1 Introduction

In any industrial process or system, unexpected failures in equipment should be detected and diagnosed for repair as soon as possible to minimize downtimes, even though this is not an easy task. The real faults are masked by dozens of symptoms, which are really a cascaded effect of the major faults. The intrinsic uncertainty in the information requires an intelligent system, to interpret the data and diagnose the abnormal components of the process.
In recent years much research has been devoted to the diagnosis of industrial processes. Most of the industrial implementations of diagnostic systems focus on either qualitative or quantitative techniques. The first, although perform well with definite observations, are weak to analyze continuous observations (e.g. sensors readings). The quantitative techniques focus on the analysis of continuous data and have weak inference engines for discrete data. They normally include mathematical models that are hard to obtain for many processes. This make quantitative approaches very difficult to use in real systems.
Other Fault diagnosis methods based on neural networks and fuzzy logic, have both their own limitations. Neural networks represent black box models that do not provide more relevant information than the one contained in the output. In neural networks is also difficult to integrate knowledge from other sources. Other disadvantages are overfitting, large learning times and lack of ability to deal with missing information. Fuzzy-based approaches can not deal with missing information in explicit form, and the overall dimension of rules, may blow up strongly even for small components or processes [8]. The nature of industrial processes, composed of both, discrete and continuous data, suggests the use of hybrid diagnostic engines. Additionally, the diagnostic system has to be able to handle noise, nonlinearities and missing information.
In this paper we show how to make a structured integration of FDI techniques [3], with artificial intelligence model-based diagnosis, within the probabilistic logic framework given by the Dynamic Independent Choice Logic. To analyze discrete signals we are using a model-based diagnosis approach. Model-based diagnosis is based on the fact that a component is faulty if its correct behavior, given by a stored model, is inconsistent with the observations. To deal with continuous signals, we use a fault detection and isolation (FDI) approach, commonly used by the control systems community, that includes dynamic probabilistic models. We have substituted the FDI classical models, such as differential equations or ARMA models, with dynamic probabilistic models, in order to deal with noisy, highly non-linear models, possibly with missing data inputs. These models have simple structure and inference engine, and provide a good approximation to the real signals. A main characteristic of our models, is that we learn the structure (causal model), and the parameters (Lagrange coefficients), from raw data without any preprocessing stage. The raw data may include noise and missing data. The probabilistic models have an accuracy enough to allow the discrimination between the steady state of the process and different mode faults.
We apply our method in a simulated industrial-strength power transmission network.

2 Description of the Approach

The main relevant aspects of our approach are:

1. We adopt as a diagnostic strategy, the repair-man perspective [13], where the aim of the diagnostic search is the location of the root-cause of the disturbance. In this case, topographic reference to the location of the disturbance must be drawn from the first observations. Once the affected area is located, more specific information is analyzed (e.g. sensor signals) to isolate the abnormal components.
2. We modularize the fault detection and diagnosis tasks with the introduction of agents. In our approach, there are three types of agents: *Nature* is regarded as an agent that provides stochastic assumptions about components behavior. The *Alarm Processor* (AP) agent produces a set of explanations consistent with first observed symptoms. *Fault Detection* (FD) agents associated to every component in the process, are modeled as dynamic agents specifying how streams of sensor data entail fault decisions. The output of the AP agent represents a partial diagnosis to be confirmed by FD agents (see figure 1).
3. The signals produced by sensors located in the candidate components of the system, are analyzed by FD agents. Every component has and FD agent that represents a transduction from inputs (the sensor values) into outputs (the fault/no-fault decision). Transductions represent an abstraction of dynamic systems [16].
4. The fault detection agent incorporates a predictive causal model, representing the no-fault behavior of the component. The model structure is generated from steady-state data, with a Bayesian learning algorithm. This model includes the temporal relationships between process signals. The model delivers a probability distribution over the forecast variable states, computed with a maximum entropy classifier algorithm.
5. The FD agent compares the one-step ahead prediction of the no-fault model and the stream of data provided by the sensors. The residual analysis gives an indication of the component behavior (normal/abnormal).
6. The logic programs representing the agents, are axiomatized in *phase space* [2] in a similar manner to the event calculus.
7. The specification for a FD agent is not evaluated as a logic program that needs to do arbitrary computation reasoning about the past. We know the data from sensors have been already received, and the reasoning about the fault decision depends on the processing of the received inputs.
8. We use the Dynamic Independent Choice Logic (DICL) [11] with a discrete time structure, as the framework to develop the fault detection and diagnosis approach. We benefit from the compact knowledge representation of logic, the handling of uncertainty with probabilities, the modularization capabilities, and the ability to represent temporal relations of the DICL.

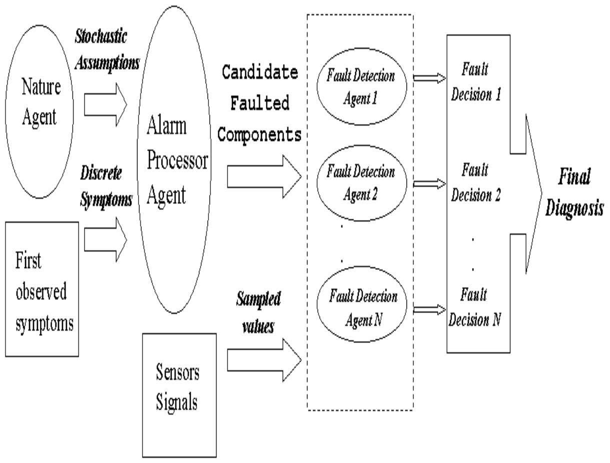

Figure 1. Agents based Diagnosis

3 REPRESENTATION

The dynamic independent choice logic is a theory built upon a general model of agents interacting in an environment [11]. We assume a discrete time structure $\mathcal{T}$, that is totally ordered and has a metric over intervals. A *trace* is a function from $\mathcal{T}$ into some domain $\mathcal{A}$. A *transduction* is a function from input traces into output traces, that is causal in the sense that the output at time t can only depend in inputs at times t' where $t' \leq t$. An agent is a specification of a transduction. A *state* is the information that needs to be remembered in order for the output to be a function of the state and the current inputs.
We modify slightly the definition of a dynamic ICL stated in [11], to introduce the diagnostic dynamic independent choice logic (DDICL) [5]:

Definition 1. A **diagnostic dynamic independent choice logic theory** (DDICL) is a tuple $\langle \mathcal{A}, \mathcal{C}_0, \mathcal{F}_{AP}, \mathcal{P}_0, \mathcal{ASM}_a \rangle$, where

- $\mathcal{A}$ is a finite set of agents containing three types of agents: *Nature*, *Alarm Processor*, and *Fault Detection* agents,
- $\mathcal{C}_0$, **Nature's choice space**, is a choice space with alternatives controlled by nature,
- $\mathcal{F}_{AP}$, is the logic program specification for the Alarm Processor agent. This agent generates a set of explanations consistent with first observed symptoms.
- $\mathcal{P}_0$ is a function $\bigcup \mathcal{C}_0 \rightarrow [0,1]$ such that $\forall \chi \in \mathcal{C}_0 \quad \sum_{\alpha \in \chi} \mathcal{P}_0(\alpha) = 1$,
- $\mathcal{ASM}_a$ is a function on $\mathcal{A} - 0 - AP$ such that $\mathcal{ASM}_a$ is an agent specification module for Fault Detection agent a.

We extend the definition for an *Agent Specification Module* (definition 2.1 given in [10]) with the notion of *probabilistic observation function*, to specify a fault detection agent:

Definition 2. An agent specification module for FD agent $a \neq \{0, AP\}$, written $\mathcal{ASM}_a$, is a tuple $\langle\, \mathcal{I}, \mathcal{O}, \mathcal{R}, \mathcal{L},\, \mathcal{F}_a,\, \phi\rangle$ where

- $\mathcal{I}$ is a set of fluents called the **inputs**. They specify what sensor values are available at various times. The range the input trace is the cross product of the ranges of the fluents in the inputs.
- $\mathcal{O}$, is a set of fluents called the **outputs**. An output is a propositional fluent that specifies a decision about the existence of a fault in a component at various times.
- $\mathcal{R}$, is a set of fluents called the **recallable** fluents. These are fluents whose previous values can be recalled.
- $\mathcal{L}$, is as set of fluents called the **local** fluents. These are fluents that are neither inputs, outputs nor recallable.
- $\mathcal{F}_a$ is an acyclic logic program. $\mathcal{F}_a$ specifies how the outputs are implied by the inputs, and perhaps previous values of the recallable fluents, local fluents, arithmetic constraints and other non-temporal relations as intermediaries.
- ϕ, is the **probabilistic observation function**, $\phi : Q \rightarrow \ \mathcal{P}_V$, mapping observation states into a distribution over predicted states.

In this paper we emphasize the description of the fault detection agents. The details about the alarm processor agent can be found in [4].

3.1 Dynamics Modeling

The specification for an FD agent, makes use of probabilistic functions as a means of modeling the steady-state dynamics of sensor measurements. The sensor measurements are discretized in fixed bins. The model represents the no-fault behavior of the associated component.
To implement the function ϕ, we developed a forecast modeling approach [5] with the statistical inference engine based on the maximum entropy principle [15] (see figure 2). The steady state causal models (structure and parameters) are learned offline from discretized process data. When a stream of sensor data needs to be analyzed, the inference engine provides a probability distribution over discrete states of the sensor variable. The probabilistic inference amounts to the computation of:

$$P[V = v|Q = q] = \frac{exp\left(\sum\limits_{i=1}^{N} \gamma(Q_i = q_i, V = v)\right)}{\sum\limits_{v'=1}^{K} exp\left(\sum\limits_{i=1}^{N} \gamma(Q_i = q_i, V = v')\right)} \qquad (1)$$

where

- Q is a finite nonempty set of observation states,
- V is a finite nonempty set of predicted states.

The subset of Lagrange multipliers $\{\gamma(Q_i = q_i, V = v), i = 1, \ldots, N, q_i = 1, \ldots, |A_i|, v = 1, \ldots, K\}$ are learned in an offline manner with a deterministic annealing algorithm.
Equation 1 is an approximate solution based on the method proposed by Cheeseman to find the maximum entropy joint probability mass function (pmf) consistent with arbitrary lower order probability constraints. The method by Yan and Miller uses a restriction of joint pmf support (during learning) to a subset of the feature space (training set support). The small size of the training set support maintains the learning times quite tractable.
The comparison between the prediction given by the sensor steady state model and the stream of data, delivers a set of residuals that provide information about possible faulty modes.

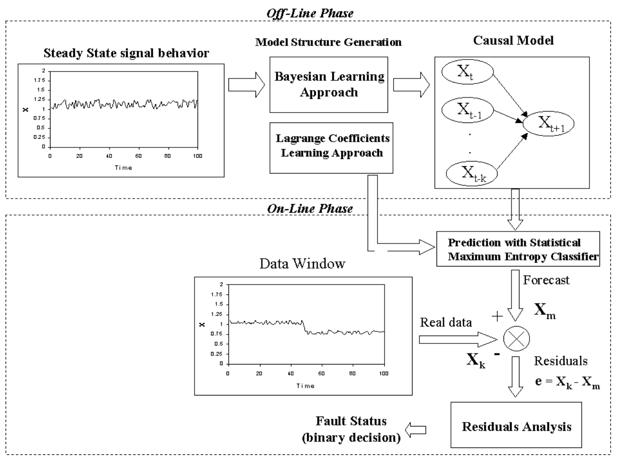
Off-Line Phase
Steady State signal behavior
Model Structure Generation
Causal Model
Bayesian Learning Approach
Lagrange Coefficients Learning Approach
X_t
X_{t-1}
X_{t+1}
X_{t-k}
Time
On-Line Phase
Prediction with Statistical Maximum Entropy Classifier
Data Window
Forecast
X_m
Real data
X_k
Residuals
$e = X_k - X_m$
Fault Status (binary decision)
Residuals Analysis

Figure 2. Structure of a Fault Detection Agent

3.2 Temporal Logic Definitions

We specify FD agents with acyclic logic programs and predicates that explicitly refer to time. The acyclicity corresponds to temporal ordering (if t_1 and t_2 are time points and $t_1 < t_2$, then the acyclic index for t_1 is lower than acyclic index for t_2). Every temporal predicate refers to the value of a fluent in a given time point. A *fluent* is a function that depends on time.
Every fault detection agent has a local time scale, what give us the ability to represent the different dynamics in a process. For example, in an industrial reheating furnace, with 1 or 2 samples a minute, we can track the temperature inside the furnace, whereas the electric power feeding the furnace needs to be known at a higher rate (probably 10 to 30 samples a second).
To recall the past values of fluents, we extend the version of temporal predicates

defined in [10], to allow the recalling of values beyond previous time point (values of fluent Fl at time point $t-n$, where $n>1$) :

was(Fl, Val, T_n,T) is a predicate that specifies that recallable fluent *Fl* was assigned value *Val* at time T_n.

$$was(Fl, Val, T_n, T) \leftarrow time(T_n) \wedge T_n < T \wedge set(Fl, Val, T_n) \wedge \sim reset\_before(Fl, T_n, T).$$

where $reset\_before(Fl, T_n, T)$ is true if fluent Fl was assigned a value in the interval (T_n, T):

$$reset\_before(Fl, T_n, T) \leftarrow time(T_2) \wedge T_n < T_2 \wedge T_2 < T \wedge set(Fl, Val2, T_2).$$

now(Fl, Val, T) is a predicate that specifies that recallable fluent *Fl* has value *Val* at time *T*.

$$now(Fl, Val, T) \leftarrow set(Fl, Val, T).$$
$$now(Fl, Val, T) \leftarrow \sim \exists V_1 set(Fl, V_1, T) \wedge was(Fl, Val, T_1, T).$$

set(Fl, Val, T_n) is a predicate that specifies that recallable fluent Fl has value Val at time T. This predicate implements the reading of sensor values received up to the time of failure.

time(T) specify that T corresponds to a point in a discrete, linear time scale.

4 Example

We show the details of our method with a small power network (see figure 3). This network, represents the interconnection of the different components. The *buses* are nodes where industrial or domestic users are connected. The *lines* allows the transference of electrical power between *buses*. The *breakers* help to isolate a fault event from the rest of the network.
The breakers are the main protection for one bus and the backup protection for the bus at the other end of the line. For instance, breaker *Br12*, is the main protection for *bus 1* and the backup protection for *bus 2*. This scheme of backup protection allows the isolation of a fault, even in the case of a malfunction in the main breaker.

We assume the buses (nodes) are the only source of faults and there is only one type of fault (e.g., three phase-to-ground fault). The fault persists during the time span of the diagnosis. The first symptoms are the alarms indicating the status of protection breakers (e.g., *open* or *failed to open*). There is also a possibility that the status of some breakers is unknown.

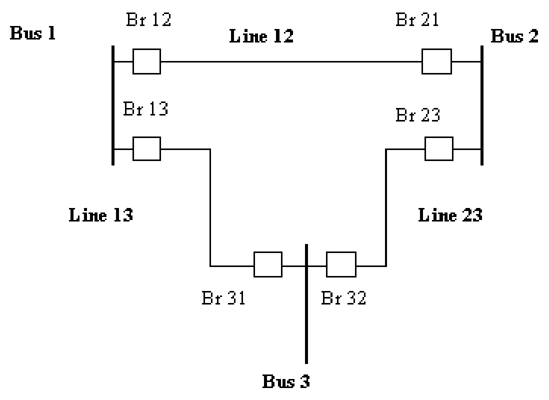

Figure 3. Single line diagram of a small power network

We axiomatise this as follows:

Nature's Alternatives $\mathcal{C}_0$

$$\begin{aligned}\mathcal{C}_0 = \{&bus1(faulted) : 0.01,\\ &bus1(ok) : 0.99, bus2(faulted) : 0.02,\\ &bus2(ok) : 0.98, bus3(faulted) : 0.025,\\ &bus3(ok) : 0.975\}\\ &\{c\_br12\_open(open, faulted, ok) : 0.99, \ldots,\\ &c\_br32\_open(open, faulted, faulted) : 0.98\}\end{aligned}$$

Alarm Processor Facts $\mathcal{F}_{AP}$.

$$br12(StBr) \leftarrow bus1(Stbus1) \ \wedge \ bus2(Stbus2) \ \wedge \ c\_br12\_open(StBr, Stbus1, Stbus2)$$
$$br21(StBr) \leftarrow bus2(Stbus2) \ \wedge \ bus1(Stbus1) \ \wedge \ c\_br21\_open(StBr, Stbus2, Stbus1)$$
$$br13(StBr) \leftarrow bus1(Stbus1) \ \wedge \ bus3(Stbus3) \ \wedge \ c\_br13\_open(StBr, Stbus1, Stbus3)$$
$$br31(StBr) \leftarrow bus3(Stbus3) \ \wedge \ bus1(Stbus1) \ \wedge \ c\_br31\_open(StBr, Stbus3, Stbus1)$$
$$br23(StBr) \leftarrow bus2(Stbus2) \ \wedge \ bus3(Stbus3) \ \wedge \ c\_br23\_open(StBr, Stbus2, Stbus3)$$
$$br32(StBr) \leftarrow bus3(Stbus3) \ \wedge \ bus2(Stbus2) \ \wedge \ c\_br32\_open(StBr, Stbus3, Stbus2)$$

All the axiomatisation above refers to the same time T_a, when the alarms where received (ignoring small delays). For this reason T_a was omitted from the clauses.

FD Agent Module
Agent Facts $\mathcal{F}_{bus1}$.

$$\begin{aligned}
val(Voltage\_bus1\_model, V_{b1}^{m}, T) &\leftarrow was(Voltage\_bus1\_real, V_{b1}^{R1}, T-1, T) \ \wedge \\
&\quad was(Voltage\_bus1\_real, V_{b1}^{R5}, T-5, T) \ \wedge \\
&\quad c\_voltage\_bus1\_model(V_{b1}^{m}, V_{b1}^{R1}, V_{b1}^{R5}) \\
val(Residual\_Voltage\_bus1, R_{Vb1}, T) &\leftarrow now(Voltage\_bus1\_model, V_{b1}^{m}, T) \ \wedge \\
&\quad now(Voltage\_bus1\_real, V_{b1}^{R}, T) \ \wedge \\
&\quad R_{Vb1} = V_{b1}^{m} - V_{b1}^{R} \\
fault\_bus1(yes, T) &\leftarrow now(Residual\_voltage\_bus1, R_{Vb1}, T) \ \wedge \\
&\quad R_{Vb1} \ > \ \lambda_{Vb1} \\
fault\_bus1(no, T) &\leftarrow now(Residual\_voltage\_bus1, R_{Vb1}, T) \ \wedge \\
&\quad R_{Vb1} \ < \ \lambda_{Vb1}
\end{aligned}$$

We omit the specification for the fault detection agents at *bus 2* and *bus 3*, because is similar to the above presented.
The stream of data received from voltage sensors, starts at T_i and ends at T_f. We assume nothing regarding the frequency of data sampling as long as we can distinguish between normal and abnormal behavior, analyzing the data. The only assumption is that $T_i < T_a < T_f$, that is the voltage sensors data are related to the alarms received. This allow us to represent different time scales in each fault detection agent.
The structure of the predictive causal model for the variable voltage of component $bus1$, was generated offline from steady-state data, with a suitable Bayesian learning algorithm (we use the algorithm given in [1]). In this case the output of the algorithm delivers that fluent $Voltage\_bus1\_model$ depends on past observations of fluent $Voltage\_bus1\_real$ at times $T-1$ and $T-5$.
The stochastic part of the agent is achieved by the function

$$\phi \quad c\_voltage\_bus1\_model(V_{b1}^{m}, V_{b1}^{R1}, V_{b1}^{R5})$$

that maps the set of past observations $\{V_{b1}^{R1}, V_{b1}^{R5}\}$ to V_{b1}^{m}.

We have applied our approach in a simulated industrial-scale electrical power network. The estimation of the fault location is difficult due to the presence of multiple faults, the overwhelming number of alarms generated, and the possibility of malfunction of protective devices.
The simulated electrical power network has 24 buses, 34 lines and 68 breakers. We have tested our approach with multiple events, multiple types of faults and missing information on sensors. More details are given in [6]. The accuracy of our method to identify the real faulted components was higher than 70 %.

5 Related Work

The concept of diagnosis agents in technical processes was applied in [9]. In Lauber's work, the agent is based on a dynamic Bayesian network (DBN) for reasoning over time. The structure of the DBN was built with reliability engineering methods (e.g. failure mode and effects analysis). Our agents handle the dynamics of the environment with a more general class of dynamic networks (i.e. We allow non-Markovian forecast models), whose structure was built with algorithms that learn Bayesian networks. HybridDX [12] is a diagnostic system used in aeronautics, that include model-based diagnosis and continuous and discrete simulations of continuous processes. To model the dynamics in HybridDX, they use physical causal models that are frequently not known in analytical form or too complicated for calculations. In [14], Sampath presents a hybrid approach that incorporates the concept of virtual sensors and discrete events diagnosis. The analysis of sensor signals is performed by using different techniques, such as spectral analysis, principal components analysis and statistical discrimination. This approach assumes single fault scenarios and does not address the problem of incomplete sensor data.

6 Conclusions

We have presented a diagnostic system framework that integrates: the dynamic independent choice logic with multiple agents, probabilistic forecast models, and fault detection and isolation techniques. The diagnostic strategy resembles the approach of a repair-man, in which the first effort is the location of the affected area and then a more specific analysis is achieved to isolate the abnormal components. We split the diagnosis task in two phases: the first phase is performed by the alarm processor (AP) agent that actuates as the disturbed area locator. The output of AP agent is a set of component candidates to be confirmed as real faulted components by the second phase. The second phase is performed by fault detection agents, that analyze the behavior of sensor measurements. The FD agent incorporates a one step-ahead forecast model describing the no-fault model of the component. The analysis of residuals, computed from the differences between the no-fault model and the sensor measurements, give the final decision about the fault in a component.

References

1. J. Cheng, D. Bell, and W. Liu (1998). Learning bayesian networks from data: An efficient approach based on information theory. Technical Report, Dept. of Computing Science, University of Alberta, Alberta CA.
2. T. Dean, and M. Wellman (1991). *Planning and control.* San Mateo, Calif.: Morgan Kaufmann, 1991.
3. P. Frank (1990). Fault diagnosis in dynamic systems using analytical and knowledge based redundancy–a survey and new results. *Automatica* 30: 789-804.

4. L. Garza, F. Cantú, and S. Acevedo (2000). A Methodology for Multiple-fault Diagnosis based on the Independent Choice Logic. In Springer *LNCS/LNAI Proc. of the IBERAMIA-SBIA 2000*, pages 417-426, Sao Paulo, Brasil, November 2000.
5. L. Garza (2001). Hybrid Systems Fault Diagnosis with a Probabilistic Logic Reasoning Framework . PhD Thesis, Instituto Tecnolgico y de Estudios Superiores de Monterrey, Monterrey, N.L., December 2001.
6. L. Garza, F. Cantú, and S. Acevedo (2002). Fault Diagnosis in Industrial Processes with a Hybrid Diagnostic System. In Springer *LNAI Proc. of the MICAI 2002*, pages 436-445, Mrida, Yuc., Mxico, April 2002.
7. R. Isermann (1997). Supervision, fault-detection and fault-diagnosis methods - an introduction. *Control Engineering Practice*, **5** (5): 639-652.
8. R. Isermann (1997). On Fuzzy Logic Applications for Automatic Control, Supervision, and Fault Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 28, No. 2, pp. 221-235.
9. J. Lauber, C. Steger, and R. Weiss (1999). Autonomous agents for online diagnosis of a safety-critical system based on probabilistic causal reasoning. In *Proc. Fourth Intl. Symposium on Autonomous Decentralized Systems 1999*, pags. 213-219.
10. D. Poole (1995). Logic programming for robot control. In *Proc. 14 th International Joint Conference on AI, Montreal, August, 1995*, 150-157.
11. D. Poole (1997). The Independent choice logic for modeling multiple agents under uncertainty . *Artificial Intelligence*, **94**:7-56 .
12. G. Provan and D. Elsley (2000). Software Toolkit for Aerospace Systems Diagnostics. In*IEEE Aerospace Conf. Proc. 2000*, Vol. 6, pp. 327-335.
13. J. Rasmussen (1993). Diagnostic reasoning in action. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**, 4: 981-991.
14. M. Sampath (2001). A Hybrid Approach to Failure Diagnosis in Industrial Systems. In *Proc. of the ACC 2001*, pp. 2077-2082.
15. L. Yan, and D. Miller (2000). General statistical inference for discrete and mixed spaces by an approximate application of the maximum entropy principle. *IEEE Trans. On Neural Networks*, **11** (3): 558-573.
16. Y. Zhang, and A. Mackworth (1995). Constraint nets: a semantic model for hybrid dynamic systems. *Theoretical Computer Science* 138: 211-239.