Abstracting Irrelevant Distinctions in Qualitative Simulation†

Pierre Fouché[1] & Benjamin Kuipers[2]

[1]Université de Technologie de Compiègne
C.N.R.S. U.R.A. 817
B.P. 649
60206 Compiègne, France
Email: fouche@frutc51.bitnet

[2]Department of Computer Sciences
The University of Texas at Austin
Austin TX 78712, USA
Email: kuipers@cs.utexas.edu

Abstract: One main problem in qualitative simulation is that it often produces too detailed qualitative descriptions of a system's possible behaviors, or produce a lot of behaviors that differ very slightly. This paper addresses the problem of summarizing the result of a qualitative simulation to make it more perspicuous. Two causes of behavior proliferation have been identified and two algorithms to aggregate behaviors that do not differ significantly are presented. Simple examples are used to illustrate the functionning of each algorithm.

† This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin, and in the AI Group in CNRS-URA 817, The University of Compiègne. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-8602665, IRI-8905494, and IRI-8904454, by NASA grants NAG 2-507, and by the Texas Advanced Research Program under grant no. 003658175. Pierre Fouché holds a grant from Rhône-Poulenc.

1. Introduction

The goal of qualitative simulation is to produce qualitative descriptions of a system's possible behaviors from a qualitative model. Two main problems in qualitative simulation can be distinguished:

- Due to the interaction of qualitative representations and local reasoning techniques, qualitative simulation sometimes produces behaviors that are not actually possible. Such behaviors are called *spurious* behaviors.
- Qualitative simulation may give too many details compared to what is expected, or produce a lot of behaviors that differ very slightly. It is then very difficult to extract the behavioral features of a system.

This papers deals with the second problem. In practice, analyzing a large behavior tree or a large envisionment graph is a difficult task and tools to summarize the result of a qualitative simulation to make it more perspicuous are necessary to make qualitative simulation useable. At first sight, there are many ways to aggregate similar behaviors, and finding general similarity criteria that could be successfully applied on a large variety of simulation results is critical. After having analyzed many envisionment graphs, it turned out that behavior proliferation could generally be attributed to two main causes: *chatter* and *occurrence branching*. Chatter happens when the derivative of a variable is unconstrained; occurrence branching happens when the temporal ordering of two or more events cannot be determined. For each of these causes, we will present an algorithm to aggregate behaviors that do not differ significantly and apply it to a simple example to illustrate its functionning. The algorithms presented here can be used or adapted to any system that produce an envisionment graph. They have been implemented in Common Lisp as extensions of Kuipers's QSIM [11],[12], and tested on TI Explorers, MicroExplorers and on VAX Workstations running Xwindows.

We will first recall some basics definitions necessary to qualitative simulation.

1.1. Basic Concepts and Definitions

A physical *system* φ is modelled by a set of state *variables* and a set of *constraints* that relate them. Variables are real-valued, continuously differentiable functions of time.

The *qualitative magnitude* or *qmag* of a variable is defined with respect to a set of values, called landmarks, that represent real qualitative distinctions for the behavior of the variable. A qualitative magnitude is either a landmark value or an open interval between two adjacent landmarks.

The *direction of change* or *qdir* of a variable is the sign of its derivative with respect to time, and can be either *dec*, *std*, or *inc* (for *decreasing*, *steady*, or *increasing*).

The *qualitative value* or *qval* of a variable is the pair (*qmag*,*qdir*)

A *qualitative state* of a system is a set of qualitative values of all the system's variables, which are consistent with the set of constraints. We distinguish states that are valid only for an instant, called *P-states*, and states that are valid over an interval of time, called *I-states*.

Qualitative simulators that use states to describe a system's behavior are called *state-based* simulators. Simulators that describe a system's behavior in terms of behaviors of its variables, considered individually, are called *history-based* simulators (see paragraph 3.2).

1.2. Behavior Generation VS Envisionment

A *qualitative behavior* (in a state-based representation) is a sequence of states which alternates P-states and I-states. Qualitative behaviors can be produced in two ways:

- directly from an initial state; the result of a simulation is a *tree of possible behaviors*.
- from a graph, called *envisionment*, whose vertices are possible states and edges are valid transitions between them. A behavior is then a *path* in the graph. An envisionment is said to be *total* if it contains all the possible states of the system, or *attainable* if it only contains states that can be reached from an initial state.

Kuipers' QSIM [11],[12] adopts the first strategy, while Forbus' QPE [4] and de Kleer & Brown's ENVISION [10] produce envisionments. On one hand, an envisionment provide a compact, finite description of a set of possible behaviors. On the other hand, reasoning over behaviors allows one to use stronger reasoning techniques, such as reasoning in a phase space representation, or reasoning about energy (see [6]). An attractive feature of behavior generation is that it allows one to introduce new distinctions during simulation, by dynamically creating new landmarks, for instance, when a variable reaches a critical value. The nature of oscillations of oscillatory systems can be determined using this mechanism, but this cannot be done using an envisionment graph.

However, when behaviors begin to proliferate, new landmarks prevent an easy comparison of similar behaviors, because they are attached to a specific behavior. This is the reason why we extended QSIM so it can now produce envisionments as well as directly generate behaviors. The remainder of this paper presents two algorithms to aggregate similar states and behaviors in an envisionment graph.

2. Chatter

The graph in figure 1 contains many short cycles. A filled circle represents a state at some time point, an empty circle a state at some time interval. Analyzing why such cycles are produced shows that many states differ only by the direction of change of one or several variables. These distinctions are very often useless and the graph as it is in figure 1 is not easy to interpret. In the following sections, we explain how to group states which, ignoring the directions of change of the variables, are identical. We introduce the concept of *qmag-equivalence* and describe how to aggregate states in a same equivalence class.

2.1. Building Equivalence Classes

The algorithm to eliminate the proliferation of such cycles is based on partitioning the graph into equivalence classes, which gather states for which all the variables have the same qualitative magnitudes. The precise definition of the equivalence relation *qmag-equivalent* is the following:

Definition: Two states S_1 and S_2 are *qmag-equivalent* if and only if:

$\forall v,\ qmag(v, S_1) = qmag(v, S_2)$

It is easy to check that this relation is an equivalence relation over the set of possible states X_φ. From this definition and the set of possible states X_φ, we build a partition P of X_φ, $P = \{C_i\}$, each C_i being an equivalence class.

The next step is to identify the differences between all the states in C_i and to aggregate them to build a new state T_i which generalizes all the states in C_i.

2.2. Generalizing Equivalent States

Since all the states in C_i are *qmag-equivalent*, they differ only by the directions of change of the variables and it makes sense to define the magnitude of a variable v in a class C_i, $qmag(v, C_i)$. The magnitude of v in T_i, $qmag(v, T_i)$ is set to $qmag(v, C_i)$. To determine the direction of change of v in T_i, we build the set $D(v, C_i)$ of its possible directions of change in C_i: $D(v, C_i) = \bigcup_{S \subset C_i} qdir(v, S)$. If $D(v, C_i)$ has only one element, then $qdir(v, T_i) = D(v, C_i)$. If it has more than one element, then we set $qdir(v, T_i)$ to *ign*[1]. Note that we can lose information in building T_i: having $qdir(v, T_i) = ign$ does not necessarily imply that v can take on any direction of change in the states that T_i generalizes (but, for instance, *inc* or *std* only). We also lose information on how states in C_i are connected together.

[1] *ign* means that the direction of change is *ign*ored.

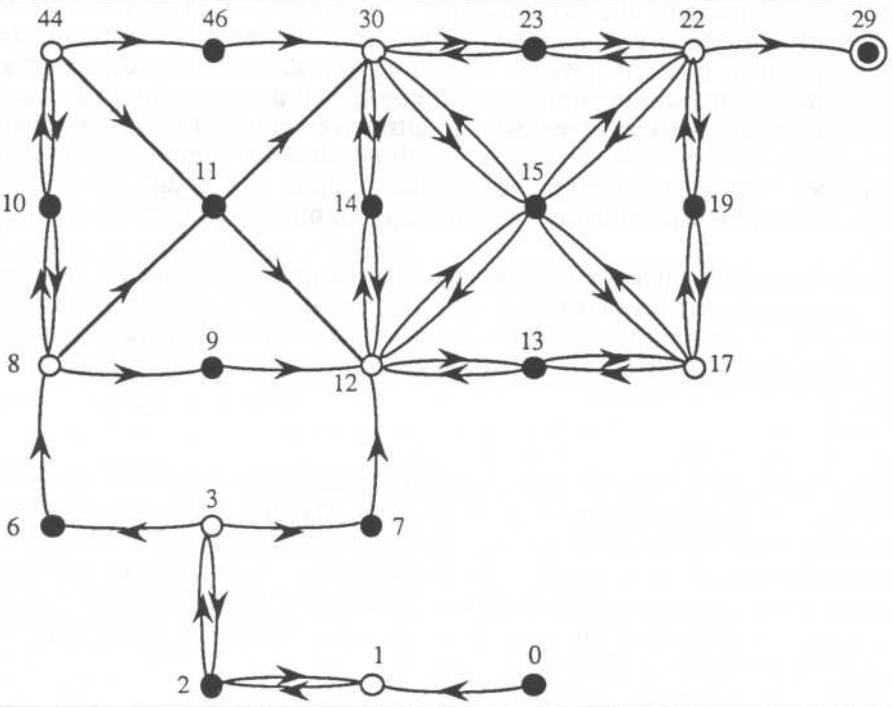

Figure 1: Envisionment graph of A → B → C → D

2.3. Setting Time Label, Predecessors and Successors

In building generalizations, we also want to keep the distinction between P-states and I-states, and connect the more general state to other states in the graph. We begin with defining the set of successors and predecessors of a class:

$$succ(C_i) = \bigcup_{S \subset C_i} succ(S) \qquad pred(C_i) = \bigcup_{S \subset C_i} pred(S)$$

We also make a distinction between P- and I-successors and predecessors, and split these sets into *P-succ* and *I-succ*:

$P\text{-}succ(C_i) = \{S : S \in succ(C_i)$ and $time(S) = point\}$

$P\text{-}pred(C_i) = \{S : S \in pred(C_i)$ and $time(S) = point\}$

$I\text{-}succ(C_i) = \{S : S \in succ(C_i)$ and $time(S) = interval\}$

$I\text{-}pred(C_i) = \{S : S \in pred(C_i)$ and $time(S) = interval\}$

The successors and predecessors of T_i are defined straight-forwardly:

$P\text{-}succ(T_i) = \{S : S \in P\text{-}succ(C_i)$ and $S \notin C_i\}$

$P\text{-}pred(T_i) = \{S : S \in P\text{-}pred(C_i)$ and $S \notin C_i\}$

$I\text{-}succ(T_i) = \{S : S \in I\text{-}succ(C_i)$ and $S \notin C_i\}$

$I\text{-}pred(T_i) = \{S : S \in I\text{-}pred(C_i)$ and $S \notin C_i\}$

Defining whether T_i is a P- or I-state depends on T_i's successors and predecessors:

- If $P\text{-}succ = P\text{-}pred = \emptyset$ then T_i must be a P-state and we set:
  $time(T_i) = point.$
  $pred(T_i) = I\text{-}pred(T_i)$
  $succ(T_i) = I\text{-}succ(T_i)$

The successors of T_i's predecessors and the predecessors of T_i's successors must be updated as well:

$\forall S \in pred(T_i), succ(S) = \{T_i\} \cup (succ(S) - C_i)$

$\forall S \in succ(T_i), pred(S) = \{T_i\} \cup (pred(S) - C_i)$

- If $I\text{-}succ = I\text{-}pred = empty$ then T_i must be an I-state and we set:
  $time(T_i) = interval.$
  $pred(T_i) = P\text{-}pred(T_i)$
  $succ(T_i) = P\text{-}succ(T_i)$
  Successors of T_i's predecessors and predecessors of T_i's successors are updated in exactly the same way.
- Otherwise, if we want to maintain the semantics of P-states and I-states, it is not possible to give a unique time label to T_i and T_i must be split into two identical states, T_i^p and T_i^i, one being a P-state and the other an I-state:

  $\forall v, qval(v, T_i^p) = qval(v, T_i^i) = qval(v, T_i)$
  $time(T_i^p) = point \text{ and } time(T_i^i) = interval$
  $pred(T_i^p) = I\text{-}pred(T_i)$
  $succ(T_i^i) = P\text{-}succ(T_i)$
  $pred(T_i^i) = P\text{-}pred(T_i) \cup \{T_i^p\}$
  $succ(T_i^p) = I\text{-}succ(T_i) \cup \{T_i^i\}$
  Predecessors and successors of T_i^i and T_i^p are updated slightly differently:

  $\forall S \in pred(T_i^p), succ(S) = \{T_i^p\} \cup (succ(S) - C_i)$

  $\forall S \in succ(T_i^i), pred(S) = \{T_i^i\} \cup (pred(S) - C_i)$

  $\forall S \in pred(T_i^i), S \neq T_i^p, succ(S) = \{T_i^i\} \cup (succ(S) - C_i)$

  $\forall S \in succ(T_i^p), S \neq T_i^i, pred(S) = \{T_i^p\} \cup (pred(S) - C_i)$

Thus, if P is an element of $I\text{-}pred(T_i)$ and Q an element of $P\text{-}succ(T_i)$, the sequence ($P\ T_i^p\ T_i^i\ Q$) is a possible behavior, which can be interpreted like this: the system is in state P for a certain amount of time, and then evolves to be in state T_i^p. This state is not instantaneous (system in state T_i^i) and the system finally reaches state Q. A transition between T_i^i and T_i^p is invalid because an I-state can only lead to a P-state in which at least one variable has a different qualitative value [12].

2.4. Example

Figure 1 shows the envisionment graph[2] of a system modeling an isothermal batch reactor where three reactions in series occur: $A\text{->}B\text{->}C\text{->}D$. C_x denotes the concentration of species x. The modeling equations are:

$$\frac{dC_A}{dt} = -k_1 C_A$$

$$\frac{dC_B}{dt} = k_1 C_A - k_2 C_B$$

$$\frac{dC_C}{dt} = k_2 C_B - k_3 C_C$$

$$\frac{dC_D}{dt} = k_3 C_C$$

The simulation started from an initial state with species A at a concentration C_0, and was constrained to produce analytic functions only (analytic functions are infinitely differentiable and

[2]We redrew the graph by hand, because it was not easy to highlight equivalence classes on the graph drawn by QSIM.

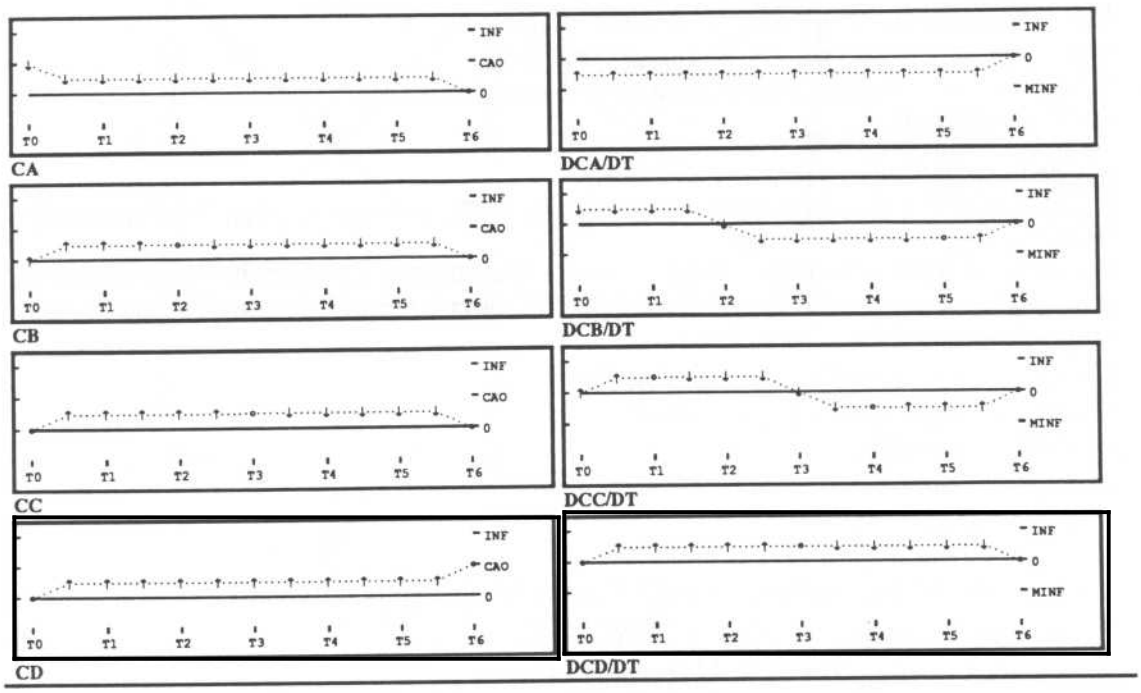

Fig. 2: One behavior of A → B → C → D

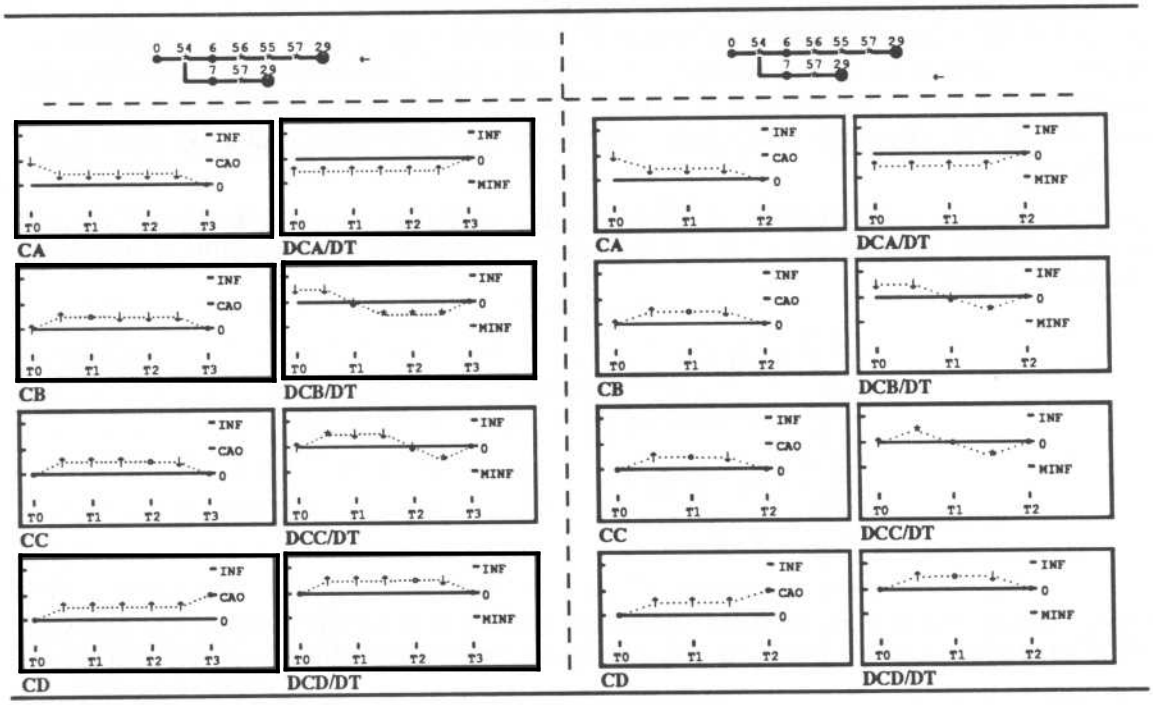

Fig. 3: Elimination of chatter

have the following property: they cannot be steady over an interval unless they are steady over their whole domain of definition). Figure 2 shows one particular behavior.

The graph can be partitioned into eight qmag-equivalent classes:

Classes	Elements
C_1	S_0
C_2	S_1 S_2 S_3
C_3	S_6
C_4	S_7
C_5	S_8 S_{10} S_{44}
C_6	S_9 S_{11} S_{46}
C_7	S_{12} S_{13} S_{14} S_{15} S_{17} S_{19} S_{22} S_{23} S_{30}
C_8	S_{29}

The direction of change of the derivative of B's concentration, $\frac{dC_B}{dt}$, can be ignored in classes C_5, C_6 and C_7 and that of $\frac{dC_C}{dt}$ in classes C_2 and C_7. Three I-states are created from class C_2, C_5 and C_7 and one P-state from class C_6. The resulting graph contains only two behaviors shown in figure 3. New states are represented by a star. The direction of change *ign* is also represented by a star.

3. Occurrence Branching

Intractable branching in an envisionment graph, as well as in a behavior tree, is often caused by another phenomenon called *Occurrence Branching*. It happens when the temporal ordering of two or more events cannot be determined. Qualitative simulation then branches on all the possible orderings.

3.1. Idea

This pathological phenomenon can be characterized in an envisionment graph. When a system evolves from a state S_i to a state S_f, there may be a lot of paths (behaviors) from S_i to S_f. If all the variables of the system, considered individually, exhibit the same behavior in all the paths from S_i to S_f, then the behaviors differ only by the temporal ordering of events taking place between S_i and S_f. In the following paragraphs we will formalize these intuitive notions and see how the algorithm works on an exemple where occurrence branching masks the main behavioral feature of a system.

3.2. Definitions and Algorithm

3.2.1. Histories

A system's behavior is a path in an envisionment graph. Thus a behavior B is represented by a sequence of states $B = [S_i]$. If v is one of the system's variables we first define the history of a variable:

Definition: the *history* $H(v, B)$ *of a variable* v in a behavior B is the sequence of v's qualitative values in B:

$H(v, B) = [qval(v, S_i)]$ if $B=[S_i]$.

Definition: the *history* $H(\varphi, B)$ *of a system* φ in a behavior B is the set of histories of all its variables:

$H(\varphi,B) = \{H(v, B)\}$, $\forall v$.

The concept of history was first introduced by Hayes [8] and Forbus [3].These definitions are similar to those in [16]. When building histories, we would like to avoid keeping distinctions that are not relevant to the behavior of a particular variable. But from the preceding definition, a

variable's history can contain subsequences of equivalent qualitative values. The concept of *history reduction* helps define the concept of concise history[3] and suggests an algorithm to build them.

Definition: *Reducing* a history $H = [V_i]$ consists of replacing a subsequence of H, $[V_k V_{k+1} V_{k+2}]$, such that $qmag(V_k) = qmag(V_{k+1}) = qmag(V_{k+2})$, $qdir(V_k) = qdir(V_{k+1}) = qdir(V_{k+2})$ and $time(V_k) = time(V_{k+2}) = interval$, with the subsequence $[V_k]$.

The motivation for this definition is that we want to eliminate irrelevant time-points. If V_k, V_{k+1} and V_{k+2} all have the same qmags and qdirs, then time-point $k+1$ is not a distinguished time point for the variable and V_{k+1} does not introduce an interesting distinction in the history. It can be withdrawn. V_{k+2} is identical to V_k and can be withdrawn as well.

Definition: The *concise history* $h(v, B)$ *of a variable* v in a behavior B is a reduced history of $H(v, B)$ which cannot be reduced anymore.

Definition: The *concise history* $h(\varphi, B)$ *of a system* φ in a behavior B is the set of concise histories of all its variables:

$h(\varphi, B) = \{h(v,B)\}, \forall v.$

A system's concise history keeps all the relevant behavioral features of all the system's variables, without taking into account the ordering of events between variables. We can define an equivalence relation based on the concept of concise history:

Definition: Two behaviors B_1 and B_2 of a system φ are *history-equivalent* if φ's concise histories for B_1 and B_2 are the same.

3.2.2. Single-Input, Single-Output Subgraph

Now the question is to determine when it is appropriate to build concise histories. Intuitively a system may exhibit occurrence branching when all the behaviors starting from a state S_i lead to another unique state S_f. The concept of *Single-Input, Single-Output* or *SISO* graph formalizes this idea.

Definition: Let $G = \{X, U\}$ be a graph. X is the set of nodes (or vertices), U the set of edges. Let $G' = \{X', U'\}$ be a subgraph of G ($X' \subset X$, $U' \subset U$). Let x_i and x_f be two nodes of G'. G' is a *SISO subgraph* of G with starting node x_i and ending node x_f if:

- G' is acyclic,
- G is no longer connected if the edges adjacent to x_i and x_f in G' are removed from G,
- x_i has at least two successors in G' and x_f two predecessors in G',
- $\forall x \in G'$, there is a path from x to x_f in G'.

Thus if a system is in a state S_i which is a starting node of a SISO subgraph G', and follows a transition from S_i to another state which is in G', it will sooner or later reach the final node S_f of the subgraph. If there are multiple paths in the subgraph, they may differ only by occurrence branching. Finding SISO subgraphs is very similar to finding triconnected components of a graph, for which an algorithm linear in the number of edges exists [9]. The algorithm we use find only minimal SISO graphs, that is, graphs that do not contain any other SISO graphs, and whose initial and final states are P-states. See [7] for a complete description of the algorithm.

3.2.3. Eliminating Occurrence Branching

Once a SISO subgraph is identified, we generate concise histories for all the possible behaviors in the subgraph, and partition the set of behaviors using the relation *history-equivalent*. For each class, we generate a new behavior corresponding to the system concise history. There is a little difficulty in building a behavior from a system's concise history, for the length of variable concise histories are not necessarily all the same. When the end of some variable's concise history is reached, we build successor states using the last value of that variable. This is not really critical, for states built in this process are not meaningful by themselves, but only the behavior as a whole is: when a state contains two events, it does not imply that they occur simultaneously, but that they

---

[3]Williams [86] defines concise histories in terms of maximal *episodes*, but they can be define without this concept.

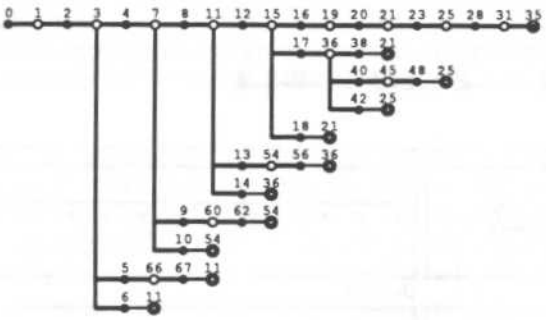

Fig. 4: Envisionment of A → B → C → D → C→ E

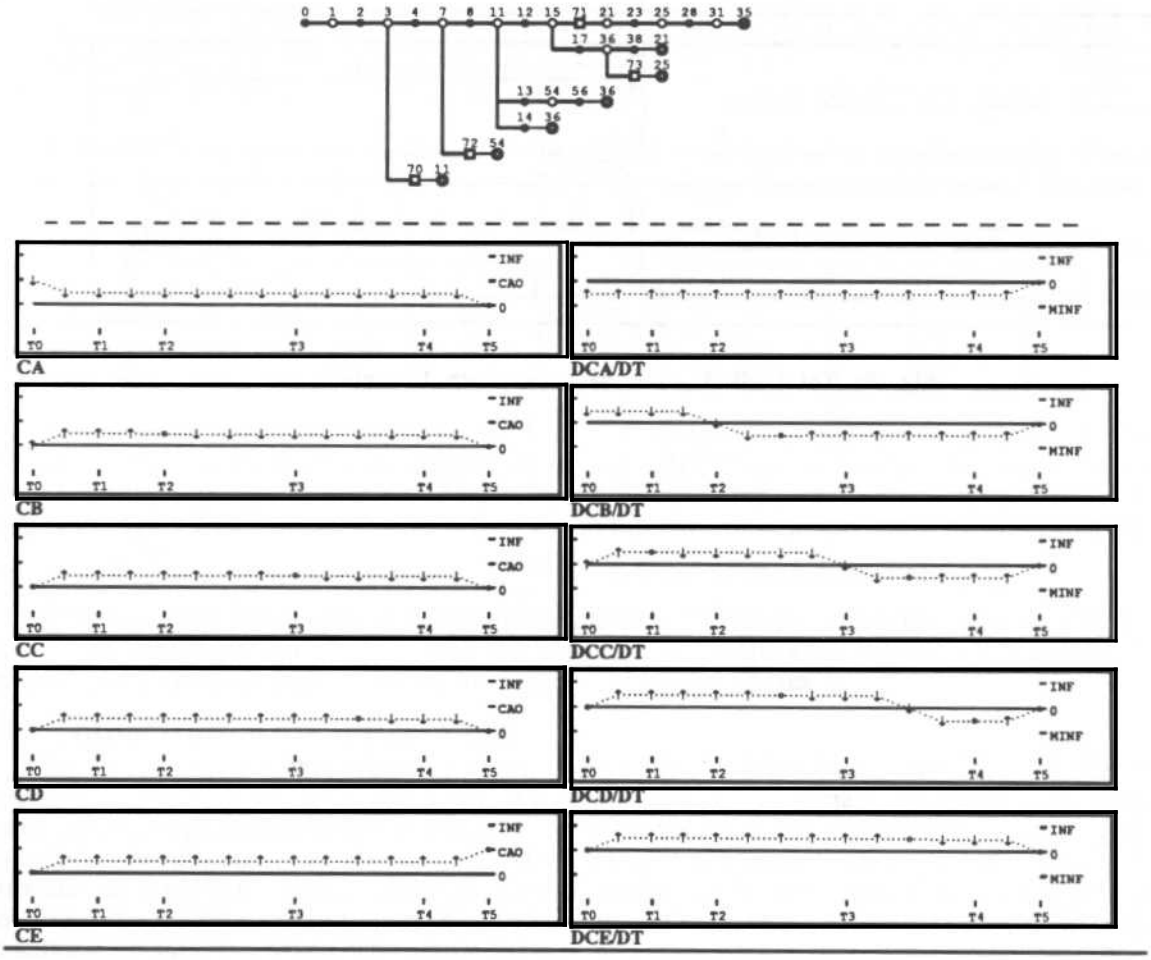

Fig. 5: Partial elimination of occurrence branching

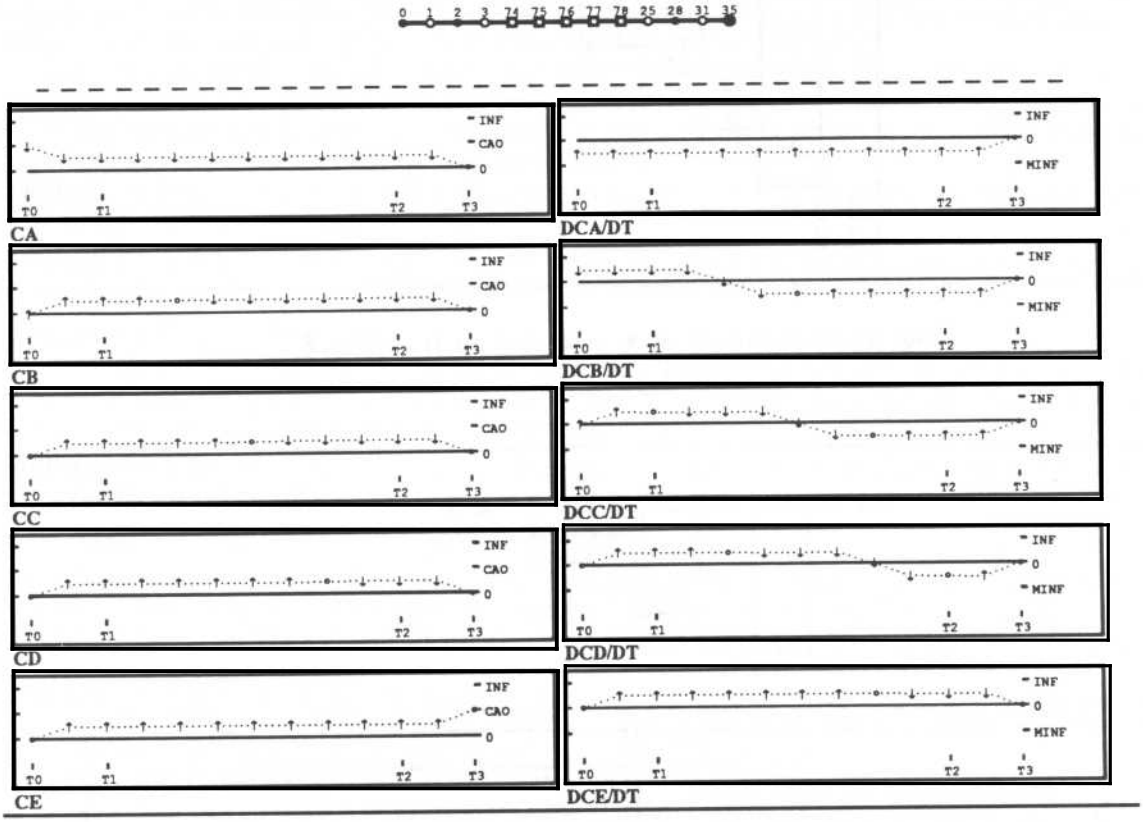

Fig. 6: Total elimination of occurrence branching

occur sometime in the behavior. States built in this aggregation phase are labelled to distinguish them from other standard states. We kept the state-based representation to have a unique representation of behaviors.

Once all the SISO subgraphs are analyzed, the algorithm is applied again on the resulting graph, and so until no more SISO subgraph can be found.

3.3. Example

Figure 4 shows the envisionment graph of a system similar to the one presented in 2.4: an isothermal batch reactor where four reactions in series occur: *A->B->C->D->E*. Dalle Molle [1] made a detailed study of that system. The simulation started from an initial state with species *A* at a concentration C_0, used second- and third-order derivative constraints [14], and was constrained to produce analytic functions only. This is an example where the simple behavioral nature of the system is masked by the occurrence branching phenomenon.

Once enumerated, the graph contains 39 behaviors, and the variables considered individually have the same behavior accross all the system's behaviors: *A*'s concentration decreases towards 0; concentrations of *B*, *C* and *D* reach a peak and then decrease towards zero; *E*'s concentration increases towards C_0. The distinctions occur because certain critical points can be achieved before, at the same time, or after other critical points depending on the value of the system's parameters. Here five pairs of critical points exhibit occurrence branching. Note that while five pairs of variables, each of which can lead to three distinctions, could generate 243 distinct possibilities, only 39 behaviors remain.

Figure 4 shows the envisionment graph as it is produced by QSIM. Arrows are not plotted, but edges are oriented from left to right and top to bottom. Cycles are not explicitly represented, but states with multiple predecessors are duplicated and printed as small circles surrounded by larger circles.

Four SISO subgraphs are detected at this level. Each subgraph contains one class of history-equivalent behaviors, and only one new state per class is built, as shown in the table below:

Subgraph	S_i	S_f	Other nodes	New nodes
G_1	S_3	S_{11}	S_5 S_6 S_{66} S_{67}	S_{70}
G_2	S_7	S_{54}	S_{10} S_9 S_{60} S_{62}	S_{72}
G_3	S_{15}	S_{21}	S_{18} S_{16} S_{19} S_{20}	S_{71}
G_4	S_{36}	S_{25}	S_{42} S_{40} S_{45} S_{48}	S_{73}

The resulting graph is shown in figure 5, as well as one particular behavior. New states are represented as squares. Behaviors are plotted slightly differently: we still have a succession of *P-qmag*s and I-*qmag*s, but *P-qmag*s no longer have a specific time point associated with them when they belong to a special state. For instance, we do not know any longer the temporal ordering of the events $\frac{dC_C}{dt}$ *reaches a critical value* and $\frac{dC_D}{dt}$ *crosses zero* in the behavior of figure 5.

When the algorithm is applied a second time, another SISO subgraph is detected, starting from state S_3 and ending at state S_{25}, and all the behaviors are history-equivalent. Thus the graph is reduced to a chain and five new states are created, as shown in figure 6.

4. Comparison with other work

Chatter is a well-known problem in qualitative simulation. It has been identified by de Kleer & Bobrow [2] and Williams [15]. Basically, if at some time point a variable transitions to a critical point (that is, its derivative becomes zero), and its derivative is only constrained by continuity, then its qualitative value in the next open interval of time is determined by its second derivative. If no information is provided about this second derivative then qualitative simulation branches on each possible future. Multiple occurrence of this phenomenon leads to intractable branching. Two ways of solving the problem have been studied so far:

The first consists of trying to constrain variables whose qdirs are not sufficiently constrained. Variables which are likely to chatter have to be identified before simulation, and expressions for the

second derivatives of these variables must be derived by algebraic manipulations. Such expressions may be hard to compute by hand. An algebraic manipulator can be used to do the algebra, as in QSIM, but designing a manipulator to produce expressions which can be efficiently used in qualitative simulation (that is, expressions which produce qualitatively non-ambiguous results) is not easy. Moreover, in many cases, one has to make additional assumptions about the nature of some constraints (for instance, the *sign-equality* assumption for M^+ and M^- constraints in QSIM) to be able to derive expressions for the sign of second-order derivatives. However, when such assumptions make sense (for instance, for quasi-linear systems) and when algebraic expressions can be obtained, using higher-order derivatives is a very powerful filtering technique. [14] gives a detailed account on how to use higher-order derivatives in qualitative simulation.

The second consists of masking the problem by ignoring the directions of change of chattering variables [13]. Similarly to the preceding method, it is necessary to identify chattering variables before simulation but tedious algebraic manipulations, as well as additional assumptions, are no longer needed. When used in conjunction with higher-order derivatives, it can weaken the power of the latter if the direction of change of a variable is both ignored and used in an expresion for the second-order derivative of some other variable. [1] provides examples of such bad interferences.

In comparison with these methods, the aggregation method does not require any assumption or manipulation before simulation, but is used as a graph simplification tool. Of course it can be used only when envisioning, as opposed to the others which can be used when directly generating behaviors as well as envisioning. It does not interfere with the use of higher-order derivatives. It provides a finer level of description than the ignore-qdirs method, for the direction of change of a variable is only ignored when necessary.

The following table summarizes the differences described above.

	Higher-order derivatives	Ignore qdirs	Aggregation
Identify chattering variables	Yes	Yes	No
Manipulate equations	Yes	No	No
Always applicable	No	Yes	Envisionment only

The problem of occurrence branching was identified by Williams [16]. He did not adopt a state-based representation, but an history-based representation. For Williams, a behavioral description is composed of two parts: the histories of the system's variables and the set of relevant temporal relations between events. Histories and temporal relations are built incrementally, using a contraint propagator called TCP. This is a major difference between his and our approach: Williams argued that eliminating occurrence branching during simulation was necessary to avoid combinatorial explosion. While this argument is theoretically valid, we never simulated a system for which occurrence branching caused a major problem at simulation time. Problems always showed up at interpretation time. For instance, simulating the second exemple took about 80 s, and the aggregation phase about 20 s. The major advantage of TCP is that it maintains dependency traces for causal explanations. On the other hand, feedback systems are not easy to handle in TCP, but are treated in a very natural way by QSIM [5]. The main advantage of our method is that it integrates state-based and history-based representations into a single framework, in which many other qualitative simulation techniques are available.

5. Conclusion and Future Work

We identified two problems, chatter and occurrence branching, which cause the proliferation of very similar behaviors. We presented two algorithms to eliminate these problems and to obtain much more concise envisionment graphs. We see these algorithms as basic tools to incorporate in a general qualitative simulation framework. In this paper we showed that histories and envisionments can be used together. As we explained in the first section of this paper, the concepts of envisionment and dynamic landmark creation are still hard to put together. Though envisioning allows algorithms based on graph theory to be used efficiently, dynamic landmark creation is a powerful tool to study important problems such as stability of controllers, but can be used only

when behaviors are generated. Trying to unify behavior generation / landmark creation and envisionment is currently one of our directions of research.

Acknowledgments

This work benefited from discussions with many members of the Qualitative Reasoning Group at the University of Texas at Austin, and people of the AI team at the University of Compiègne. Special thanks to Pierre Villon for his help in graph theory and Anne Charles for a careful reading of a previous draft of this paper. Pierre Fouché thanks Prof. Jean-Paul Barthès for his constant support in this research.

References

[1] D. Dalle Molle, "Qualitative Simulation of Dynamic Chemical Processes", Ph.D. dissertation, University of Texas at Austin, 1989.

[2] J. De Kleer, and D.J. Bobrow, "Qualitative Reasoning With Higher Order Derivatives ", in *Proceedings of AAAI-84, Austin, TX,* 1984, pp. 86-91.

[3] K.D. Forbus, "Qualitative Process Theory", *Artificial Intelligence*, vol. 24, pp. 85-168, 1984.

[4] K.D. Forbus, "The Qualitative Process Engine", In *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufman Publishers, D.S. Weld, and J. De Kleer, Eds., pp. 220-235, 1990.

[5] P. Fouché, and B.J. Kuipers, "Qualitative Simulation of Feedback Control Systems", in *Proceedings of MIM-S2 90, Bruxelles, Belgium,* September 1990.

[6] P. Fouché, and B.J. Kuipers, "An Assessment of Current Qualitative Simulation Techniques", In *Recent Advances in Qualitative Physics*. MIT Press, P. Struss, and B. Faltings, Eds., 1991.

[7] P. Fouché, and B.J. Kuipers, "Abstracting Irrelevant Distinctions in Qualitative Simulation", Tech. Rept. Forthcoming, The University of Texas at Austin, Department of Computer Science, 1991.

[8] P.J. Hayes, "The Naive Physics Manifesto", In *Expert Systems in the Microelectronic Age*. Edinburgh University Press, D. Michie, Ed., pp. 242-270, 1987.

[9] J.E. Hopcroft, and R.E. Tarjan, "Dividing a Graph into Triconnected Components", *SIAM Journal of Computing*, vol. 2, No 3, September, 1973.

[10] J. de Kleer, and J. Brown, "A Qualitative Physics based on Confluences", *Artificial Intelligence*, vol. 24, pp. 7-83, 1984.

[11] B.J. Kuipers, "The Limits of Qualitative Simulation", in *Proceedings of IJCAI-85, Los Angeles, CA* , 1985, pp. 128-136.

[12] B.J. Kuipers, "Qualitative Simulation", *Artificial Intelligence*, vol. 29, pp. 289-338, 1986.

[13] B.J. Kuipers, and C. Chiu, "Taming Intractible Branching in Qualitative Simulation", in *Proceedings of IJCAI-87, Milan, Italy,* 1987, pp. 1079-1086.

[14] B.J. Kuipers, C. Chiu, D. Dalle Molle, and D. Throop, "Reasoning with Higher-Order Derivatives in Qualitative Simulation", Tech. Rept. AI89-110, The University of Texas at Austin, Department of Computer Science, 1989.

[15] B.C. Williams, "Qualitative Analysis of MOS Circuits", *Artificial Intelligence Journal*, vol. 24, pp. 281-346, 1984.

[16] B.C. Williams, "Doing Time: Putting Qualitative Reasoning on Firmer Ground", in *Proceedings of AAAI-86, Philadelphia, PA,* August 1986, pp. 105-113.