3 and 4. It also originates the detection of an area where the open and trip fluents are valid and by ECEX7, and hence the simple trip fluent presented in Fig. 4.
- The simple trip fluent is finally terminated by a closure event by ECEX8 as is also presented in Fig. 4.

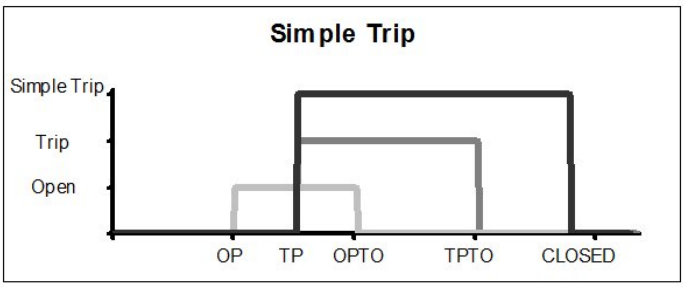


Figure 4. Simple trip fluent.

This example shows one usual alarm correlation performed by operators in CCs. It is robust to lack of chronology because only the overlapping of the temporal relevance windows is used.

Notice also the non-monotonic abilities of the EC. If other information arrives, or information in the fact base is altered for any reason, the fluents change accordingly and new information is derived to explain the actual events. This can be seen in this particular example. If closure information is received and its time stamp places it before the opening timeout, the fluents change.

5.2 Default Reasoning Incoherence Management

The DRIM is used to fill information when incompleteness occurs. An example will be presented for a better understanding of the handling of incompleteness. Consider the following defaults (Defaults 4, 5 and 6):

$$\frac{RTpT(T1) : TpT(T2) \wedge less\_than(|T2 - T1|, 1000)}{DTpT(T1)}$$

Default 4

$$\frac{RTpT(T1) \wedge Trip(T2) \wedge more\_than(T2, T1) : Open(T3) \wedge less\_than(|T3 - T2|, 300)}{TpT(T2)}$$

Default 5

$$\frac{RTpT(T1) \wedge Open(T2) \wedge more\_than(T2, T1) : Trip(T3) \wedge less\_than(|T3 - T2|, 300)}{TpT(T2)}$$

Default 6

Default 4 means that if there has been a reclosed three-phase tripping ($RTpT$) and is possible to assume that another three-phase tripping (TpT) has occurred within a 1000 ms neighbourhood, then it is possible to conclude that a definitive three-phase tripping ($DTpT$) took place. *Default5* states that if an RTpT happened followed by a trip, and it is possible to assume that the tripping device is open within a 300 ms neighbourhood of the trip, then it can assume that there was a tripping. *Default6* means that if there has been both an RTpT and an opening, and it is possible to assume that the opening device has tripped within a 300 ms neighbourhood, then it can assume that there was a tripping. Both default theories *Th1=({RTpT(T1), Trip(T2) }, { Default4, Default5, Default6 })* and *Th2=({RTpT(T1), Open(T2) }, { Default4, Default5, Default6 })* will yield include $DTpT$ in their single extension. In other words, if the ECTE has concluded $RTpT$ and due to information incompleteness one and only one event the set *{Trip(T2), Open(T2)}* arrives, instead of the normal occurrence of both events, the DRIM can assume the other event in the absence of contrarian information and therefore infer one of the possibly correct diagnosis, which is $DTpT$. Notice however that as one and only one event the set *{Trip(T2), Open(T2)}*,

with the inclusion of additional default rules, other diagnoses would be possible. These additional diagnoses could include, for instance, a falsely generated alarm, due to generator device malfunction, which is an easily achievable conclusion.

The interaction of the ECTE and the DRIM allows default reasoning to be performed over an arbitrarily complex temporal inference, raising the abstraction level and allowing the direct representation of the expert's views on the task. As an example, *Defaults 4, 5 and 6* capture the sentence "If an RTpT occurred, and shortly afterwards another tripping can be assumed, then the diagnosis is one of DTpT".

6. Application

The application of the described techniques was implemented in the Portuguese Electrical Transport Network, in Vermoim. The Component-Based Development [16] was performed using a development environment based on the PVM (Parallel Virtual Machine) [17]. Upon PVM were developed the following components:

- ECTE, deployed in PROLOG
- DRIM, deployed in PROLOG
- GUI, deployed using X/Motif
- SCADA interconnector, deployed in C++

The interaction with the SCADA system SIEMENS Sinault Spectrum was achieved, integrating the system in the Sinault Spectrum data network. This system's Graphical User Interface (GUI) network representation is a replica of the actual network representation used in the SCADA GUI. Nevertheless, this GUI has the ability to be event driven and to adapt to incident situations, allowing an automatic zoom on incident areas, based on the intelligent components inferred conclusions. The GUI can be seen in Fig. 5.

A deeper study on the computational complexity of this approach is needed but nevertheless, the application has performed very well in the validation scenarios. The validation scenarios available include the simulation of previous occurrences, using alarm files which have been generated by the SCADA system in previous incidents and recorded for future use. The other validation scenario is the real, online diagnosis of the SCADA alarms in the Portuguese Electrical Transport Network.

7. Conclusion

An architecture for a decision support system applied to incident analysis in an electrical power transmission network has been presented. It has the ability to deal with temporal, incomplete and domain incoherent information. The objective is to be able to deploy a system, which can intelligently reason about time and action and do so with a response time neglectable for a CC operator. For this purpose, an hybrid architecture was designed, using the EC for temporal reasoning and robustness to chronological incoherence and the DL for domain incoherence detection with hypotheses generation for a more accurate decision support.

A prototype implementation is being tested and validated in the Portuguese Electrical Network Company (REN). It is important to refer that its results are more reliable than those of SPARSE, its predecessor (in cases where events have chronological problems, or some incompleteness like one missing event). The DRIM, raises

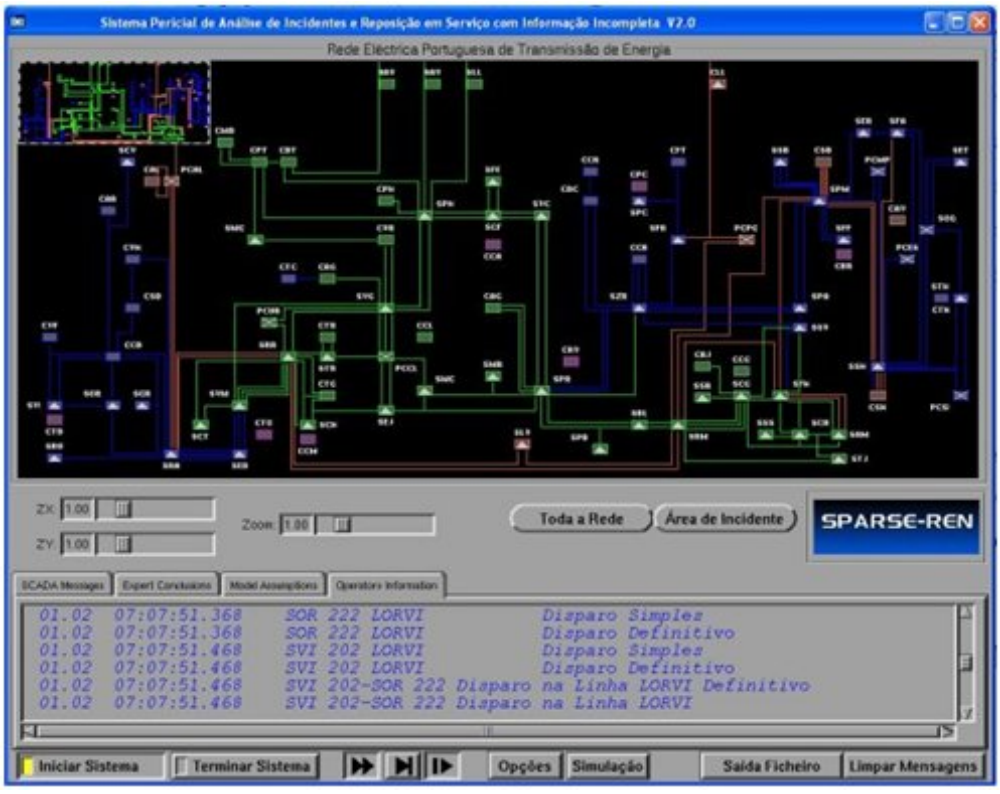


Figure 5. Graphical user interface.

reasonable doubt and presents the various scenarios when information is incomplete, allowing the CC operator to be advised for all possible incidents. With the reception of more information the scenarios can merge, originating only one incident. In every case, the CC operator gets all the possible decision support: conclusions when the information is complete or reasonable assumptions when information is incomplete.

In terms of future directions, the paradigms and its interconnection could easily be used as the bulk of an intelligent agent, applicable to any area in which real-time information, temporal knowledge and decision support are needed. Also both the existing intelligent tutoring and the validation systems will be adapted (or redesigned) for this new architecture.

References

[1] CIGRE—Working Group 38.06.03, Expert systems: Development experience and user requirements, *Electra, 146*, 1993, 29–67.

[2] A. Insfrán, A. Alves da Silva, & G. Lambert-Torres, Fault diagnosis using fuzzy sets, *Proc. Int. Conf. on Intelligent Systems and Applications (ISAP'99)*, Rio de Janeiro, Brazil, 1999.

[3] P. Kádár, A. Kovács, & A. Mergl, A tolerant event recogniser and alarm filter based on sequential pattern matching under introduction into the national dispatching centre, *Proc. Int. Conf. on Intelligent Systems and Applications (ISAP'97)*, Seul, Korea, 1997.

[4] Z. Vale, Power systems control centers, in C.T. Mondes (Ed.), *Intelligent systems—technology and applications*, VI (CRC Press, 2003), VI-63–VI-112.

[5] N. Malheiro, Z. Vale, C. Ramos, J. Santos, & A. Marques, *Enabling client–server explanation facilities in a real-time expert system*, Lecture Notes in Computer Science, Vol. 1611 (Springer-Verlag, 2004), 333–342.

[6] R.A. Kowalski & M.J. Sergot, A logic-based calculus of events, *New Generation Computing*, *4*, 1986, 67–95.

[7] M. Shanahan, The event calculus explained, *Artificial intelligence today*, 1600 of Lecture Notes in Computer Science, in M. Wooldridge & M. Veloso (Eds.) (Springer-Verlag, 1999), 409–430.

[8] R.A. Kowalski & F. Sadri, Reconciling the situation calculus and event calculus, *Journal of Logic Programming*, *Special Issue on Reasoning about Action and Change*, *31*, 1997, 39–58.

[9] F. Sadri & R. Kowalski, Variants of the event calculus, *Proc. of the Int. Conf. on Logic Programming*, Kanagawa, Japan, 1995, 67–81.

[10] I. Cervesato, M. Franceschet, & A. Montanari, The complexity of model checking in modal event calculi with quantifiers, *Electronic Transactions on Artificial Intelligence*, *2*(1–2), 1998, 1–23.

[11] R. Reiter, A logic for default reasoning, *Artificial Intelligence*, *13*, 1980, 81–132.

[12] G. Antoniou, *Nonmonotonic reasoning* (MIT Press, 1997).

[13] T. Finin, D. McKay, R. Fritzson, & R. McEntire, KQML: An information and knowledge exchange protocol, in K. Fuchi & T. Yokoi (Eds.), *Knowledge building and knowledge sharing* (Ohmsha and IOS Press, 1994).

[14] L. Chittaro & A. Montanari, Efficient temporal reasoning in the cached event calculus, *Computational Intelligence*, *12*(3), 1996, 359–382.

[15] J. Rintanen, Lexicographic priorities in default logic, *in Artificial Intelligence*, *106*(2), 221–265.

[16] C. Szyperski, *Component software: Beyond object-oriented programming*, 2nd edn (Addison-Wesley, 2002).

[17] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, & V. Sunderam, PVM: Parallel virtual machine—A users' guide and tutorial for networked parallel computing, in *Scientific and engineering computation* (MIT Press, 1994).

Biographies

N. Malheiro has currently the rank of Assistant Professor at the Polytechnic Institute of Porto. He received his Ph.D. in Electrical Engineering from the University of Trás-os-Montes e Alto douro. His research interests concern temporal and non-monotonic reasoning. He has been involved in the Knowledge Engineering and Decision Support Research Group, connected to the Portuguese Transmission Network company, developing a decision support system for incident situations.

Z. Vale is a Coordinator Professor of Electrical Engineering at the Polytechnic Institute of Porto's Institute of Engineering. Her research interests are decision support and artificial intelligence. She received her Ph.D. in Electrical Engineering from the University of Porto. She is a member of the IEEE, IEE and ACM.

C. Ramos is a Coordinator Professor of Computer Engineering and Director of the Knowledge Engineering and Decision Support Research Group at the Polytechnic Institute of Porto's Institute of Engineering. His research interests are artificial intelligence and decision support systems. He received his Ph.D. in Electrical Engineering from the University of Porto.

M. Cordeiro is a full-time professor at the University of Trás-os-Montes and Alto Douro. His research interests are rational energy use, grounding systems and power systems applications. He received his Ph.D. in Electrical Engineering from the University of Trás-os-Montes and Alto Douro.

A. Marques is the Director of the Exploration Division in the Portuguese Transmission Network. He received a Diploma in Electrical Engineering and an MS from the Faculty of Engineering at the University of Porto.

V. Couto is the Head of the Department of Operation in the System Management Division of the Portuguese Transmission Network. He received a Diploma in Electrical Engineering from the University of Porto.