Searching for similar semiqualitative temporal patterns in time-series databases

J.A. Ortega, R.M. Gasca, M. Toro
Departamento de Lenguajes y Sistemas Informáticos.
Universidad de Sevilla
Avda. Reina Mercedes s/n. Sevilla (Spain)
{ortega,gasca,mtoro}@lsi.us.es

Abstract

A way to obtain behaviour patterns of semiqualitative models of dynamic systems automatically is proposed in this paper. The temporal evolution of these models is stored into a database. This is a time series database. This database may be obtained as is explained in [Ortega *et al.* 1999] or by means of sensor data. In any way, the database contains the values of state variables and parameters.
Searching for similar patterns in such database is essential, because it helps in predictions, hypothesis testing and, in general, in data mining and rule discovery.
A language to carry out queries about the qualitative and temporal properties of this time-series database is also proposed. This language allows us to study all the states of a dynamic system: the stationary and the transient states. The language is also intended to classify the different qualitative behaviours of our model. This classification may be carried out according to a specific criterion or automatically by means of clustering algorithms. The semiqualitative behaviour of a system is expressed by means of hierarchical rules obtained by means of machine learning algorithms.
The methodology is applied to a logistics growth model with a delay.

Keywords: Knowledge acquisition, Semiqualitative models, Qualitative behaviours patterns

1 Introduction

In real systems studied in science and engineering, it is difficult to find mathematical models that represent them in an appropriate way. The modeling techniques should obviate certain aspects of the system. The simulation of these models helps us to study the evolution of the real system. A way to carry out these simulations is described in [Ortega *et al.* 1999] in depth.

On the other hand, it is not always possible to obtain a mathematical model of a system. Thus, it is necessary to apply other techniques in order to carry out its study. A possibility may be placing data sensors in the real system. The analysis of these data allows to study the system evolution.

Knowledge about dynamic systems may be quantitative, qualitative, and semiqualitative. When these models are studied all this knowledge should be taken into account. Different levels of numeric abstraction have been considered: purely qualitative [Kuipers 1994], semiqualitative [Kay 1996], [Berleant and Kuipers 1997] and [Ortega *et al.* 1998], and quantitative.

Different approximations have been developed in the literature when qualitative knowledge is taken into account: distributions of probability, transformation of non-linear to piecewise linear relationships, MonteCarlo method, fuzzy sets [Bonarini and Bontempi 1994], causal relations [Bousson and Travé-Massuyès 1994], and combination of all levels of qualitative and quantitative abstraction [Kay 1996].

In this paper, a technique to carry out the analysis of dynamic systems with qualitative and quantitative knowledge is proposed. The idea follows: *the quantitative behaviours of a real system are stored into a database and techniques of Knowledge Discovery in Databases (KDD) are applied to study the system.* The way to obtain the behaviours does not matter: by means of the simulation of a model or by means of the data sensors.

The term KDD [Adriaans and Zantinge 1996] is used to refer to the overall process of discovering useful knowledge from data. The problem of knowledge extraction from databases involves many steps, ranging

from data manipulation and retrieval to fundamental mathematical and statistical inference, search and reasoning. Although the problem of extracting knowledge from data (or observations) is not new, automation in the context of databases opens up many new unsolved problems.

KDD has evolved, and continues to evolve, from the confluence of research in such fields as databases, machine learning, pattern recognition, artificial intelligence and reasoning with uncertainty, knowledge acquisition for expert systems, data visualization, software discovery, information retrieval, and high-performance computing. KDD software systems incorporate theories, algorithms, and methods from all of these fields.

The term *data mining* is used most by statisticians, database researchers and more recently by the business community. Data mining is a particular step in the KDD process. The additional steps in KDD process are data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge and proper interpretation of the results of mining ensure the useful knowledge is derived from the data [Rastogi99]. A detailed descriptions of these steps may be found in [Ortega 2000].

On the other hand, historical, temporal and spatial databases have been profusely studied in the bibliography [Agrawal *et al.* 1995]. Specific applications include financial, marketing and production time series, such as stock prices, sales numbers, and also scientific databases with time series of sensor data. For example, in weather data, geological, environmental, etc. In this paper, we are interested in time-series databases corresponding to the evolution of semiqualitative dynamic systems.

Databases theories and tools provide the necessary infrastructure to store, access, and manipulate data. In this paper, a new way to study dynamic systems that evolve in the time is proposed merging "data mining", "time-series" and "databases engine". The proposed perspective tries to discover the underlying model in the database by means of a query/classification language.

It is also possible to obtain the behaviour patterns of these systems automatically by means of clustering techniques. Clustering is a discovery process in data mining. It groups a set of data in a away that maximizes the similarity within clusters and minimizes the similarity between two different clusters. These discovered clusters can help to explain the features of the underlying data distribution. The semiqualitative behaviour of a system is expressed by means of hierarchical rules obtained by means of machine learning algorithms.

The paper is organized as follows: firstly, our approach is explained and the concept of semiqualitative model is defined. Secondly, the kind of qualitative knowledge we are using in the language is introduced. Thirdly, the query/classification language on the database is described. Next, the clustering algorithms is described in detail and, finally, the way to obtain the hierarchical rules of the systems is explained. The proposed methodology is applied to a logistics growth model with a delay.

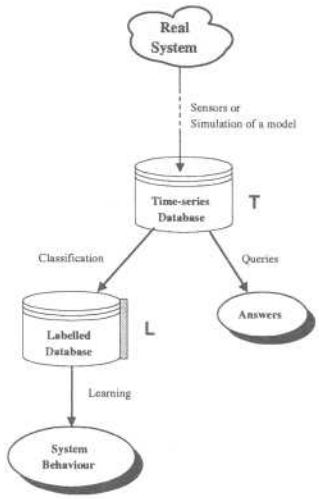

Figure 1: Our approach

2 Our approach

There is enough literature that studies stationary states of dynamic systems, however, the study of transient states is also necessary. Stationary and transient states of a semiqualitative dynamic system may be studied with the proposed approach. It is shown in figure 1.

We begin with a time-series database. It may be obtained by means of semiqualitative simulations [Ortega *et al.* 1999] or by means of data sensors. This is a trajectory database. A *trajectory* is a time-serie and it contains the values of the parameters and the values of all state variables from their initial value until their final value. Therefore, every trajectory stores the values of the transient and the stationary states of these variables of the system. Every trajectory is set of time-series of state variables.

We propose a language to carry out queries about the qualitative properties of the set of trajectories included in the database. A labelled database is obtained when these trajectories are classified according to some criteria. It is also possible to classify the database by means of an automatically process. In such case, it is necessary apply clustering techniques.

Qualitative behaviours patterns of the system may be automatically obtained from this database by

applying rule discovery based on genetic algorithms [Aguilar *et al.* 1998]. The steps of our approach are detailed in the following sections.

3 Semiqualitative models

In this paper, we focus on those dynamic systems where there may be qualitative knowledge in their parameters, initial conditions and/or vector field. They constitute the semiqualitative differential equations of the system.

A semiqualitative model S is represented by means of

$$\Phi(\dot{x}, x, q, t), \qquad x(t_0) = x_0, \qquad \Phi_0(q, x_0) \qquad (1)$$

being $x \in I\!R^n$ the state variables, q the parameters, t the time, $\dot{x}$ the derivative of the state variables with respect to the time, Φ constraints among $\dot{x}, x, q, t$, and Φ_0 constraints with the initial conditions. These constraints may be composed of qualitative knowledge, arithmetic and relational operators, intervals, predefined functions, $(ln, exp, sen, \ldots)$ and numbers.

4 Qualitative knowledge

Qualitative knowledge about a model may be composed of qualitative operators, qualitative labels, envelope functions and qualitative continuous functions [Ortega *et al.* 1999]. In this paper, we are interested in applying this knowledge to carry out queries with the language. Therefore, the representation and transformation techniques of this qualitative knowledge are described bellow.

The representation of the qualitative knowledge is carried out by means of operators which have associated real intervals. It simplifies the integration of qualitative and quantitative knowledge, and it also facilitates the incorporation of expert knowledge in the definition of the range of qualitative variables and parameters [Gasca 1998].

Every qualitative operator *op* is defined by means of an interval I_{op} which is supplied by the experts.

4.1 Unary qualitative operators

Every magnitude of the problem with qualitative knowledge has its own unary operators defined.

Let U_x be the unary operators for a variable x, i. e., $U_x = \{VN_x,\ MN_x, LN_x, AP0_x, LP_x, MP_x, VP_x\}$. They denote for x its qualitative labels: ***very negative, moderately negative, slightly negative, approximately zero,*** ***slightly positive, moderately positive,*** and ***very positive*** respectively.

The transformation rule for a unary operator is

$$op_u(e) \equiv \begin{cases} e - \alpha = 0 \\ \alpha \in I_u \end{cases} \qquad (2)$$

being α a new generated variable, and I_u the interval associated with operator op_u which is established in accordance to [Travé-Massuyès *et al.* 1997].

4.2 Binary qualitative operators

Let e_1, e_2 be two arithmetic expressions. A binary qualitative operator $b(e_1, e_2)$ denotes the qualitative order relationship between e_1 and e_2. These operators are classified into

- Operators related to the difference $\geq, =, \leq$. The following transformation rules are applied

$e_1 = e_2$	$\equiv$	$e_1 - e_2 = 0$
$e_1 \leq e_2$	$\equiv$	$\begin{cases} e_1 - e_2 - \alpha = 0 \\ \alpha \in [-\infty, 0] \end{cases}$
$e_1 \geq e_2$	$\equiv$	$\begin{cases} e_1 - e_2 - \alpha = 0 \\ \alpha \in [0, \infty] \end{cases}$

Table 1: Transformation rules

- Operators related to the quotient $\ll, -<, \sim, \approx, \gg, Vo, Ne, \ldots$. The applied transformation rule is

$$op_b(e_1, e_2) \equiv \begin{cases} e_1 - e_2 * \alpha = 0 \\ \alpha \in I_b \end{cases} \qquad (3)$$

being α a new variable and I_b the interval associated to op_b in accordance to [Travé-Massuyès *et al.* 1997].

5 Query/classification language

We propose a language to carry out queries and to classify with labels a time-series database T obtained with the quantitative simulations of a family of models [Ortega *et al.* 1999]. Therefore, this language allows us to classify the behaviour patterns of the system.

5.1 Abstract Syntax of the language

Let T be the time-series database and let r be every trajectory in this database. The abstract syntax of the proposed language is A query Q on the database T may

Q:	$\forall\, r \in C \bullet P$	P:	P_b
	$\exists\, r \in C \bullet P$		$P \wedge P$
			$P \vee P$
			$\neg P$
P_b :	P_d	P_d :	EQ
	O_T		CL
	$e_b(O_L([F], \{L\}))$		...
O_T :	*always* F	O_L :	*increase*
	sometime F		*decrease*
	always F *before* F		*periodic*
	always F *until* F		*maxim*
	sometime F *before* F		*minimum*
	sometime F *until* F		*constant*
	always F *after* F		*length*
	...		...
F :	F_b	F_b :	e_b
	F & F		$e \in I$
	$F \mid F$		$u(e)$
	$F \Rightarrow F$		$b(e_1, e_2)$
	$!F$		

Table 2: Queries abstract syntax

be a quantifier $\forall, \exists$ applied to T. This query try to determine if all the trajectories ($\forall$) or if there is at least one ($\exists$) verify the property P. This property may be formulated by means of the composition of other properties using the Boolean operators $\wedge, \vee, \neg$ and its result is the application of these operators among the partial properties.

A basic property P_b may be: a predefined property P_d, a Boolean expression e_b applied to a list operator $O_L([F], \{L\})$ applied to a list L of points or intervals which verify the formula F, or a temporal operator O_T.

A list operator $O_L([F], \{L\})$ returns the list L of points or intervals of the trajectory which verify the formula F.

A temporal operator O_T is used to describe properties in a concrete time of a trajectory, or to compare among different times of a trajectory, or to establish a sequence of behaviours of a trajectory. We have chosen a set of temporal operators from the temporal logic. The chosen had been those whose definition would be obtained by means of a final recursive definition [Lipeck and Saake 1987].

A defined property P_d is one whose formulation is automatic. They are queries commonly used in dynamical systems. There are two predefined: EQ, which is verified when the trajectory ends up in a stable equilibrium; and CL that it is verified when it ends up in a limit cycle.

A formula F may be composed of other formulas combined by means of Boolean operators &, |, ! and its result is the application of these operators among the partial formulas.

A basic formula F_b may be: a Boolean expression e_b, or if a numeric expression e belongs to an interval I, or a unary u or binary b qualitative operator.

Classification

A classification rule is formulated as a set of basic queries with labels and possibly other expressions. A classification problem is proposed in accordance to the following abstract syntax:

C :	$[r \in C, \text{automatic}]$
	$[r \in C, P_A] \Rightarrow A, e_{n1}, \ldots$
	$[r \in C, P_B] \Rightarrow B, e_{n2}, \ldots$
	...

Table 3: Classification abstract syntax

5.2 Semantics of the language

The semantics of every proposed statement is translated into a query of the database. A query $\forall\, r \in C \bullet P$ is *true* when all trajectories $r \in C$ verifies property P. To prove that an $\exists$ statement is *true*, it is necessary to find at least one trajectory $r \in T$ that verifies the property P.

A property P which is formulated by means of the application of Boolean operators $\wedge, \vee, \neg$ is *true* when the result of the application of these operators among the partial properties is true.

The result of the evaluation of a temporal operator O_T depends on its semantics. For example, *always* F returns a *true* value if all the values of r verify F. If this operator is *sometime* F returns a *true* value when at least a value of r verifies F. The semantic of all these temporal operators are described in accordance to [Lipeck and Saake 1987].

Let $e_b(O_L([F], \{L\}))$ be a basic property. This property is *true* if the Boolean expression applied to the list operator O_L returns a *true* value. The operator O_L returns the intervals or points of a trajectory which verify a formula F.

In order to evaluate a formula F, it is necessary to substitute its variables by their values. These values are obtained from T.

Let $[r, P_A] \Rightarrow A, e_{A1}$ be a classification rule. A trajectory $r \in T$ is classified with the label A if it verifies property P_A. It is also included into the database for this trajectory the result of the evaluation of e_{A1}. This

process is repeated with every classification rule. When the classification rule is [$r \in C$, automatic], it is necessary to apply clustering techniques in order to classify the database automatically.

5.3 Implementation details

The semantics of all queries and classification rules is translated into an algorithm. The execution of this algorithm yields the answers to the queries and labelled the database, and it is only necessary go once through the database. This algorithm is automatically generated.

Let $Q = \{Q_1, ..., Q_m\}$ be a set of m queries on a subset $C \subseteq T$. Let $b = \{b_1, ..., b_m\}$ be the m answers to Q. The template algorithm to obtain b follows:

```
1.  QueryClassifyQ(Q, C) return b
2.      b := e
3.      for  i := 1 to  i ≤ Card(C)
4.          t := t_f
5.          y := y_0
6.          while ¬(t = t_0)
7.              t := t − Δt
8.              y := f_t(y, C(i)_t)
9.          end while
10.         y := T_y(y)
11.         b := T_b(b, y)
12.     end for
```

being $y = \langle y_1, ..., y_p \rangle$ auxiliary variables. Every Boolean variable b accumulates the result of the query on the database and Boolean variable y accumulates the result of the query on the trajectory. The initial value of b (line 2) is the neutral element of the operation that it accumulates. The initial value for y (line 5) is calculated by applying the operation to the last instant of the trajectory. In order to modify these variables (lines 8 and 11) the operations f_t and T_y to the y variables and the operation T_b to the b are applied. The function f evaluated at time t is denoted by f_t. The algorithm proceeds backwards in time due to the transformation of the temporal operators [Ortega 2000].

The execution of this algorithm returns a Boolean value *true* or *false* as result of every query. The complexity of the algorithm is lineal, since each trajectory is read once for all the queries. This algorithm can be always obtained. The algorithm is instantiated depending on the language sentences as follows.

5.3.1 Queries

Table 4 collects the generated code for a query.

where every query Q_i generates at least an auxiliary variable y to contain the evaluation of P on the trajectory.

Q	Q_i
$\forall r \in C \bullet P_i$	$true$ $b := b \wedge y$
$\exists r \in C \bullet P_i$	$false$ $b := b \vee y$

Table 4: Query transformation

5.3.2 Properties

Table 5 collects the generated code for a property.

Property P_i	line	code
$P_1 \wedge P_2$	10	$y := y_{P1} \wedge y_{P2}$
$P_1 \vee P_2$	10	$y := y_{P1} \vee y_{P2}$
$\neg P$	10	$y := \neg y_P$

Table 5: Properties transformation

being y an auxiliary variable to store the result of P_i, and y_P to store the result of the elemental properties, and y is obtained by applying the $\wedge, \vee, \neg$ operators among the y_P variables.

5.3.3 List operators

Let $e_b(O_L([F], \{L\}))$ be a Boolean expression applied to a list operator. The generated code contains a list l to store the intervals of the trajectory that verify the formula F and a flag a to indicate when the formula F is activated. A record is generated for every interval of each trajectory r that verifies consecutively the formula F. This record contains the initial and final time of the interval and the result of the evaluation in the time t of the expressions that appears in L, and it is denoted as L_t. The operator O_L is applied on the obtained list l. The Boolean expression e_b is applied on the result of this application. The generated code is in table 6.

	List operator $e_b(O_L([F], \{L\}))$
line	code
5	$l := \{\}$
8	**if** F_t **then** $l := L_t \cup l$
10	$y := e_b(O_{L_i}(l))$

Table 6: List operator transformation

where F_t denotes the evaluation of formula F at instant t.

5.3.4 Temporary operators

The table 7 collects the generated code for temporal operators.

línea	Temporary operator O_T código generado
	always F
5	$y := F_t$
8	$y := F_t \wedge y$
	sometimes F
5	$y := F_t$
8	$y := F_t \vee y$
	always F before G
5	$y := F_t \vee G_t$
8	$y := F_t \vee (G_t \wedge y)$
	always F until G
5	$y := (F_t \wedge G_t) \vee F_t$
8	$y := (F_t \wedge G_t) \vee (F_t \wedge y)$
	sometime F before G
5	$y := (F_t \wedge \neg G_t) \vee (\neg G_t)$
8	$y := (F_t \wedge \neg G_t) \vee (\neg G_t \wedge y)$
...	

Table 7: Temporary operators transformation

This algorithm has been obtained in accordance to the definition of temporal operators provided in [Lipeck and Saake 1987].

5.3.5 Predefined properties

Predefined properties are queries whose formulation is automatic. They are queries commonly used on dynamic systems.

A trajectory ends in a stable equilibrium EQ when the states variables do not oscillates in the end of the simulation. Therefore the derivative of the state variables may be approximately equals to zero.

$$EQ \equiv always\ (t \in R_F \Rightarrow \dot{x} \approx 0) \qquad (4)$$

where R_F denotes the final range of the simulation, $\dot{x}$ is the derivative of the state variable x and $\approx$ is a binary qualitative operator. The generated algorithm is the corresponding to this temporal operator *always*.

A trajectory ends in a limit cycle CL when it oscillates in the final time of the simulation.

$$CL \equiv periodic([t \in R_F \wedge \dot{x} = 0 \wedge \dot{\dot{x}} > 0], \{x\}) \qquad (5)$$

This predicate obtains a list with the relative maxima (or minima) in the final range of the trajectory. A limit cycle appears when these maxima are periodically repeated.

5.3.6 Formulas

Formulas study specific features of a concrete instant in a trajectory. A formula F_t applied to r is verified when F is satisfied in the instant t. Transformation instructions depend on F, and they are in table 8. In order

Formula F	line	code
$F_t \wedge G_t$	8	$Eval(F_t) \wedge Eval(G_t)$
$F_t \vee G_t$	8	$Eval(F_t) \vee Eval(G_t)$
$F_t \Rightarrow G_t$	8	$Eval(F_t) \Rightarrow Eval(G_t)$
$\neg F_t$	8	$\neg Eval(F_t)$
F_t	8	$Eval(F_t)$

Table 8: Formulas transformation

to evaluate a formula F, it is necessary to substitute its variables for their values. These values are obtained from T.

6 Behaviour patterns

A behaviour pattern is each one of the possible behaviours that may appears in a dynamic system from its initial state until its final state. A qualitative model has a group of qualitative behaviours. Each one of them is a behaviour pattern. In figure 2 there are three different quantitative trajectories but they follow the same qualitative pattern: *begins with a value next to zero, then it goes growing until it reaches a maximum and next it oscillates until arriving to a positive value where it remains stable.*

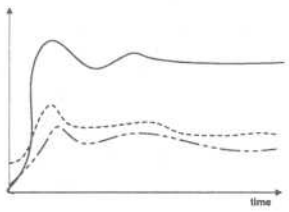

Figure 2: Similar qualitative behaviours

It is possible, that there are different ways to reach a stationary state. If the study is only limited to the stationary, it is not possible to recognize these differences and we would conclude that all the trajectories have the same behaviour. It is therefore necessary to further out the study with the transient state to discover these behaviours.

In order to obtain the patterns, the database must be labelled. Once the database is labelled, the techniques

to obtain the behaviour patterns are applied.

7 Labelling of the database

The labelling of the database assigns one or more labels to every trajectory. These labels may be assigned by a certain criterion provided by the user or by means of an automatic process. The data normalization is necessary to classify automatically the database. This normalization helps us to assign the labels to the trajectories by means of clustering algorithms.

7.1 Labelling with specific criteria

Let T be the database for the following classification rules: $C = \{C_1, C_2, C_3, ...\}$, being:

$$\begin{array}{l} C_1 \equiv [r \in T, P_{L_1}] \Rightarrow L_1, e_1 \\ C_2 \equiv [r \in T, P_{L_2}] \Rightarrow L_2 \\ C_3 \equiv [r \in T, P_{L_3}] \Rightarrow L_3, e_{31}, e_{32} \\ ... \end{array} \quad (6)$$

being r the trajectories of T, P_{E_i} the i-th predicate of the classification rule C_i, L_i labels and e expressions.

A label L is assigned to trajectory r when it verifies the property P. In order to apply the behaviour pattern algorithms, it is necessary that all trajectories have at least one label, otherwise, an empty label is assigned.

7.2 Automatic labelling

The automatic labelling classifies the database in accordance to the different behaviours of the dynamic system. It is necessary to supply the *similarity degree* to carry out this classification. It is a number in the interval [0,1] and its functionality is explained bellow.

Let T be the trajectories database. The idea is to find the different patterns that appear in T. Every trajectory is classified with a label L in accordance to its behaviour pattern.

Each trajectory $r \in T$ contains a temporal sequence for every state variable, therefore, a trajectory is a vector of time sequences. It is necessary to apply a clustering algorithm on these time series of the database to discover the behaviours. These sequences may be previously normalized. Any clustering algorithm needs to define a metric among the elements to be classified. This metric is provided by the *similarity degree.*

8 Normalization

Two trajectories with a similar behaviour may have a distance next to 0. In figure 2, the three trajectories follow the same qualitative pattern, but if we apply among them a distance like the Euclidean, it is concluded that they follow different behaviours. It is necessary to normalize these data to obtain the same pattern for similar qualitative behaviours. Therefore, these data will be scaled, translated and weighted.

8.1 Scaling

Let r be a trajectory, and let $x_1, ..., x_k$ be the values stored in the database of the state variables of r. These values will be scaled to the interval $[0, 1]$. For every variable $x \in r$ is necessary to obtain its maximum and minimum values x_{max}, x_{min} respectively. The value $x_r = x_{max} - x_{min}$ is the range of values of x. Let x_t be the value of x at time t. Let x'_t be the escalated value. It is obtained by means of the expression

$$x'_t = \frac{x_t - x_{min}}{x_r} \quad (7)$$

8.2 Translation

The trajectories $r_2, ..., r_N$ are translated to the first trajectory r_1. Let s be a trajectory $s \in \{r_2, ..., r_N\}$. For each trajectory s is defined $T_s = s_2 - s_1$ (figure 3).

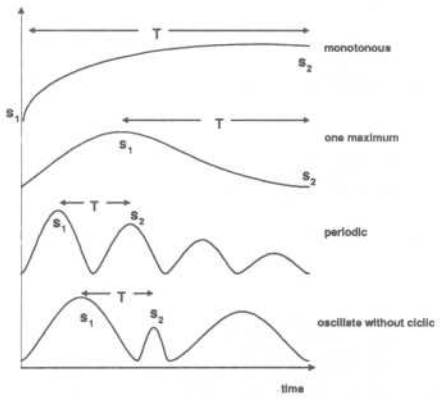

Figure 3: Translation and periodicity

The translation of s is obtained by solving the following system:

$$\begin{cases} t_1 = a * s_1 + b \\ t_2 = a * s_2 + b \end{cases} \quad (8)$$

being a the homotecy between both trajectories and b the translation. Solving this system, it is obtained that

the values of a, b are:

$$a = \frac{t_2 - t_1}{s_2 - s_1} = \frac{T_1}{T_s}, \quad b = \frac{t_1 s_2 - t_2 s_1}{s_2 - s_1} = \frac{t_1 s_2 - t_2 s_1}{T_s} \tag{9}$$

By substituting (9) in (8) and operating, we obtain the translation

$$t_s = \frac{t_1 s_2 + t_2 s_1 - t_1 T_s}{T_1} \tag{10}$$

8.3 Weighting

Once every trajectory has been translated, it is not possible to compare them directly. Therefore, it is necessary to interpolate these values.

Let r be a fixed trajectory and let s be the one that will be interpolated. Let t_s be the value obtained by the translation (10). Let t_i, t_j be where $t_i = t_0 + k\Delta t$ and $t_j = t_i + \Delta t$ where k is a natural number that verifies that $t_s \in [t_i, t_j]$. Let y_i, y_j be the translated values stored in the database for t_i, t_j respectively. We are interested in calculating the value t_s. There are several ways to calculate it: linear approximations, β-splins, ... In this paper, the value y_s is calculated by means of a linear approximation as follows:

$$y_s = y_i + \frac{y_j - y_i}{\Delta t}(t_s - t_i) \tag{11}$$

being $\Delta t = t_j - t_i$. This is the value of trajectory s that may be compared with the corresponding on the trajectory r.

9 Distance among trajectories

Let T be a database with N normalized trajectories. It is calculated a distance matrix D among these trajectories. It is a triangular matrix with its main diagonal equal to zero. This matrix is necessary for the clustering algorithm.

Let $\delta(r_i, r_j)$ be the distance between two trajectories. A possibility consists of calculating δ by means of the Euclidean distance. However, we know that trajectories are time series, therefore is better to calculate δ by means of the Fourier coefficients $a_0, a_1, ...$ There are several reasons to elect the Fourier coefficients [Agrawal *et al.* 1995]: the distance is preserved, they are easy to calculate, they concentrate the signal energy in a few coefficients and there is an algorithm *Fast Fourier Transform (FFT)* that calculate this coefficients efficiently. Therefore, Fourier transform is an appropriate way to calculate the distance in time series.

In this paper the distance δ between two trajectories r_i, r_j is defined by means of:

$$\delta_x(r_i, r_j) = \begin{cases} p_0|a_{0i} - a_{0j}|^2 + \\ p_1|a_{1i} - a_{1j}|^2 + \\ p_2|a_{2i} - a_{2j}|^2 + \\ p_3 \int |x_i(t) - x_j(t)|^2 dt + \\ p_4 \int |x_i'(t) - x_j'(t)|^2 dt + \\ p_5 \int |x_i''(t) - x_j''(t)|^2 dt \end{cases} \tag{12}$$

being x the state variables, $p_0, p_1, ..., p_5$ weights, a_{uv} the u-th Fourier coefficient, x_i, x_j normalized values stored in the database for the trajectories i, j respectively, and x_i', x_j' and x_i'', x_j'' first and second derivative of x_i, x_j values respectively.

This distance (12) is defined as an expression depending on: Fourier coefficients, weights, variable magnitude and first and second variable derivatives. The justification is as follows:

- In accordance to [Agrawal *et al.* 1995] only with a few Fourier coefficients the features of the original trajectory are obtained. Therefore, the term $p_3 \int |x_i - x_j|^2 dx$ provides the same information as the three first Fourier coefficients.
- Besides the magnitude, there are other features that are interesting to take into account from a qualitative perspective: the shape (first derivative) and the concavity (second derivative).
- Weights are introduced to take precedence over the magnitudes, the shapes or the concavity of the trajectories. We had not study the relation and relevance among these weights and the distance. All the tests have been carried out with a weight of 1.

A distance matrix D is obtained using the proposed definition of δ. The completeness of the algorithm to obtain D is exponential.

10 Clustering and decision rules

Clustering is a discovery process in data mining. It groups a set of data in a away that maximizes the similarity within clusters and minimizes the similarity between two different clusters. These discovered clusters can help to explain the features of the underlying data distribution. In recent years, a number of clustering algorithms for databases have been proposed: *DBScan* [Ester *et al.* 1996], *CURE* [Guha *et al.* 1998], *Chameleon* [Karypis *et al.* 1999], ...

A scalable clustering algorithm is proposed in this paper. Its execution puts together a trajectory and a label. This label determines the behaviour pattern of the trajectory.

1. **Clustering**(D, M) **return** M_E
2. $d_{mean} = Mean(D)$
3. $G := Graph(D, d_{mean})$
4. $M_E := Clusters(G, a)$

being a the similarity degree previously commented. This algorithm begins obtaining the distance matrix D among trajectories. Next, it calculates the mean of D. This mean distance d_{mean} is calculated to know the magnitude of the distance.

A weighted graph G is obtained with the k-neighbours of every trajectory. The vertex of G are the trajectories. The arcs are weighted with the relationship between d_{means} and the distance between two trajectories. Figure 4 shows examples of graphs building: original (a) and with 1- (b), 2- (c) and 3-(d) neighbours.

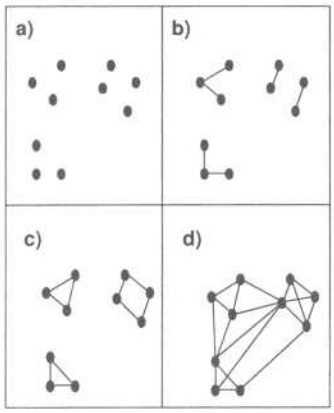

Figure 4: Graph with k-neighbours

It is applied an algorithm to the k-neighbours graph. The algorithm breaks those arcs between neighbours vertices whose weight w is less than

$$w < \frac{a * d_{mean}}{100}$$

being a the similarity degree.

The connected graphs represent every different behaviour pattern of the dynamic system. All the trajectories of every connected graph are classified with the same label. This is carried out by the function $Clusters(G, a)$.

Once the database has been labelled, the pattern behaviour of the system is represented by means of a set of hierarchical decision rules. These rules are obtained using the program *COGITO* [Aguilar *et al.* 1998] to the bidimensional dynamic array described in the section 7 of this paper.

11 Application to a logistics growth model with a delay

It is very common to find growth processes in which an initial phase of exponential growth is followed by another phase of approaching to a saturation value asymptotically (figure 5). These are given the following generic names: logistic, sigmoidal, and s-shaped processes.

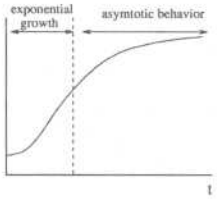

Figure 5: Logistics growth curve

This growth is exhibited by systems for which exponential expansion is truncated by the limitation of the resources required for this growth. This behaviour is due to a positive feedback that is dominant in the initial phase, and a negative feedback that is dominant in the final phase.

In literature, these models have been profusely studied. They abound both in natural processes, and in social and socio-technical systems. They appear in the evolution of bacteria, in mineral extraction, in world population growth, and in economic development. Learning curves also show this type of behaviour.

The same thing happens with some diffusion phenomena within a given population, such as epidemics or rumors. Other examples of this behaviour are a population that grows in a habitat with limited resources, a technological innovation that is being introduced, or a new product that is being put on the market. In all these cases, their common behaviours are shown in figure 6. There is a bimodal behaviour pattern attractor: A stands for normal growth, and O for decay. It can be observed how it combines exponential with asymptotic growth. This phenomenon was first modeled by the Belgian sociologist P. F. Verhulst in relation with human population growth. Nowadays, it has a wide variety of applications, and some of them have just been mentioned.

Let S be the qualitative model. If we add a delay in the feedback paths of S, then its differential equations are

$$\Phi \equiv \begin{cases} \dot{x} = x(n\,r - m), \\ y = delay_\tau(x), \quad x > 0, \quad r = h_1(y), \\ h_1 \equiv \{(-\infty, -\infty), +, (d_0, 0), +, (0, 1), \\ \qquad +, (d_1, e_0), -, (1, 0), -(+\infty, -\infty)\} \end{cases}$$

being n the increasing factor, m the decreasing fac-

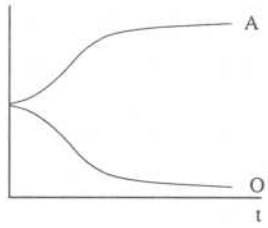

Figure 6: Logistics growth model

tor, and h_1 a qualitative continuous function defined by means of points and the derivative sign among two consecutive points. These functions are explained in detail in [Ortega 2000]. This function has a maximum point at (x_1, y_0). The initial conditions are

$$\Phi_0 \equiv \begin{cases} x_0 \in [LP_x, MP_x], \\ LP_x(m), \\ LP_x(n), \\ \tau \in [MP_\tau, VP_\tau] \end{cases}$$

where LP, MP, VP are the qualitative unary operators *slightly positive, moderately positive and very positive* for x, τ variables.

We would like to know:
1. if an equilibrium is always reached
2. if there is an equilibrium whose value is not zero
3. if all the trajectories with value zero at the equilibrium are reached without oscillations.
4. To classify the database in accordance to the behaviours of the system.

The methodology is applied to this model. Firstly, it is necessary to define the intervals associated with every qualitative operator, they have been defined for this problem by the experts as follows:

$$LP_x = [0, 1] \quad MP_x = [1, 3]$$
$$MP_\tau = [0.5, 4] \quad VP_\tau = [4, 10]$$

The methodology described in [Ortega *et al.* 1999] is applied to obtain the trajectory database T. The proposed methodology transforms this semiqualitative model into a family of quantitative models. Stochastic techniques are applied to choose a quantitative model of the family. The simulation of every quantitative model generates a trajectory. All trajectories put together constitute the database T.

Applying the proposed language, the proposed queries are formulated as follows:
1. $\forall r \in T \bullet EQ$
2. $\exists r \in T \bullet (EQ \wedge sometimet \simeq t_f \Rightarrow x \not\approx 0)$
3. $\forall r \in T \bullet (EQ \wedge sometimet \simeq t_f \Rightarrow x \approx 0 \wedge length(\dot{x} = 0 \wedge \dot{\dot{x}} < 0))$

The list of points where $\dot{x} = 0$ and $\dot{\dot{x}} < 0$ is the list with the maximum points. There are no oscillations when its length is 0.

The answers to the proposed questions were:
1. $b_1 = True$, all the trajectories of T reach a stable equilibrium. Therefore, we conclude: *there is no cycle limit.*
2. $b_2 = True$, some trajectories of T reach an equilibrium whose value is not zero. Therefore, this is the first behaviour we have obtained. We know it as *recovered equilibrium.* 3. $b_3 = False$, there are at least two ways to reach this equilibrium: with oscillations (this behaviour is called as *retarded catastrophe*) and the other way is without oscillations (that it is called as *decay and extinction*).

We apply to T the described clustering algorithm with a similarity degree of $a = 0.1$. This algorithm found the three possible behaviours patters for this system. This result is in accordance to the previous queries. The

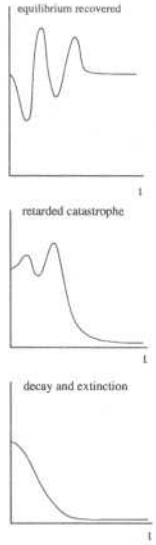

Figure 7: Logistics growth model with a delay

obtained results with this way to discover the behaviour patterns are in accordance to others appeared in the bibliography [Aracil *et al.* 1997] and [Karsky *et al.* 1992] where the results are concluded by means of a mathematical reasoning. This circumstance encourages us to continue developing this methodology and to apply it to other systems with qualitative and quantitative knowledge.

12 Conclusions and further work

In this paper, we have presented a way to obtain temporal and semiqualitative behaviour patterns of dynamic systems with qualitative and quantitative knowledge. This approach is based on a transformation process, definition of a query/classification, language on a quantitative behaviours database, and clustering techniques.

There is enough bibliography that studies stationary states of dynamic systems. However, the study of transient states is also necessary. These studies are possible with the proposed language.

In the future, the query/classification language must be enriched with operators for comparing trajectories, spatial operators, etc. Dynamic systems with constraints and with multiple scales of time are also one of our future points of interest.

The methodology is being applied in a real computer-controlled process. It is a production industrial system. *Altos Hornos de Sevilla* is a metallurgical Company interested in modifying its steel control production system applying the whole methodology ([Ortega *et al.* 1999] and this paper). The production engineers of this company wish to improve the steel quality, and, if possible, reduce the production costs. This collaboration is now developing and in forthcoming papers, we will describe this system in detail and the conclusions we shall obtain.

Acknowledgments

This work was partially supported by the Spanish Interministerial Committee of Science and Technology by means of the program TIC98-1635-E.

We thank the QR reviewers for their constructive criticism and helpful comments.

References

[Adriaans and Zantinge 1996] Adriaans P.; and Zantinge D. 1996 Data Mining. *Addison Wesley Longman.*

[Aguilar *et al.* 1998] Aguilar J.; Riquelme J.; and Toro M. 1998. *Decision queue classifier for supervised learning using rotated hyperboxes. Lecture Notes in Artificial Intelligence* 1484: 326–336.

[Agrawal *et al.* 1995] Agrawal R.; Lin K.I.; Sawhney H.S.; and Shim K. 1995 Fast similarity search in the presence of noise, scaling, and translation in time series databases. In *The* 21^{th} *VLDB Conference*, Zurich (Switzerland).

[Aracil *et al.* 1997] Aracil J.; Ponce E.; and Pizarro L. 1997. Behavior patterns of logistic models with a delay *Mathematics and computer in simulation* 44:123–141.

[Berleant and Kuipers 1997] Berleant D., and Kuipers B.J. 1997 Qualitative and quantitative simulation: bridging the gap. *Artificial Intelligence* 95: 215–255

[Bonarini and Bontempi 1994] Bonarini A. and Bontempi G. 1994 Qua.SI.: Qualitative simulation approach for fuzzy models. In *Proceedings of the 1994 Modelling and Simulation Multiconference,* 420–424, Barcelona, Spain.: Society for Computer Simulation International.

[Bousson and Travé-Massuyès 1994] Bousson K. and Travé-Massuyès L. 1994 Putting more numbers in the qualitative simulator CA-EN. In *Proceedings of the Second International Conference on Inteligent Systems Engineering.* Hamburg.: International Conferences on Intelligent Systems Engineering, Inc.

[Ester *et al.* 1996] Ester M.; Kriegel H.-P.; Sander J.; and Xu X. 1996 A density-based algorithm for discovering clusters in large spatial database with noise. *International Conference on Knowledge Discovery in Databases and Data Mining* (KDD-96), Portland (USA), 226–231.

[Gasca 1998] Gasca R.M. 1998 Razonamiento y Simulación en Sistemas que integran conocimiento cualitativo y cuantitativo. Ph.D. diss., Dept. of Computer Science, Seville Univ.

[Guha *et al.* 1998] Guha S.; Rastogi R.; and Shim K. 1998 CURE: And efficient clustering algorithm for large databases. In *Proceedings ACM SIGMOD International Conference Management of Data* New York (USA), 73–84.

[Kay 1996] Kay H. 1996. Refining imprecise models and their behaviors. Ph.D. diss., Texas Univ.

[Karsky *et al.* 1992] Karsky M.; Dore J.-C.; and Gueneau P. 1992, Da la possibilité d'apparition de catastrophes différès. *Ecodecision No 6*

[Karypis *et al.* 1999] Karypis G.; Han E.-H.; and Kumar V. 1999 Chameleon: Hierarchical clustering using dynamic modeling *Computer* nº 32 (8), 68–75.

[Kuipers 1994] Kuipers B.J. 1994 *Qualitative reasoning. Modeling and simulation with incomplete knowledge,* Cambridge, Massachussetts.: The MIT Press.

[Lipeck and Saake 1987] Lipeck U.W.; and Saake G. 1987 Monitoring dynamic integrity constraints based on temporal logic *Information Systems* nº 12(3), pp. 255-269.

[Ortega *et al.* 1998] Ortega J.A.; Gasca R.M.; and Toro M. 1998a. Including qualitative knowledge in semiqualitative dynamical systems. *Lecture Notes in Artificial Intelligence* 1415: 329–338.

[Ortega *et al.* 1999] Ortega J.A.; Gasca R.M.; and Toro M. 1999. A semiqualitative methodology for reasoning about dynamic systems. In *The* 13^{th} *International Workshop on Qualitative Reasoning* Loch Awe (Scotland), 169–177.

[Ortega 2000] Ortega J.A. 2000 Patrones de comportamiento temporal en modelos semicualitativos con restricciones. Ph.D. diss., Dept. of Computer Science, Seville Univ.

[Rastogi99] Rastogi R.; and Shim K. 1999. *Data mining on large databases* Bell laboratories.

[Travé-Massuyès *et al.* 1997] Travé-Massuyès L.; Dague Ph.; and Guerrin F. 1997. *Le raisonement qualitativ pour les sciences de l'ingénieur.* Paris, France: Hermes Ed.