Managing Large Temporal Delays in a Model Based Control System

Fano Ramparany
ITMI
BP 87, 38244 Meylan, France

Abstract:

In this paper we explain how we have integrated the functionalities of a constraint management system and a temporal data base system to enable a model-based control of systems that exhibit large delays between the events characterizing its behaviour.

Examples of problems where our approach is appropriate, include the monitoring and controlling of complex and geographically distributed systems. Such applications require a robust modeling of the behaviour of the systems, in terms of causal relationships among its state variables, and the handling of temporal delays that may span between an event and its causal influences all over the system.

Our approach as been applied to build a KBS for assisting heating central operators to optimize the efficiency and profitability of the heating process.

1 Introduction

The main difficulty in controlling urban heating systems stems from the geographical distribution of the system to be controlled as it introduces large delays between events and their effects. Another source of difficulty is related to the complexity of the heating system, in terms of the number of its components, the behaviour of these components and the interactions among them.

An example of the topology of an urban heating system is displayed in figure 1. End user heating stations are depicted with the symbol ⋈. Such a network involves more than a hundred stations, all interconnected with heating flow pipes.

When an operator located at Bissy station, modifies the temperature of the primary network (heating flow), its effect impacts on the temperature of the secondary network (heated flow), from one to three hours later depending on the speed of the primary flow.

The problem is complicated by the number of parameters which need to be measured, and perhaps

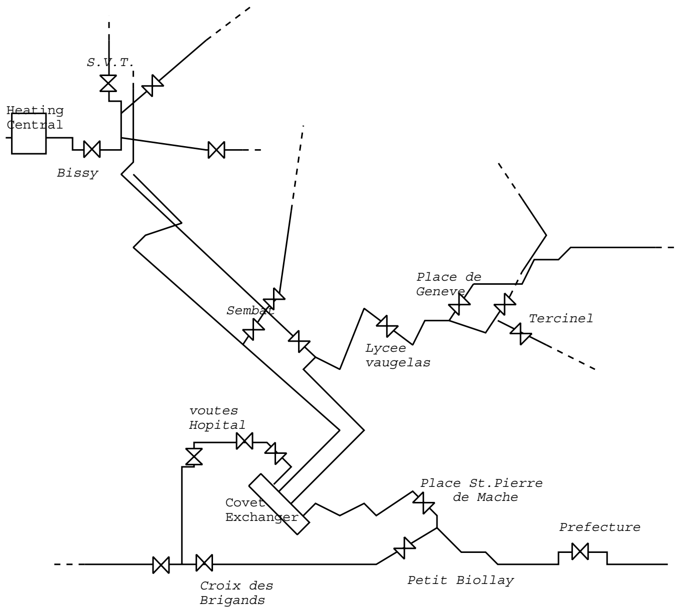

Figure 1: Part of the map of Chambery city's urban heating system

eventually modified to change the behaviour of the system. Furthermore the space of control parameters itself contains redundancy.

For example, fulfilling an increase of the users consumption needs can be realized either by opening the primary network floodgates, or by raising its temperature.

Using a deep model of a system to control and monitor its process has largely been proven a successful approach [3]. Model based control and more generally model based reasoning qualify reasoning processes that draw their inferences upon functional, structural or behavioural models of the system to be controlled.

Several techniques to build and exploit such so called "deep models" have been developed and used so far. Quantitative models are usually defined as a set of numerical or symbolical constraints that relate relevant variables of the system to each other.

When the number of equations is too high, or if some of them are not available, a qualitative approach is often used. This constructs an approximate model of the system, and reasons more abstractly about it to derive consistent states which encompass physical qualitative behaviors.

However, none of current "deep" modeling techniques is able to manage large delays between variables. Instead of extending one of them to fit this specific requirement, we propose a more flexible approach which integrates a constraint management system with a temporal database system. The constraint management system enables the causal modeling of the system. The temporal database system is used to store the temporal histories of some properties of the system that are required when modeling the behaviour of the system.

In this paper, we describe the problems of integrating causal and temporal reasoning. We then introduce the main concepts of our approach for tackling them. We show how we have applied this approach for developing a system assisting urban heating power station operators to tune the control parameters of the system, in order to optimize the efficiency and profitability of the heating process, while satisfying the user consumption requirements. We finally discuss our approach and compare it to related approaches.

2 Problem Statement

When controlling complex systems with large temporal delays, two types of requirements have to be faced. Reasoning about time, and reasoning about system functions and behaviour.

More specifically, the following capabilities are generally required:

Temporal projection: makes it possible to know which properties of the system are known and what those properties are at some future point in time.

Anticipatory action: enables the planing of an action sufficiently in advance, so that it effects will occur at the right time. For example, if we want somebody to receive a parcel within two days, we should better post it now.

Temporal queries: are queries about which property holds within a specific temporal window. For example, a football match organizer might be interested in checking whether it has rained yesterday evening as to how the terrain grass will look like.

Causal modeling: allows describing the set of parameters that are relevant to describe the state of the system, and to explicitly represent the network of causal relationship among these parameters.

Simulation: completes the description of the system state or infers its subsequent states, based on a partial description of the initial state. This kind of inference is mainly supported by a causal model of the system to be controlled.

At the requirements analysis level, we can clearly separate representation and reasoning about time, from representation and reasoning about causal dependencies. Temporal projection, anticipation and queries are typical inferences expected from a temporal reasoning system. Causal modeling and simulation are typical inferences expected from a causal modeling tool.

From a methodological point of view we propose to preserve the separation between temporal and causal reasoning at the design and implementation stage of the system. It has been argued that such methodological option allows a modular design of the target system, thus clarifying its conceptual architecture and easing its development as well as its maintenance, specially when different development teams are to be involved. This decomposition approach to design is also a very pragmatic way to "reuse" existing modules within new applications.

In the following sections we will show that reusing and combining existing and standard techniques for handling temporal reasoning and causal reasoning, can reveal a very powerful and efficient development strategy.

3 Integration framework

We describe here the integration framework that has been adopted to enable temporal and causal reasoning to cooperate. We very briefly summarize the basic notions of temporal databases and constraints satisfaction problems as far as they are relevant for this paper. The reader who is conversant in these areas may skip the respective subsections, or just browse through to pick up the terminology.

3.1 Temporal databases (TDBMS)

As a conventional database system a TDBMS store facts and objects, the specificities of TDBMS is that it provides facilities to describe their temporal properties [10]. For example one can specify the time at which some fact becomes true, and when it becomes false. So that one can query at which time(s) some fact was true in the past, or will be true in the future. One can can even ask about which facts are true within a specific temporal window.

The temporal database that we have used is based on Sergot and Kowalski Event Calculus (EC) framework [6].

The EC is a treatment of time in first-order classical logic, based on the notion of events in order to temporally generate properties that hold for a given time interval. EC combines the expressive power of situation calculus and the computational power of logic programming. The EC axioms allow generation of answers to queries about the holding properties.

Although many extensions to the EC framework are possible, such as the handling of different temporal granularities and continuous change, we have basically restricted its use to the basic framework.

3.2 Constraint management systems (CMS)

The idea of using constraints as a problem solving and programming paradigm is very useful for model based reasoning, such as diagnosis, simulation and control. This is because much of the modelling deals with relationships which are naturally described as constraints. For example, the underlying mathematical models for a range of physical phenomena such as the conservation of heat, fluid flow, the relationship between current and voltage, etc. can be described by constraints.

The constraint management system built here is based on the constraint logic programming language CLP($\mathcal{R}$) [5, 4]. The essential idea of the constraint programming paradigm is that constraints represent relationships between the objects. The underlying constraint solver then guarantees the consistency of the constraints as a whole in the system and also solves for the values whenever possible. When we do not have specific unique values for the variables then "answer constraints" also serve as output to describe the solution space. Thus when posing constraints to the CMS we do not care if there are values for the variables, we know that those values will be computed either now or later implicitly when more information is obtained or when more constraints are added.

Constraint systems are characterised by the kinds of constraints expressible and the computational power of the constraint solver. The underlying CLP($\mathcal{R}$) solves all linear arithmetic constraints directly and completely solves these class of constraints. More complex non-linear constraints are not solved directly and are deferred to be solved by local propagation when they become sufficiently linear.

3.3 Interfacing the TDBMS with the CMS

We will now see how these two paradigms can be associated into a system that can manage a constraint network of temporal properties.

Specifying the interface between the TDBMS and the CMS amounts to defining the *conceptual mapping* between their respective representation (in terms of what a conceptual primitive in one representation corresponds to the conceptual primitives of the second representation), the *functional interface* which define the services that one of the system will request from the other system in runtime, and the *control architecture* which specifies the conditions required for such interaction.

3.4 Conceptual Mapping

CMS events, i.e. variables and their associated values, are simply stored as items in the TDBMS which are dated according to the absolute date of their computation. Not all variables histories are worth storing in the TDBMS, only those that will effectively be used for causal simulation are.

3.5 Functional Interface

The TDBMS offers its full functionalities to the CMS. Thus allowing the CMS to post dated events, to perform temporal queries including retrieving of properties that hold at a given time point, or interval.

Conversely, the CMS may be requested by the TDBMS to post new constraints, or variable instantiations (which are considered by the CMS as a special constraint specification), in which case the CMS will automatically propagate the effects of these new constraints on all the variable of the system to derive an updated consistent state, or detect some constraint violations. This mechanism allows the simulation of effects of future events.

3.6 Control architecture

The control architecture is specified using metarules that identify specific situations that are recognized by one of the reasoning schemes, in which it has to request services from the other reasoning scheme. The description of the service to be requested is also given by the metarule.

An example of such metarule is:

" If the temperature of component A has not exceeded a critical threshold T_c^o during the past 2 hours, then component A exhibits the nominal behaviour, and its output temperature will be twice that of its input temperature."

In this metarule, the situation which is *"component A's temperature is below T_c^o during the last 2 hours"* is recognized by the temporal data base, whereas the action to be taken *"output temperature*

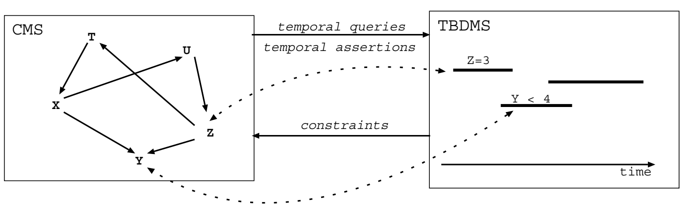

Figure 2: Integration architecture

of component A is twice that of its input value" is interpreted by the causal reasoner.

Metarules can be domain dependent, or task dependent. In the latter case this metarule can be reused in other domain for similar problem.

The figure 2 summarizes the features of the integration framework. In the following section we show how it instantiates in the context of a real application.

4 Application

We have implemented a system for providing operators with recommendations on which control actions to take in order to optimize the heating process of Chambery city.

A urban heating system can roughly be described as interconnected networks of heating components such as heat exchangers and urban heating power stations, and mechanical components such as Pump and Flow pipes.

Heat exchangers allow the heating of a secondary flow, using the heat brought by an incoming primary flow.

Urban heating power stations consume fuel and gas to produce heat which is transferred to a circulating flow.

Heating components are linked together through flow pipes carrying liquid that transmit heat from one component to the others.

Hydraulic pumps maintain a controlled flow of the heating liquid that circulate in the pipes.

The basic association of a central and exchanger is shown in figure 3.

During the analysis phase of the system we have interviewed two types of experts:

System operators who daily monitor the heating power station and tune its control parameter to satisfy the users requirements, and adapt the system to the weather condition fluctuations. Operators possess implicit models of how some part of the heating network behave. For example, they know that increasing the speed of the primary flow pump will increase the temperature of the secondary flow through the heat exchanger.

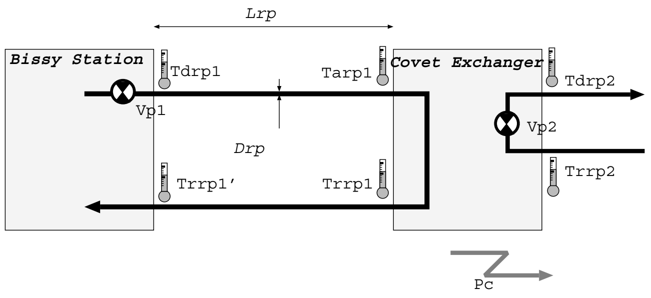

Figure 3: Example of two components of the urban heating system

System designers who know the physical characteristics of the heating system, such as the length of the various networks, the characteristics of the exchangers.

A first analysis with the operators helped identify which parameters are relevant for characterizing the state of the heating system.

This preliminary analysis was completed by interviewing the system designers, to refine the relationships between those parameters and make explicit their underlying theoretical justifications. Several additional parameters were introduced by the system designer that were ignored or unconsciously skipped by the operators. For example, the dependency between the pump speed Vp and the temperature of the secondary flow exiting from the heat exchanger $Tdrp2$ that has been identified by the operators, can be explained with a causal chain comprising four links.

The first one relates Vp to the flow $Drp1$ circulating in the primary flow pipe. The second one relates $Drp1$ to the power transmitted to the exchanger PC. The third one relates PC to the "loss" of temperature between the primary and secondary flow of the exchanger $DTCe$, which is itself obviously related to $Tdrp2$.

Each of these causal link can be formalized as a numerical constraint, that involves the two variable linked and eventually other variables. For example, the first causal link described above represents the following equation:

$Drp1 = K_p * Vp$

where K_1 is a constant characterizing the pump.

This two stages analysis phase has produced a behavioural model of the system. Part of this model is displayed in figure 4, where arrows depict causal links.

Each of the arcs of the network depict a numerical constraint that has been registered in the CMS constraints base.

Prediction of the users power consumption is computed by a specific external module which is not described here. They are stored in the TDBMS along with the corresponding time of the prediction.

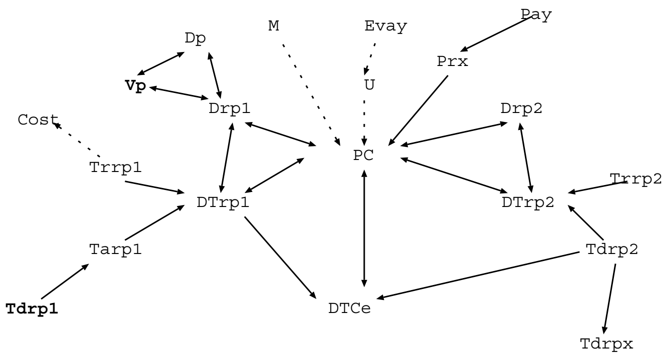

Figure 4: Part of the behavioural model of the heating system

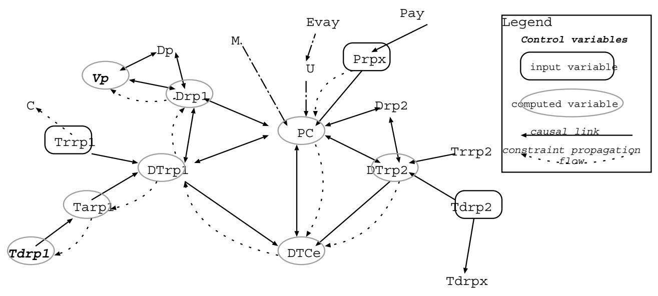

Figure 5: Example of a causal constraint propagation for control recommendation generation

The constraints on the prediction of users future power consumption, are registered by the CMS which also propagates the effects of these constraints. Due to the geographical distribution of the various user stations, the effects of a constraint occur at various instants in time depending on the location of the stations and the speed of the flows that feed them. This temporal information is stored in the TDBMS. The CMS ensures the constraints are applied and kept consistent. As constraints are setup to describe the causal model of the heating system network, the input parameters to the system cause variables which are directly related to be solved, and this effect propagates throughout the causal model which in turn enables solving other variables. The constraint propagation ends when the control parameters (those used by the operators to tune the heating system) are assigned their values. Figure 5 displays a trace of such constraint propagation that instantiates the pump speed Vp and the temperature of primary flow $Tdrp1$ exiting from the "Bissy" principal heating power station. Vp and $Tdrp1$ are the control parameters of the "Bissy" heating power station.

5 Discussion

The problem of handling of large temporal delays for process control has been identified for some time by research in control theory [2]. Specific techniques have been introduced for this matter.

Model based approaches of process control problems, have been introduced during the last decade in order to overcome the shortcomings of classical control theory techniques with respect to the needs for high level explanations and incompleteness in the knowledge of the process to control.

We first looked into qualitative physics and systems like FTQ [1] and QSIM [7], as such models have been used in the application domain of central heating systems [8]. However, FTQ requires variables histories to be continuous, as well as a rather complete knowledge of the system to model in terms of qualitative transfer functions. On the other hand, QSIM allows more approximations to be done while modeling the physical system and its behaviour, but QSIM doesn't provide for managing large delays.

The power of qualitative modeling technique for modeling transient states was not necessary in our case as the time granularity we worked at, was far coarser than the time constant of thermodynamical phenomena. Thus steady state equations that don't involve any temporal derivatives are sufficient in our case.

Due to the limitation of current causal modeling techniques when reasoning on a phenomenon with large temporal delays, we have proposed a solution that loosely integrates the functionalities of a temporal database management system and that of a constraint management system. This approach enabled us to combine existing robust temporal and causal reasoning techniques. For this purpose we have introduced a general framework for integrating heterogeneous representation and reasoning formalisms, that we have applied to the more restricted problem of integrating temporal and causal reasoning.

From a more general perspective, characteristics shared by complex dynamic applications that hamper their deployment include the requirement to handle more than one of the following: evolution of the information with time, incompleteness, unreliability and uncertainty of information which has to be managed. Complex applications also often require a cooperation between specialized knowledge-based facilities for simulation, fault diagnosis, planning/scheduling, emergency decision making, etc...

We believe that such approach of loosely integrating heterogeneous formalisms helps in developing such applications. Along this direction, we are currently engaged in the ongoing Esprit-III project UNITE [9] which aims at tackling these integration issues. More specifically, we are developing appropriate techniques for integrating heterogeneous knowledge models including incompleteness, uncertainty and time-dependency (internal integration) and for the cooperation between knowledge based facilities (external integration). These techniques will be implemented in an integrated support environment composed of a generic platform which supports the development and integration of complex applications. This support environment and the applicability of our approach are being assessed in the framework of two other pilot applications in the domain of space control centers and neonatal intensive care unit in hospitals.

One main lesson that we have learned through the work described in this paper is that the following three aspects should be carefully specified when integrating heterogeneous formalisms.

Conceptual mapping: "what a conceptual primitive in one representation corresponds to in terms of the conceptual primitives of the second representation?"

Functional interface: "which services will one system request from the other system at runtime?"

Control architecture: " what the conditions required for interaction are, and which corresponding actions should then be triggered?"

The second lesson is that such loosely integration approach reveals a efficient development strategy, as it allows existing techniques and tools to be effectively reused.

Acknowledgements

Thanks to Renaud Zigmann, Roland Yap and the anonymous referees for their valuable comments on early drafts of this paper. The research reported here was carried out in the course of a project partially funded by the "Agence pour la Maîtrise de l'Energie". The partners in this project are Cofreth, LIAC-Lyonnaise des Eaux, SCDC and ITMI.

References

[1] P. CALOUD. *Raisonnement Qualitatif.* PhD thesis, INPG, 1989.

[2] S. DADEBO and R. LUUS. Optimal control of time-delay systems by dynamic programming. *Optimal Control Applications & Methods*, 1992.

[3] F. HAYES-ROTH, D.A. WATERMAN, and D.B. LENAT. *Building Expert Systems.* Addison Wesley, 1983.

[4] N.C. HEINTZE, J. JAFFAR, S. MICHAYLOV, P.J. STUCKEY, and R.H.C. YAP. *The CLP($\mathcal{R}$) Programmer's Manual*, 1992.

[5] J. JAFFAR, S. MICHAYLOV, P.J. STUCKEY, and R.H.C. YAP. The CLP($\mathcal{R}$) language and system. *ACM Transactions on Programming Languages and Systems*, 1987.

[6] R. KOWALSKI and M. SERGOT. A logic-based calculus of events. *New Generation Computing*, 4, 1986.

[7] B. KUIPERS. Qualitative simulation. *Artificial Intelligence*, 1986.

[8] F. LACKINGER and W. NEJDL. Diamon: A model-based troubleshooter based on qualitative reasoning. *IEEE Expert*, 1993.

[9] F. Ramparany and al. Knowledge integration and interchange issues in imperfect and time dependent information systems. In *Proceedings of the IJCAI'93 Knowledge Sharing and Information Interchange workshop*, 1993.

[10] S. M. SRIPADA. A logical framework for temporal deductive database. In *Proceedings of the 14ht VLDB Conference*, 1988.