10

Chronicle Learning and Agent oriented techniques for network management and supervision

Joël Quinqueton, Babak Esfandiari, Richard Nock
LIRMM and INRIA
161 rue Ada 34392 Montpellier Cedex 5, France, +33(4)67418532, Fax +33(4)67418500 {jq,nock}@lirmm.fr

Abstract

This paper presents some aspects of the "Réseau futé (Smart Net)" project, whose aim is to introduce Artificial Intelligence 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 events notifications received by network management operators. We provide experimental and theoretical results on learning patterns called chronicles in order to design a machine assistant to network operators. Theoretical results investigate different levels of help that could be brought by the operator to the assistant. The tests were performed in two distinct realworld situations. They showed the circumstances under which chronicle learning is possible without the help of the operator or another assistant.

Keywords

Interface agents, Chronicles, Machine Learning, Network management, Learnability

1 INTRODUCTION

The aim of the "Réseau Futé (Smart Net)" (Esfandiari, Nock & Quinqueton 1996) project is to introduce Artificial Intelligence 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, which dates of detection may vary depending on the traffic. Therefore time has to be explicitly taken into account. We have chosen the Chronicle model (Ghallab 1994) 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 (Dousson 1994), letting the supervision operator focus on more important tasks. Although it is possible to have a

Intelligent Networks and Intelligence in Networks D. Gaiti (Ed.)
Published by Chapman & Hall

model-based approach, we will assume that we do not have a complete knowledge of the network, and that the model 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 (Maes & Kozierok 1993), 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. But chronicle learning can raise theoretical problems, and we have to know under which assumptions it is reasonable to build such a system. For example, (Dousson 1994) advocates for the dialog between the learner and the experts. This paper brings some theoretical results showing how much such a dialog can be useful, and provide some algorithms which have been tested so far. We have implemented a simplified simulation of a network management platform which respects network management standards (GDMO and CMIS) and encloses our assistant. The algorithms present in some cases the desirable properties (Dousson 1994) to provide neither too specific, nor too general chronicles.
Section §2 presents some definitions refering to chronicles; section §3 presents the assistant's structure. Positive and negative results for chronicle learning are investigated in section §4, followed in section §5 by the algorithms and the practical results that were obtained.

2 DEFINITIONS

Temporal knowledge is required for reasoning on events, actions and change (Ghallab 1994), 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 (Ghallab 1994) :

```
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''; }}
```

Realtime (and therefore with low complexity) chronicle recognition (Dousson 1994) 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.

3 THE ASSISTANT'S STRUCTURE

Our machine assistant is mainly inspired by Pattie Maes's Interface Agents (Maes & Kozierok 1993). 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 (Dousson 1994).
- 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

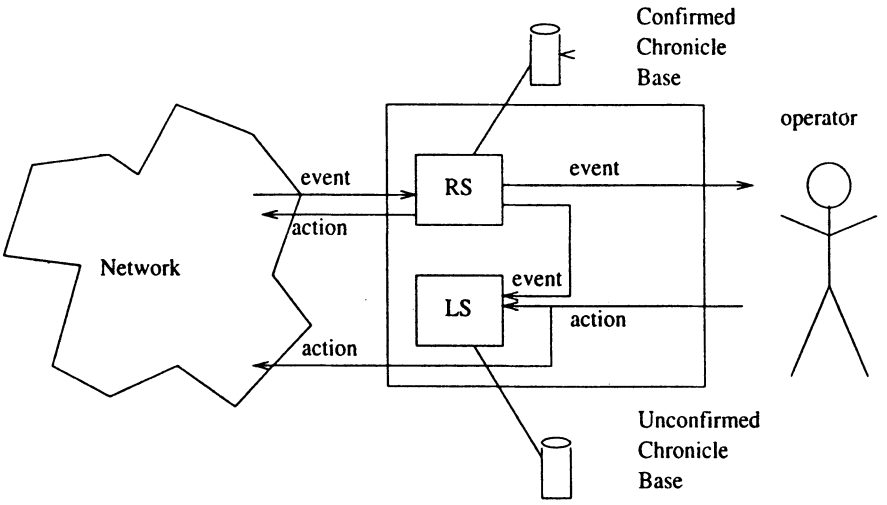

Figure 1 The assistant's structure

System with new chronicles. New chronicles are stored in a so-called Unconfirmed Chronicle Base before being "mature" enough in order to be confirmed (*i.e.* transferred to the Confirmed Chronicle Base). In this paper, we shall describe more precisely this component.

4 LEARNABILITY AND CHRONICLES

Above all, we shall use in all that follows a simpler notation for chronicles: *e.g.* the chronicle

Chronicle X {
event (a_{t_1}, t_1); ⋯
event (a_{t_i}, t_i);
$t_2 - t_1 = \delta_1$; ⋯
$t_i - t_{i-1} = \delta_{i-1}$
when recognized {**report** ''α_j''; }}

is rewritten as $\{(a_{t_1}, t_1), ..., (a_{t_i}, t_i); \delta_1 = t_2 - t_1, ..., \delta_{i-1}\} \Rightarrow \alpha_j$, where $a = \{a_1, ..., a_n\}$ is a set of events, and α_j belongs to a set of actions $\alpha = \{\alpha_1, ..., \alpha_{M_1}\}$ which depend

on a. As the reader shall remark, we perform some limitations on chronicles, that shall not disturb our study: first of all, *we remove the "holds" on chronicles.* Indeed, they are not that more complicated to manage than "events" or time constraints. Our modelling takes into account that any action is triggered off by some events (i) given some ordering of them, (ii) given some time constraints, and (iii) provided each event can appear many times.

In the discussion of some of the results we give later on learning chronicles, we take into account a type of limitations that are framework dependant. We emphasize on them, since they can be very useful to accelerate the learning task on our main problem: network supervision. In particular, they include the human operator limitations: chronicles are supposed to help him along, simulating, why not, his actions on the network. Any human operator, be him as efficient as possible, cannot recall (i) events that are too spaced out, (ii) the number of times some event has appeared, beyond some threshold, (iii) neither the exact time, nor sufficiently good approximations, that separates two events triggering off an action. And these remarks are all the more valid as the operator generally receives a treamount of events. The first two approximations we shall make are therefore the following:

1. Time is discretized. Any two consecutive events triggering off an action are spaced out by a time measured in the set $\{0, 1, ..., M_3\}$, for some constant M_3.
2. In order to trigger off an action, any event needs at most multiple appearances bounded by some constant M_2.

Therefore, we can rewrite any chronicle involving action α_j as a formula $(m, t) \Rightarrow \alpha_j$, where $m \in \{1, *\}^{nM_2}$ and $t \in \{-M_3, ..., M_3, *\}^{2C^2_{nM_2}}$, where "*" means no value specified, and $C^2_{nM_2}$ is the binomial coefficient $C^k_n = n!/(k!(n-k)!)$. The first part, m, is a monotonous Boolean formula, showing whether an event has to appear for triggering off α_j, taking in account of the fact that there are n distinct events, each one being able to appear M_2 times. The second part, t, is a vector representing the time constraints δ's, be they equality or inequalities. We can even model a chronicle as a complete monotonous Boolean formula, by replacing in t any δ by a word of constant size on the alphabet $\{1; *\}$. Therefore it becomes possible to look for the translation of any positive result concerning Boolean formulas into positive results for learning chronicles.
What follows is composed of three parts. The first one presents models for learning chronicles; it is followed by positive results for chronicle learning. The last section presents negative results.

4.1 Learning chronicles

We suppose that the assistant has access to *oracles*, that supply events, actions and/or messages ("Yes", "No") according to some fixed protocol. We present them from the

most passive of them, to the most active ones, for which there can be interactions between assistant and oracles. We have on purpose searched to make the oracles behave as simply as possible. In particular, we did not search to adapt the complex behaviour of experts recommended by (Dousson 1994).

- The most simple of these oracles is entirely passive, and simply supplies on-line streams of events from a and actions from α, according to their appearance on the network. We refer to him as PASSIVE(α).
- The second oracle behaves as PASSIVE(α), except that its output does not present overlapping. PASSIVE$^S(\alpha)$ supplies "examples", that is, couples (stream of events, action) that can be viewed as chronicles subsumed by the chronicles to model. The learner knows that the events provided are exactly those that have triggered off α; PASSIVE$^S(\alpha)$ therefore can be viewed as providing a "one-way help", from the operator to the assistant (the assistant knowing that there is no overlapping). From this point of view, PASSIVE$^S(\alpha)$ is an interesting intermediate between PASSIVE(α), where no explicit help exists between the assistant and the operator, and the following oracles, that represent "two-way helps".
- ACTIVE$_{MQ}(\alpha)$ takes for parameter some set of events from a and some action from α, and is answered "Yes" or "No", depending on the action can be triggered off by this set of events.
- ACTIVE$_{EQ}(\alpha)$ takes for parameter a chronicle (or a set of chronicles) involving an action from α, and is answered (by the operator) either "Yes" if it is judged as correct (and the chronicle is added to the Confirmed Base), or "No", and in that latter case, a stream of events is provided by the operator, for which the chronicle fails.
- ACTIVE$_{GQ}(\alpha)$ takes for parameter a chronicle (or a set of chronicles) involving an action from α, and is answered (by the operator) "Yes" iff any stream of events triggering off the action from α would be triggered off by the chronicle (or one of the set). Otherwise, a stream of events is returned, that can trigger off the action, for which the chronicle fails. This can be viewed as a "generality" query.

Concerning the two passive oracles, we suppose that events (or the examples) are provided according to some unknown, but fixed probability distribution D. In the end of this section we examine a relaxation on the main flaw of this strong hypothesis, which is to take into account the evolution of D through time. The assistant is therefore supplied with streams of type $s_{i_1}, \alpha_{i_1}, s_{i_2}, \alpha_{i_2}, \ldots$ indexed by time points. s_{i_k} is a stream of events, having a certain probability to appear, that triggers off α_{i_k}. The aim of the assistant is to guarantee, with a certain confidence, that, from a fixed and reasonable time, he shall trigger off the actions from α approximatively "at the same time" they are triggered off in the stream he receives from the passive oracle(s). This means that, if the operator is removed (when we can consider that the learning task for α has ended, see figure 1), and if, say, the assistant re! ceives a stream $s_{i_t}, s_{i_{t+1}}, \ldots$, then its "failure" probability will be no more than, say $0 < \epsilon < 1$. Which means that, with probability $1 - \epsilon$, α_{i_t} will be triggered off after the appearance of the last event of s_{i_t}, but not after the appearance of the first event of $s_{i_{t+1}}$.

We define two models of learnability, using passive and active oracles, that can be related to the classical PAC-model and to the Exact-Identification-model. In order not to laden our notations, we keep these two names. We note as $c(\alpha)$ the set of chronicles to learn, that trigger off the actions from α, $|c(\alpha)|$ the size of the writing of $c(\alpha)$, and Ω some oracle(s) belonging to the previously enumerated set.

Definition 1 *Let $a = \{a_1, ..., a_n\}$ some set of events, and $\alpha = \{\alpha_1, ..., \alpha_{M_1}\}$ some set of actions to learn, depending on a. We shall say that α is PAC-learnable using Ω if there exists a learning algorithm L using Ω and a polynomial $p(.,.,.,.)$ such that, for any $0 < \epsilon, \delta < 1$ (resp. accuracy and confidence parameters), for any D, after a time bounded by $p(n, \frac{1}{\epsilon}, \frac{1}{\delta}, |c(\alpha)|)$, L puts a set of chronicles $c'(\alpha)$ in the confirmed chronicle base such that*

$$P(P_D(c'(\alpha)\ fails) > \epsilon) < \delta$$

Definition 2 *Let $a = \{a_1, ..., a_n\}$ some set of events, and $\alpha = \{\alpha_1, ..., \alpha_{M_1}\}$ some set of actions to learn, depending on a. We shall say that α is Exactly-Identifiable using Ω if there exists a learning algorithm L using Ω and a polynomial $p(.,.)$ such that, after a time bounded by $p(n, |c(\alpha)|)$, L puts a set of chronicles $c'(\alpha)$ in the confirmed chronicle base equivalent to $c(\alpha)$.*

In the case of the PAC-model, we always have PASSIVE(α)$\subseteq \Omega$ or PASSIVE$^S(\alpha)\subseteq \Omega$; in the case of the Exact-Identification-model, we always have ACTIVE$_{MQ}(\alpha)\subseteq \Omega$. As the reader shall remark (§5), the algorithm proposed for the assistant uses a threshold to put the learnt chronicles in the confirmed base, corresponding to a number of times the actions of the corresponding chronicle were triggered off. Under some assumptions, the algorithm can be viewed as learning monotonous monomials from positive examples only ((Kearns 1989, Kearns & Vazirani 1994)). For the sake of clarity, if we suppose that $|\alpha| = 1$, the threshold t_α for putting α in the confirmed base can be calculated in the same way as (Rivest 1987), theorem 4. If α is triggered by k (constant) chronicles, depending on the fact that the size of a chronicle can be considered as unfixed or constant, the dependance on n in t_α comes from $\mathcal{O}(n)$ to $\mathcal{O}(ln(n))$. The logarithmic dependance in n is important, as n can be very large, and therefore the human factor can be of great help for the threshold calculation.
If we consider D, the main problem it raises is that many set of events might have a probability of appearing varying through time, because of the evolution of the network characteristics. In the other hand, this is not the case for all events: some of them (such as natural events, or unpredictable accidents) may be considered as having a constant appearing probability. Thus, a better framework would allow such variations of D. For example, an unoptimal model would allow the "weight" of any event e at time t, $w_t(e)$, to behave as $w_t(e) \leq \frac{1}{\beta} w_{t'}(e), \forall t' \leq t$. In that case, the learning model remains the same, except that L is required to learn for any such fixed β, and the alloted time is required to have polynomial dependence in $\frac{1}{\beta}$. The fixation of β as small as desired allows to keep closer to reality. The drawback of

fixing β as small as desired is of course that the threshold for any chronicle will be much higher: in the case of PASSIVE$^S(\alpha)$,its dependance in β is $\mathcal{O}\left(\frac{1}{\beta}\right)$ (Proof similar as (Rivest 1987), theorem 4).

4.2 Positive results for learning chronicles

Proposition 1 *If any action from the set α can be supposed to appear in a single chronicle, then*

1. *α is PAC-learnable using PASSIVE$^S(\alpha)$.*
2. *α is PAC-learnable using ACTIVE$_{MQ}(\alpha)$ and PASSIVE(α), and the chronicles built are at least as accurate as those of 1.*
3. *α is Exactly-Identifiable using ACTIVE$_{MQ}(\alpha)$ and ACTIVE$_{GQ}(\alpha)$.*

Parts 1–2 of the previous property raise the problem of learning without ACTIVE$_{MQ}(\alpha)$ or PASSIVE$^S(\alpha)$, thus using only PASSIVE(α), when there exists overlapping between chronicles. Such a problem seems hardly manageable since, provided there are only two disjoint chronicles, we can create some distribution over the events that forces the two chronicles to trigger off almost at the same time, and that forces nealry all their events to appear before any triggering off of one of the two actions. By this, we force at least one chronicle to appear to the assistant much bigger than it should be; this tends to show the usefulness of PASSIVE$^S(\alpha)$ or ACTIVE$_{MQ}(\alpha)$. It is worthwile remarking that even if PAC-learning is not manageable, we could find heuristics in some particular cases, satisfying from an algorithmic point of view. Denote as a "size-k"-overlapping an overlapping where for any two consecutive actions α_i and α_j (triggered off in this order), at most k events happening between the two actions do not participate to the triggering off of α_j. In that case, without learning the chronicles according to definition 1, we could come close to good chronicles, in a satisfying way (possibly closer than $2k$ from the chronicles we would have learnt without overlapping). Such a procedure would be all the more satisfying if the chronicles could be considered as having a large size.
Concerning the cases where an action can be triggered off by multiple chronicles, the learning problem is at least as difficult as for k-term-DNF. It seems that ACTIVE$_{MQ}(\alpha)$ and ACTIVE$_{GQ}(\alpha)$ are not sufficient to Exactly-Identify such multiple chronicles, particularly for the exact-identification of the time constraints. The exact identification apparently needs another oracle quite similar to ACTIVE$_{GQ}(\alpha)$, and is therefore very costly with respect to the operator.

4.3 Hardness results for learning chronicles

The purpose of this section is to show how useful is the operator (in the form of ACTIVE$_{MQ}(\alpha)$) for the learning task. Namely, even if the chronicles are simple,

and already under simple situations, even using different types of oracles, we show that trying to identify them as exactly as possible, or even sometimes learning them, can be very hard. As the reader might have remarked before, some results turn out to be positive when the operator can help by $\text{ACTIVE}_{MQ}(\alpha)$. Exact identification is a stronger task than learning; however, this can be useful to achieve better understanding of the actions; furthermore, the application algorithms are generally implicitly constructed quite as optimization algorithms than learning algorithms.
In order to prove negative results already in simple cases, the propositions of this section make the following assumptions on the learning framework:

1. The events concern only a single action, that is, $|\alpha| = 1$.
2. Calls to $\text{PASSIVE}^S(\alpha)$, $\text{ACTIVE}_{EQ}(\alpha)$ and $\text{ACTIVE}_{GQ}(\alpha)$ are authorized.

In order to formulate our optimization result, we fix as $k-$chronicles the set of chronicles having size not bigger than k. RP denotes the class of polynomial-time randomized algorithms as defined by (Balcazar, Diaz & Gabarro 1988).

Proposition 2 *Under the learning framework of assumptions 1-2, if α only belongs to one chronicle, unless $RP = NP$, no PAC-learning algorithm can learn $c(\alpha)$ by $|c(\alpha)|$-chronicle.*

(Proof made by reduction from the Set-Cover problem (Garey & Johnson 1979)).

Proposition 3 *Under the learning framework of assumptions 1-2, even if the events appear at different times each, with constant time between events and actions, if α is triggered off by only three sets of events, α is not PAC-learnable unless $RP = NP$.*

(Proof made by reduction from Graph-3-Colorability (Garey & Johnson 1979)).

5 PRACTICAL RESULTS: THE LEARNING SYSTEM

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

Definition 3 *A filter is a chronicle* $\{(a_{t_1}, t_1), (a_{t_2}, t_2)...(a_{t_i}, t_i); \delta_1 = t_2 - t_1, \delta_2, ..., \delta_{i-1}\} \Rightarrow silence$

Filters are used to decrease the sum of the events displayed to the operator: when the corresponding events are received, the system simply ignores them. Given that many different sequences of events can lead to the action "silence", it is hard to learn

them without using the operator or the assistants' help (see §4). 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. The chronicle learning is processed in three steps :

- The chronicle creation
- The chronicle evaluation
- The chronicle confirmation

5.1 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".

In some cases, the operator triggers several actions in a row, without waiting for the reception of new events. This can have several meanings:

1. these actions all correspond to the whole sequence of the received events;
2. each action corresponds to a subsequence of the received events;
3. a combination of the two previous possibilities.

In the actual system, we have assumed that we only have to face the first possibility. The sequence of actions is therefore considered as one single action which is the concatenation of the sequence. The second possibility corresponds in fact to the more general problem of overlapping chronicles, which is hard to manage. This issue has been discussed in section §4.2. A possible heuristic would be to bufferize the k last received events independently of the triggered actions. When the sum of the received events reaches a certain threshold T which models the memory capacity of the operator, only the T last events are buffered (this also helps to satisfy learnability constraints expressed in section §4.1). Obviously, when the action is triggered by the Recognition System, the corresponding events are deleted, since the corresponding chronicle has already been learnt.

5.2 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. Obviously, we only create temporal constraints between two consecutive events, since otherwise trying to create some kind of adequate "constraint-tree" (the root being the first event) would be too costly and not necessarily useful. Remember that we try to preserve a realtime learning algorithm ! Here is the evaluation of a new chronicle C, created following the previous step, given his action a :

C is a chronicle with an action a
Begin
  Trust(C) := 1 ;
  For each $C' \in CB$ so that $C'(action) = a$
    Switch C
      Case $C \subseteq C'$: exit ;
      Case $C \supset C'$: create-and-evaluate-filter
        $C'' := C \setminus C'$; exit ;
      Otherwise : continue ;
  For each $C' \in$ the UCB so that $C'(action) = a$
    Switch C
      Case $C \subset C'$ add C to the UCB
        $Trust(C) := Trust(C') + 1$;
        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 interval ;
        Trust(C')++ ;
      Otherwise : continue ;
  add C to the UCB ;
End

Filters, like C'', created during the evaluation algorithm must also be evaluated. However, their evaluation is easier, since we do not need to create other chronicles during the evaluation. So far we have favoured the creation of "long" filters, *i.e.* when detecting inclusions between filters, we keep the longer. The opposite point of view can also be considered. It must be noted that the above algorithms may be a valid learning algorithm when no overlapping of events occurs. However, when it is not the case, it becomes a heuristic.

5.3 The Chronicle Confirmation

Now that we have put chronicles in the Unconfirmed Chronicle Base, the question is : when can we put those in 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. Here, the operator plays approximatively the role of the ACTIVE_{EQ} oracle. To play it completely, it would be necessary for him to bring the assistant a case where the chronicle fails.

Note that there is a need for a coherence maintenance system in the Confirmed Chronicle base. Indeed, if the system encounters cases where the same sequence of events leads to different actions, it has to detect them. This can be done by comparing each new chronicle to those stored in the Chronicle Base and trying to find inclusions (this time with different actions). 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".

5.4 Experimentation and First Results

The learning system has been integrated in our experimental and simple network management platform, MAGENTA (Esfandiari, Deflandre, Quinqueton & Dony 1996), which is roughly compatible with ISO and CCITT specifications, namely the Manager-Agent model, GDMO (ISO 1990*b*) and CMIS (ISO 1990*a*). MAGENTA agents can answer queries sent by the MAGENTA manager by browsing network resources. 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. Figure 2 shows MAGENTA's interface for the assistant, which is composed of the following parts:

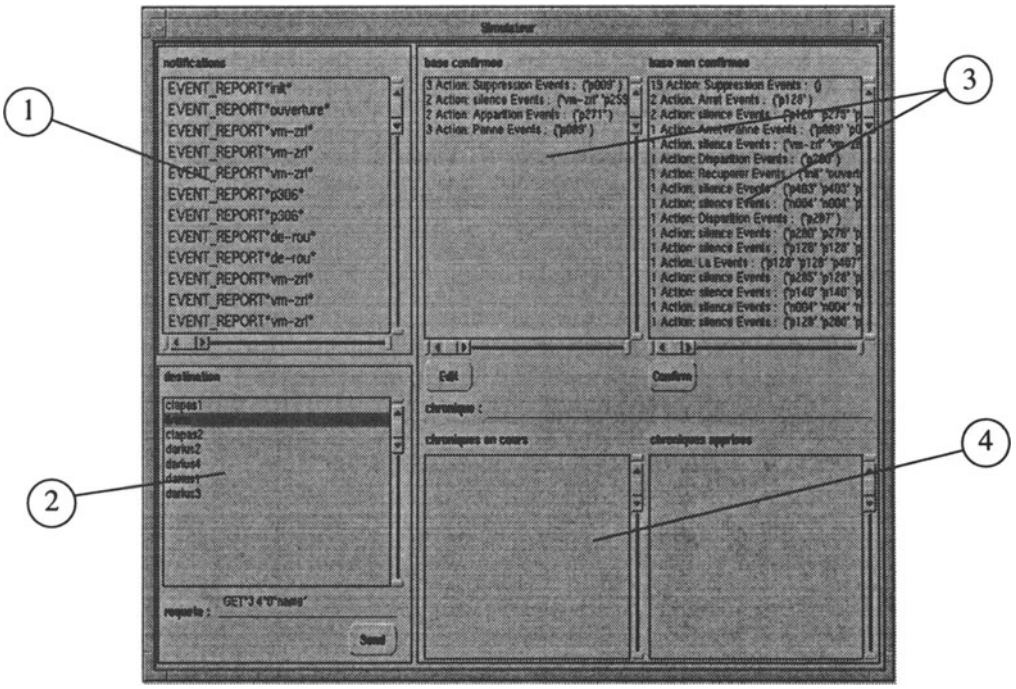

Figure 2 The assistant's interface

1. The "notification" window displays the received events;
2. The "query" window enables the sending of queries to MAGENTA agents by typing the query and designating the agent. These queries can be considered as actions;
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.

The assistant has been tested with data taken from a simulation of a french packet-switched data network. The actions to be triggered off are part of these data (they have been generated by a specific expert system). Results show that with a very small sample of events corresponding to thirty hours of simulation, the assistant behaves quite well, learning five acceptable chronicles out of ten that reached the confirmation threshold. These results are all the more valid as these data contain lots of overlaps, and the actions are not necessarily triggered off by a single sequence of

events: these two cases make the algorithm behave only as a heuristic. However, the learnt filters were not satisfying: they did not even reach the confirmation threshold.

We have also tested the assistant on a problem where we could reasonably suppose that there were no overlaps: the data were generated by the behaviour of a user of a programming environment. Some of the generated chronicles were very pertinent and really corresponded to the desire of the user.

6 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 (Maes & Kozierok 1993) or (Schlimmer & Hermens 1993). 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 (Gaïti 1991) (Garijo & Hoffman 1992). The authors only mention the addition of learning mechanisms as a promising perspective.

7 CONCLUSION AND PERSPECTIVES

In this article, we present a chronicle learning system that has been implemented and tested in a realworld domain: network management. We have also studied a learnability model relevant to our framework and raised theoretical results which show the utility of cooperation between the machine assistant and the operator. These results show also that the construction of chronicles is a difficult task. The main difficulty, which is specific to our framework, is the possibility of overlapping between chronicles, that raises the heuristical character of our algorithms. However, without such overlaps, the algorithm behaves quite well, particularly by preventing excessive generalizations, a desirable property (Dousson 1994). In order to improve our system, it would be interesting to better the management of the overlaps, as well as to quantify the loss in performances created in our heuristic. Moreover, the extension of events to first order formalism could increase the semantical value of the learnt chronicles.

But our results, be they theoretical or practical, showed the importance of the help provided by the operator or other assistants. The next important step in our work will be to interconnect assistants in order to accelerate (or to allow !) the learning task. Thus, the assistants would be able to query each other instead of replacing a treamount of events received by the operator by a treamount of questions.

REFERENCES

Balcazar, J. L., Diaz, J. & Gabarro, J. (1988), *Structural Complexity I*, Springer Ver-

lag.

Dousson, C. (1994), Suivi d'évolutions et reconnaissance de chroniques, University thesis, Université Paul Sabatier, Toulouse, LAAS.

Esfandiari, B., Deflandre, G., Quinqueton, J. & Dony, C. (1996), 'Agent-oriented techniques for network supervision', *Annals of Telecommunications* **51**(9-10), 521–529.

Esfandiari, B., Nock, R. & Quinqueton, J. (1996), Réseau futé: Techniques d'intelligence artificielle distribuée pour la supervision-maintenance des réseaux, Progress Report 5, CNRS/CNET 93 1B 142 Proj 5115.

Gaïti, D. (1991), L'utilisation des techniques de l'intelligence artificielle pour la gestion des réseaux, Thèse d'université, Université Paris 6.

Garey, M. & Johnson, D. (1979), *Computers and Intractability, a guide to the theory of NP-Completeness*, Bell Telephone Laboratories.

Garijo, F. J. & Hoffman, D. (1992), A multi-agent architecture for operation and maintenance of telecommunication networks, *in* J. P. Haton, ed., 'Proceedings of the 12th Int. Avignon Conference', EC2 and AFIA, EC2, Paris, pp. 427–436.

Ghallab, M. (1994), Past and future chronicles for supervision and planning, *in* J. P. Haton, ed., 'Proceedings of the 14th Int. Avignon Conference', EC2 and AFIA, EC2, Paris, pp. 23–34.

ISO (1990*a*), *Information Technology - Open Systems Interconnection - Common Management Information Service Definition*. 2nd DP N3070.

ISO (1990*b*), *Information Technology - Structure of Management Information - Part 4. Guidelines for the Definition of Managed Objects*.

Kearns, M. J. (1989), *The Computational Complexity of Machine Learning*, M.I.T. Press.

Kearns, M. J. & Vazirani, U. V. (1994), *An Introduction to Computational Learning Theory*, M.I.T. Press.

Maes, P. & Kozierok, R. (1993), Learning interface agents, *in* 'Proceedings of the 11th Nat Conf on Artificial Intelligence', AAAI, MIT-Press/AAAI-Press.

Rivest, R. (1987), 'Learning decision lists', *Machine Learning* pp. 229–246.

Schlimmer, J. & Hermens, L. (1993), 'Software agents: Completing patterns and constructing user interfaces', *Journal of Applied Intelligence Research* **1**, 61–89.

8 BIOGRAPHY

Joël Quinqueton is graduated from the french Ecole Polytechnique, 1971, and got both PhD thesis (1976) and State Thesis (1981) from the Paris 6 University, in the field of Pattern Recognition and Artificial Intelligence. He is currently Research Director at INRIA and works in the Knowledge Acquisition and Representation Department of LIRMM, in Montpellier, France. His Research fields are Multi agent systems and Machine Learning.

Richard Nock got an Engineer Degree from the Ecole Nationale Superieure Agronomique

de Montpellier, and is now terminating a PhD thesis at LIRMM, in the field of Machine Learning, specially on Computational Learning Theory.
Babak Esfandiari got an Engineer Degree from the Institut des Sciences de l'Ingenieur de Montpellier, and a PhD thesis at LIRMM (January 1997), in the field of Agent Oriented techniques applied to Telecommunication Networks. He participated actively to the the "Réseau futé (Smart Net)" project, and is now at Mitel Corp, Ottawa, Canada.

ACKNOWLEDGMENTS

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.