An interface agent for network supervision

Babak Esfandiari (LIRMM)
Gilles Deflandre (France Telecom - CNET)
Joël Quinqueton (INRIA-LIRMM)
LIRMM - 161 rue Ada F-34392 Montpellier Cedex 5
France Telecom - CNET
2 avenue Pierre Marzin Technopole Anticipa F-22307 Lannion Cedex
esfandia@lune.lirmm.fr, jq@lirmm.fr
deflandr@lannion.cnet.fr

Keywords: Machine Learning, Interface Agents, Network Management, Chronicles.

1 Introduction

The aim of the "Réseau Futé (Smart Net)" project is to introduce AI techniques (such as machine learning and multi-agent systems) in network management and supervision in order to help the processing of the large volume of alarms and various event notifications received by network management platforms. Actually, many of these alarms prove to have a user-depending utility and have to be filtered. Other events become meaningful when associated to their context, which partly consists in the previous and following events. The dates of detection of those events may vary depending on the traffic.

Therefore time has to be explicitly taken in account. We have chosen the Chronicle model [Gha94] in order to incorporate temporal reasoning in our experimental platform. Thus some of the tasks (alarm filtering, log recording, fault detection...) can be automated via a chronicle recognition system [Dou94], letting the supervision operator focus on more important tasks. Although it is possible to enter the chronicles initially, we will assume that we do not have a complete knowledge of the network, and that the managed domain can evolve quickly.

Therefore "on-line" knowledge acquisition seems to be a good solution. We have been mainly inspired by Pattie Maes's Interface Agents [MK93], where an agent learns by "looking over the shoulder" of the human operator. The association of the chronicle recognition system and the chronicle acquisition system can eventually provide us a true intelligent assistant to network management and supervision.

In order to test our ideas, we have implemented a simplified simulation of a network management platform which respects network management standards (GDMO and CMIS) and encloses our assistant. The correct use of CMIS-primitives could eventually lead us to a true cooperation between network management entities and assistants.

2 Intelligent assistance to a network supervision operator

2.1 The need for an explicit temporal reasoning

Temporal knowledge is required for reasoning on events, actions and change [Gha94], in order to model facts such as : precedence, overlapping, simultaneity between events. While "numerical" approaches based on Operations Research are not adequate for symbolic reasoning, classical and modal logic approaches have problems in finding a good balance between expressiveness and algorithmic complexity.

The *chronicle model* proposed by Ghallab is based on two elementary types of formulaes taken from the reified temporal logic : *events* and *holds*.

- a "hold" expresses that some ground domain attribute holds over some interval, for instance : *Hold (position (robot1, docking-site), (t5, t6))*
- an "event" specifies a discrete change of the value of an attribute, for instance : *Event (state (switch): (off, on), t8)*

A chronicle model is a set of *event patterns* and temporal constraints between them and with respect to a context specified by *hold* assertions. If some observed events match the event patterns, and if their occurence dates meet the specified constraints within its context, then an instance of this chronicle occurs. Here is an example of a chronicle taken from [Gha94] :

```
Chronicle RobotLoadMachine {
event (Robot: (outRoom, inRoom), e1);
event (Robot: (inRoom, outRoom), e4);
event (MachineInput: (UnLoaded, Loaded), e2);
event (Machine: (Stopped, Running), e3);
e1 < e2;
1' ≤ e3 - e2 ≤ 6';
3' ≤ e4 - e2 ≤ 5';
hold (Machine: Running, (e2, e2));
hold (SafetyConditions: True, (e1, e4));
when recognized {report ''Successful load''; }}
```

Low-complexity realtime chronicle recognition [Dou94] is processed in several steps :

- Transform "holds" into "forbidden events", *i.e.* an event should not change the value of the "held" attribute within the duration of the "hold".
- Possibly create a new instance of a possible chronicle and update (in fact this is always a restriction as time goes by) the *window of relevance* (acceptable time intervals for the expected events in order to complete a chronicle pattern) of all possible chronicles when a new event has been observed.

- Detect "deadlines" and occurence of "forbidden events". In these cases, the corresponding chronicle will be removed from the list of the possible chronicles.
- Trigger off the action corresponding to the completed chronicle.

2.2 The assistant's structure

Our machine assistant is mainly inspired by Pattie Maes's Interface Agents [MK93]. An Interface Agent is "a computer program that employs Artificial Intelligence techniques in order to provide assistance to a user dealing with a particular computer application. Such agents learn by 'watching over the shoulder' of the user and detecting patterns and regularities in the user's behaviour". Since we had to deal with realtime aspects and time was a very important parameter, we chose to manipulate chronicles.

As shown in figure 1, our assistant has two main components :

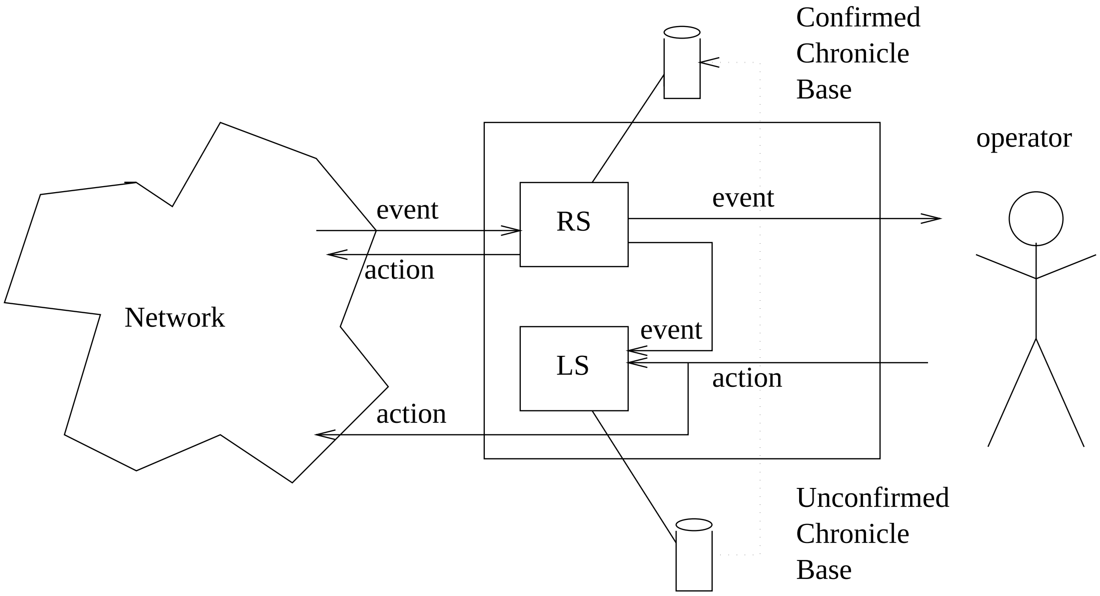
- The Chronicle Recognition System (RS), which goal is to receive dated event notifications (such as alarms) and try to match them with chronicles stored in the so-called Confirmed Chronicle Base. Whenever a chronicle has been recognized, the RS processes the corresponding action (such as filtering, fault diagnosis...). If the received events do not match any chronicle, it is up to the supervision operator to take a decision. For more details about the Recognition System, see [Dou94].
- The Learning System (LS), which goal is to watch "over the shoulder" of the supervision operator when he takes a decision, in order to feed the Recognition System with new chronicles. New chronicles are stored in a so-called Unconfirmed Chronicle Base before being "mature" enough in order to be transferred to the Confirmed Chronicle Base (i.e. confirmed). In the following subsections, we describe more precisely the Learning System.

2.3 Learnability issues

In [EN96], we show that given certain assumptions (no "overlaps" between the received sequences of events, an action appearing only in one chronicle), chronicle learning can be viewed as learning monotonous monomials, and therefore is PAC-learnable. But when releasing those assumptions, there is a need for interaction with an oracle. The more complex and numerous are the interactions and the oracles, the more efficient can be the learning task. Of course, a certain degree of cooperation can be expected from the supervision operator, but since we do not want to disturb him too much (do not forget that our goal is to simplify his task !), we have to restrict ourselves to a heuristic, except if we can interact with other oracles, for instance other assistants.

2.4 The learning algorithm

Before beginning the description of our system, let us assume that the set of events and actions is finite and "relatively small", in order to generate a reasonable amount of chronicles with enough genericity. We will furthermore consider a particular action



RS : Recognition System

LS: Learning System

Figure 1: The assistant's structure

called "silence", which does not correspond to any actions of the supervision operator, but which is useful to generate particular chronicles : filters.

Learnt chronicles will consist in a set of events terminated by an action. We will see later how we deal with temporal constraints and "holds".

As the supervision operator has to interact with the Learning System (for instance to confirm chronicles), it is important for him to understand how our system learns. We believe that incremental learning is easier to follow by a human operator. Furthermore, if the learning algorithm is fast enough, it can be used in realtime, just like the Recognition System, thus enabling better cooperation.

Chronicles are (1) created, (2) evaluated and (3) confirmed. The following section describes these steps.

2.4.1 The chronicle creation

A chronicle is created in two cases :

- When the supervision operator triggers an action : the created chronicle is the set of the received events before the action, plus the action itself.
- When nothing happens during a given time interval : neither received events nor actions. The created chronicle (which will be used as a filter) is the set of the received events since the previous action, followed by the action "silence".

2.4.2 The chronicle evaluation

Now we have to know whether the created chronicle is worth being added to the Unconfirmed Chronicle Base.

First we need to define two operators : the inclusion ($\subset$) and the subtraction ($\setminus$) between two chronicles. A chronicle A is included in chronicle B iff their actions are equal and A's sequence of events is a "subword" of B's sequence of events. Dates of occurence of the events are not compared. If A $\subset$ B, then we can subtract A from B by removing A's events from B's sequence of events and replacing B's action with the action "silence". The result is what we call a "filter". Of course the result of a subtraction can depend on how to select the subword in B. An easy way is to select the first occurence of each event. For more "realism", it can be interesting to minimize the date differences, but this can be too costly, except if we do not want to obtain necessarily the "best" subword.

Another operation is the creation of temporal constraints. By increasing time intervals, we can somehow "generalize" chronicles by accepting chronicles with the same sequence of events and with different time points.

Here is the evaluation of a new chronicle C, created following the previous step, given his action α :

Algorithm 1: EVALUATE(C)

% C is a chronicle with an action α
begin
  $Trust(C) := 1$
  for *each* $C' \in$ *the CB so that* $C'(action) = \alpha$ **do**
    switch C **do**
      case $C \subseteq C'$: exit
      case $C \supset C'$: create-and-evaluate-filter $C'' := C \setminus C'$
      case $C \neq C'$: continue
    endswitch
  endfor
  for *each* $C' \in$ *the UCB so that* $C'(action) = \alpha$ **do**
    switch C **do**
      case $C \subset C'$
        add C to the UCB; $Trust(C) := Trust(C') + 1$
        widen C time intervals given C'
        remove C' from the UCB; create-and-evaluate-filter $C'' := C' \setminus C$
      case $C \supseteq C'$
        create-and-evaluate-filter $C'' := C \setminus C'$; widen C' time intervals given C
        $Trust(C')$++
      case $C \neq C'$: continue
    endswitch
  endfor
  add (C) to the UCB
end

Filters, like C'', created during the evaluation algorithm must also be evaluated similarly. However, their evaluation is easier, as less cases are possible, and since the results of a subtraction do not need to be evaluated again.

2.4.3 The chronicle confirmation

Now that we have put chronicles in the Unconfirmed Chronicle Base, the question is : when can we transfer those to the Confirmed Chronicle Base ? We have basically two ways of doing this :

- When the "trust" in a chronicle reaches a certain threshold, it can be automatically confirmed.
- The operator can confirm manually a chronicle by simply "clicking" on it.

Note that there is a need for a coherence maintenance system in the Confirmed Chronicle base. Indeed, if the system meets cases where the same sequences of events lead to different actions, it has to detect them. This is done by comparing each new chronicle to those stored in the Confirmed Chronicle Base and trying to find mutual inclusions (this time with different actions). When incoherence is detected, the user has the choice to cancel the confirmation, or to confirm the chronicle anyway (sometimes events can trigger several chronicles). Coherence maintenance is not important concerning the Unconfirmed Chronicle base, since those chronicles are not used yet in the recognition system. But this could also help creating "holds": the events obtained by the subtraction of two chronicles with different actions would correspond to "holds".

3 Experimentation and First Results

The learning system has been integrated in our experimental and simple network management platform, MAGENTA [EPQD96], which is roughly compatible with ISO and CCITT specifications, namely GDMO [ISO90b], CMIS [ISO90a], and the Manager-Agent model. MAGENTA agents can, by browsing network resources, answer queries sent by the MAGENTA manager. These agents can also spontaneously send notifications to the manager when some specified events occur in the network. These notifications will be considered as events by our assistant. Picture 2 shows MAGENTA's interface for the assistant, which is composed of the following parts:

1. The "notification" window displays the received events;
2. The "query" window enables the sending of queries and actions to MAGENTA agents by typing the query or the action and designating the agent;
3. The "chronicle bases" window displays both the UCB and the CB and therefore shows the Learning System's activity. It is possible to "confirm" a chronicle of the UCB and "edit" any chronicle. The interface should be more ergonomic in the future in order to be easily understood and accepted by the user. For instance, chronicles could be displayed in a graphical way.
4. The lower right part of the interface is dedicated to the Chronicle Recognition System. We have not connected it to our Learning system yet.

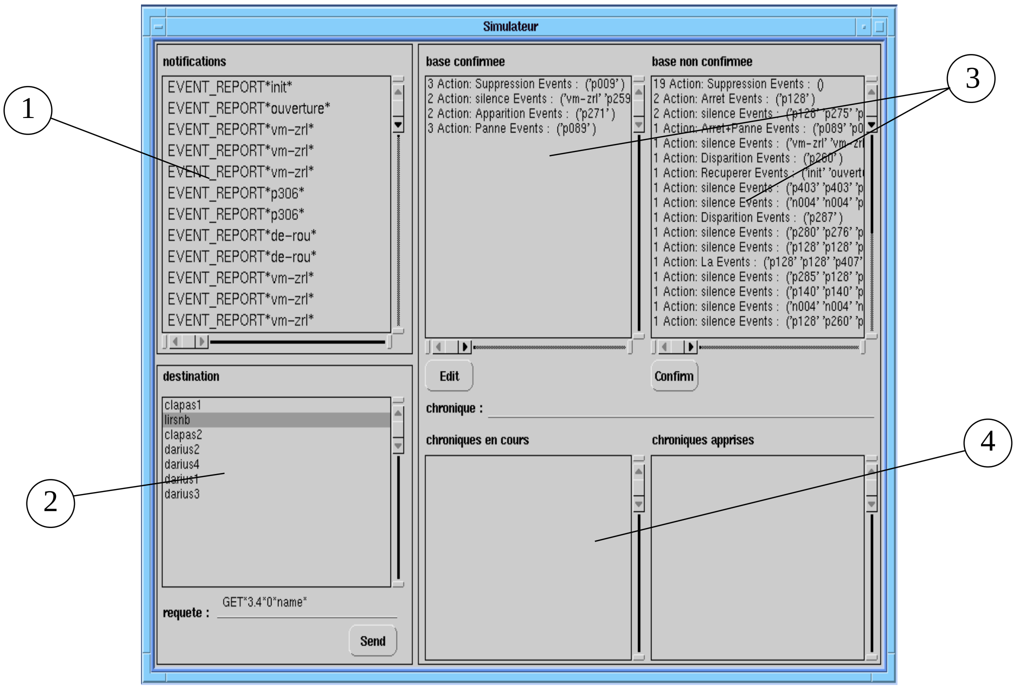


Figure 2: The assistant's interface

The assistant has been tested with supervisory data of a french packet-switched data network. Its components send a lot of notifications to the supervisor dealing with their state and the state of related components. Thus, whenever a failure occurs, lots of notifications are brought to the supervisor. An expert system has been developped to identify failures out of these notification logs. In order to define chronicles, we have considered the notifications as events and the conclusions of the expert system as actions. Then, we have tested our system with a small sample of events corresponding to thirty hours of diagnosis. The behaviour of the assistant on this sample is promising for it proved to learn quite well chronicles containing up to 3 events with a good relative trust. However, the presence of "overlaps" is a problem, since it can cause an over-reduction of some chronicles. We have therefore imagined to bufferize the k last received events independently of the triggered actions. But then our algorithm had to be adapted to those chronicles: instead of trying to find inclusions between chronicles (less probable), we chose to find the longest common subword between two chronicles triggering the same action. Results obtained with this algorithm deal much better with overlapping, but on the other hand we cannot store anymore two different chronicles triggering the same action (searching the longest common subword between them could lead to empty chronicles).

4 Related works

Building interface agents (sometimes also called software agents) is a very active research field, mostly due to the development of the World Wide Web. The reader shall read for instance [MK93] or [SH93]. However, we have not so far found papers dealing explicitly with temporal knowledge. Learnability issues are also seldom dealt with.

In network management and supervision, many intelligent architectures, such as expert systems, have been designed to help the operator's tasks, taking in account realtime aspects [Gai91] [GH92]. The authors only mention the addition of learning mechanisms as a promising perspective.

5 Conclusion and Perspectives

In this paper we have presented a chronicle-learning interface agent adapted to network supervision. So far the experimentation proved very promising. Now we are planning to make several assistants cooperate throughout the network, each assistant being an oracle for the others. This could accelerate the learning curve, making for instance a new assistant benefit from other assistants' experience.

Acknowledgements
This work is part of the "Réseau Futé" project, which is supported by France Telecom - CNET under contract no 93 1B 141/142/143 5115.

References

[Dou94] Christophe Dousson. *Suivi d'évolutions et reconnaissance de chroniques*. Thèse d'université, Université Paul Sabatier, Toulouse, LAAS, Septembre 1994.

[EN96] Babak Esfandiari and Richard Nock. Chronicle learning for assistance to a network supervision operator. Research Report 96028, Lirmm, Janvier 1996.

[EPQD96] Babak Esfandiari, Michel Plu, Joël Quinqueton, and Gilles Deflandre. A multi-agent perspective for assistance to a network supervision operator. In *MAAMAW poster session*, 1996.

[Gai91] D. Gaiti. *L'utilisation des techniques de l'intelligence artificielle pour la gestion des réseaux*. Thèse d'université, Université Paris 6, 1991.

[GH92] Francisco J. Garijo and Donn Hoffman. A multi-agent architecture for operation and maintenance of telecommunication networks. In Jean Paul Haton, editor, *Proceedings of the 12th Int. Avignon Conference*, pages 427–436. EC2 and AFIA, EC2, Paris, Juin 1992.

[Gha94] Malik Ghallab. Past and future chronicles for supervision and planning. In Jean Paul Haton, editor, *Proceedings of the 14th Int. Avignon Conference*, pages 23–34. EC2 and AFIA, EC2, Paris, Juin 1994.

[ISO90a] ISO/DIS 9595. *Information Technology - Open Systems Interconnection - Common Management Information Service Definition*, Janvier 1990. 2nd DP N3070.

[ISO90b] ISO/DP 10165-4. *Information Technology - Structure of Management Information - Part 4. Guidelines for the Definition of Managed Objects*, Juin 1990.

[MK93] Pattie Maes and Robyn Kozierok. Learning interface agents. In *Proceedings of the 11th Nat Conf on Artificial Intelligence*. AAAI, MIT-Press/AAAI-Press, 1993.

[SH93] J.C. Schlimmer and L.A. Hermens. Software agents: Completing patterns and constructing user interfaces. *Journal of Applied Intelligence Research*, 1:61–89, 1993.