Domain Modeling for Monitoring Systems[1]

Jeff Primus and Boi Faltings

Laboratoire d'Intelligence Artificielle
Swiss Federal Institute of Technology (EPFL)
MA-Ecublens, 1015 Lausanne, Switzerland
FAX: +41-21-693-5225, e-mail: primus@eldi.epfl.ch

January 16, 1992

Abstract

Applying a knowledge-based approach to monitoring and controlling real-time systems is often infeasible because the execution time of backward-chaining queries is unpredictable. In this paper, we show how a decomposition of reasoning into forward and backward inference can bound the execution time of queries. We present a domain modeling technique with algorithms for classifying knowledge into forward and backward categories.

Topics: Reasoning about Physical Systems: Modeling
Knowledge Representation: Temporal Reasoning

Short paper

[1]This work is done in the context of ESPRIT II project #2409 EQUATOR.

1 Introduction

Monitoring and controlling complex real-time systems is a promising application for knowledge-based systems. However, most knowledge-based systems use backward chaining which makes query execution time in principle unpredictable. Worse, just determining if the system already has enough information to decide the truth value of a query can take an indeterminate amount of time. The unpredictability, which is a major obstacle to real-time applications, is in particular due to the unbounded recursions entailed by backward chaining. On the other hand, using a pure forward chaining mechanism would result in large amounts of useless inferences, besides being insufficient for planning and scheduling applications.

As a tutorial example, consider a traffic control system for a major airport (Figure 1).

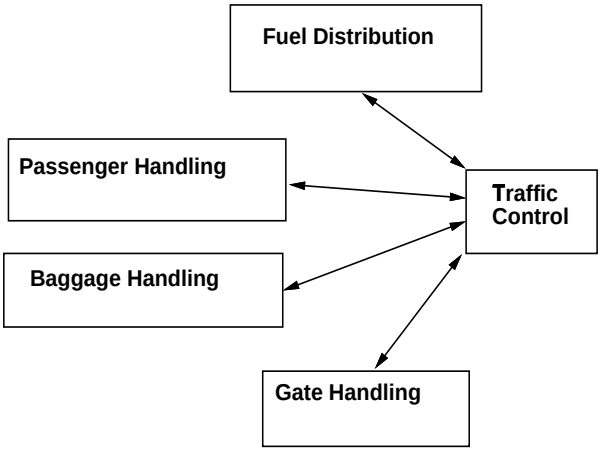

Figure 1: *Interaction between models of different subsystems in an airport.*

When scheduling an aircraft for departure at a particular time, the air traffic control system must verify that it actually will be ready to do so. Any problems with passenger and baggage loading, refueling, gate handling and other services influence decisions in the scheduling system. Checking them using backward chaining can lead to wasting enormous computational ressources, since often lengthy reasoning chains find no conclusion because the required information is not available yet. For example, verifying that baggage will be loaded might involve inconclusive reasoning about possible schedules of baggage loaders which would better be put off until sufficient information about their activity is available.

The key idea of this paper is that the problems can be solved by a proper combination of forward and backward reasoning. We define a *cut set* of predicates with the following characteristics:

- backward chaining will stop when a predicate which is member of the cut set is encountered: if it is not already asserted, the query will fail.
- forward rules are set up so that all predicates which are members of the cut set are inferred as soon as the available information permits it, thus avoiding useless backward reasoning.

When predicates in the cut set are properly chosen, (i) the execution times of queries can always be bounded, and (ii) whenever queries fail this is due to insufficient information about the state of the system.

For example, in the air traffic control system the processes responsible for readying an aircraft would assert the predicates requested by other processes as soon as enough information is available to do so. The ATC module will thus find the response to its query in the database as soon as information is sufficient to evaluate it.

In this paper, we show how domain modeling can automatically determine a suitable cut set and the forward rules necessary for this inference structure. It is based on the General Representation Formalism (GRF) [GRF91], a process-oriented language for modeling time-dependent systems based on Event Calculus [Kowalski86] and developed in the ESPRIT II Project EQUATOR.

2 Cut Sets in Temporal Problems

For the example of air traffic control shown in Figure 1, processes and propositions are shown in more detail in Figure 2. In order to evaluate whether an airplane is ready to take off at time T, the backward chaining mechanism (BCM) will query if the cut set predicates 'Passengers-OK' 'Fuel-OK', 'Baggage-OK' and 'Gate-OK' have already been satisfied at a time earlier than T. Evaluating these predicates can lead to recursive inferences of similar queries: for example, the gate agents could be delayed by a late departure of an earlier flight, thus again leading to a query of the same predicate type. Such recursive chaining means that inference takes an indeterminate amount of time even if it ends up failing due to a lack of information.

A better approach is to use forward chaining to assert those propositions which are curcial for queries as soon as the necessary information arrives. In order to avoid excessive inference of useless propositions, one needs to know which propositions are important to assert. This requires explicit construction of a complete *model* of the system. We call this set of crucial propositions the *cut set*, since the assured forward inference also means that any backward chaining can be cut on encountering a predicate which falls in this set.

Figure 2 shows a model of the airport processes according to the GRF formalism developed in the ESPRIT II project EQUATOR. The GRF is based on a hierarchy of *macro-events*, which correspond to the notion of a *history* in Qualitative Process Theory (QPT) [Forbus84]. Each macro-event includes a list of preconditions and consequences.

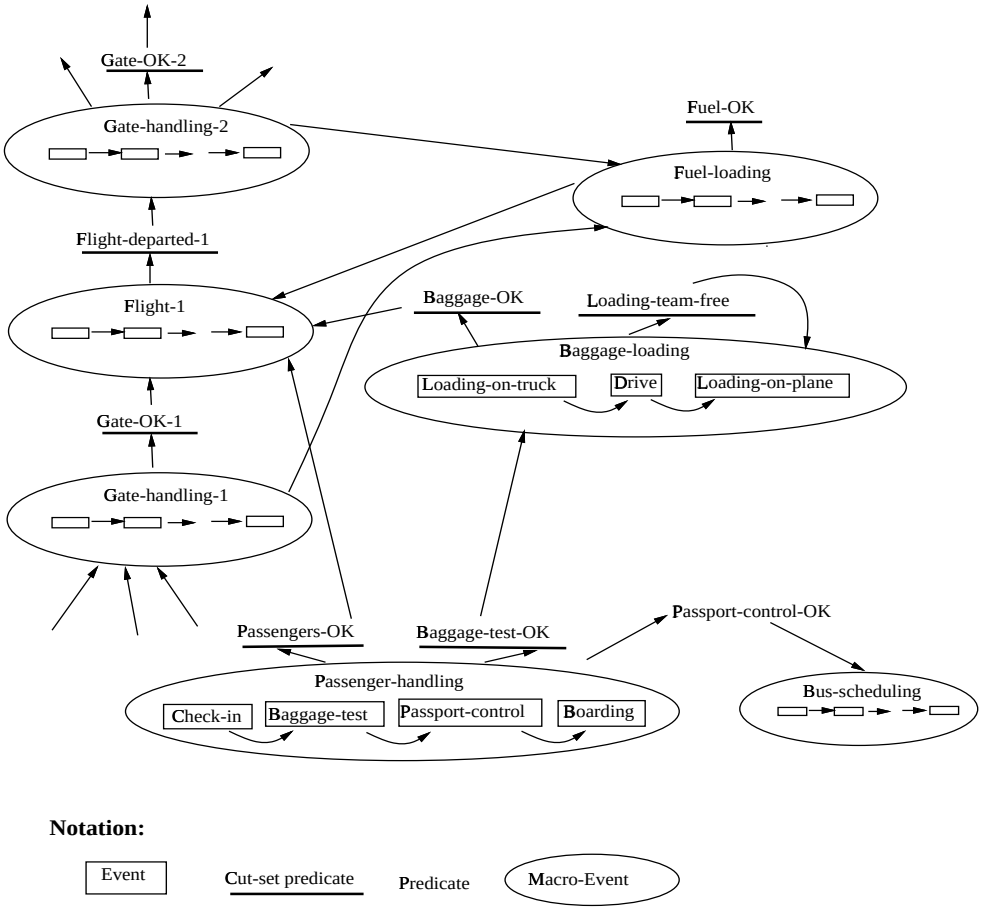

Figure 2: *Example of inferences involving different processes in an airport.*

A macro-event starts when all its preconditions are satisfied. Its consequences become true upon termination of the macro-event. The GRF structure means that the existence of any temporal property can be reduced existence of a macro-event that contains it.

The predicates which form the cut set are defined by considerations of efficiency of backward chaining. In particular, all predicates which could lead to recursive calls of similar predicates should be member of this set. Within the definition of a single macro-event, predicates which must belong to a cut set can be explicitly indicated by the modeller. More complex interactions arise from the combinations of different macro-events. Specifically, cut-set predicates must decouple recursive dependencies of different instances of the same type of macro-event. The underlined predicates in Figure 2 are all potential members of recursive backward chaining and therefore members of the cut set. For example, the predicate $Gate - OK$ is a member of the cut-set since its instance $Gate - OK - 1$ is a precondition for the macro-event $flight - 1$, whose existance could be a subgoal in the

inference of $Gate-OK-2$, thus establishing a recursive inference involving the same predicate types. Other predicates are placed in the cut set because of other circular inference possibilities or to complete the forward inference procedure. On the other hand, many other propositions have only local effects, an example of this is $Passport-control-OK$ as shown in the figure.

3 An Algorithm for Constructing Cut Sets

Dependencies between macro-events occur through preconditions and consequences. The interactions between different *types* of macro-events can be represented as a graph, as shown in Figure 3 for the earlier example of Figure 2.

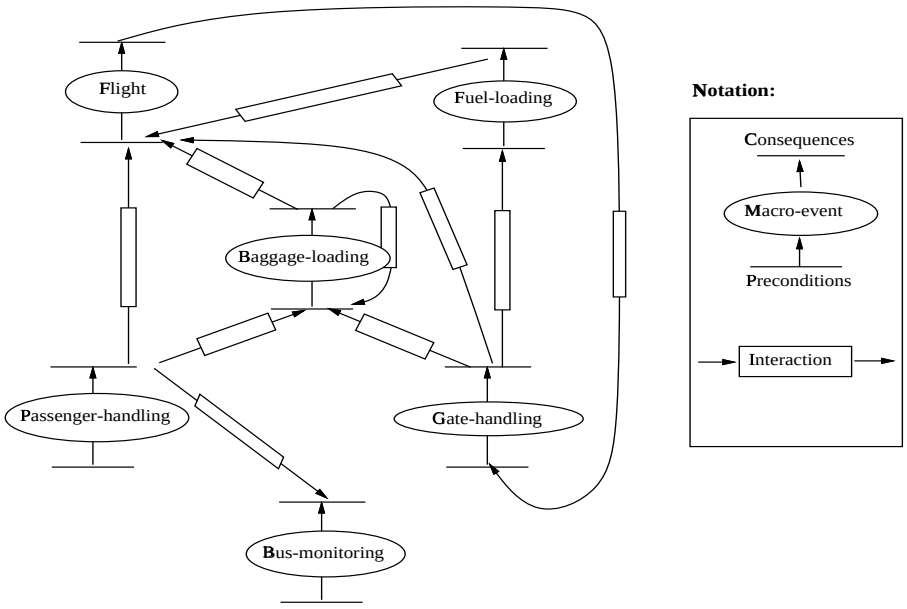

Figure 3: Graph showing the interactions between macro-events.

Nodes in the graph represent macro-event *types*, and a directed arc from x to y represents a predicate which occurs both as a consequence of some instance of macro-event x and as a precondition of some instance of macro-event y. Recursive inference of different instances of the same type of macro-event can occur whenever it is part of a *loop* in this graph. All predicates which represent links in such a loop are placed in the cut set[2]. When backward chaining cannot proceed beyond a predicate in the cut set, its depth is bounded as it can never encounter the same type of macro-event twice in a recursion.

[2]The cut set could be optimized by only including one link of each loop, but we have not considered this question yet.

The other task of the modeling procedure must ensure that forward rules are generated to assert predicates in the cut set as soon as the available information permits it. For each predicate in the cut set, we find the complete tree of potential inference links which connects it to measured predicates or other members of the cut set. By generating forward rules for each of the links, the predicate is guaranteed to be asserted as soon as the available information permits it. Conversely, whenever a query does not find that the predicate is already in the database, it is certain that information is not yet sufficient to deduce it.

The following algorithm constructs cut set and corresponding forward rules:

1. For each macro-event M figuring in a loop DO
   (a) Add to the cut set the preconditions p_i in $p_1, p_2, ...$ of M such that p_i is not already a member of the cut set
   (b) Generate a forward rule for M:
   IF p_1 AND p_2 ... THEN happens(M)
   (c) For each condition p_i which is not an external input or a member of the cut set DO
      i. Find all antecedent macro-event M_A having p_i as a consequence
      ii. Generate forward rules to assert the required consequences of M_A:
      IF M_A TERMINATED THEN p_i (where p_i is a precondition for M)
      iii. IF M_A is in a loop Recurse on step (1), using M_A for M
      ELSE Recurse on step (1.a), using M_A for M

Using this algorithm on the structure of Figure 3, we obtain the cut set predicates as indicated in Figure 2, and corresponding forward rules for asserting them.

4 Conclusions and Future Work

We have shown how the use of explicit domain modeling can solve problems of inference control which often prohibit the application of knowledge-based systems in real-time environments. Our work is related to the observation made by Manny Rayner [Rayner91] that modeling can avoid the need for non-monotonic reasoning.

We are currently working on a complete implementation of these techniques and their integration into the GRF formalism. Future work will involve optimization of the cut set construction, taking into account different forms of event interactions, and integrating the work within the context of a more complete modeling environment such as described in [FF91].

References

[Forbus84] **Kenneth D. Forbus** "Qualitative Process Theory", Artificial Intelligence 24, 1984

[FF91] **Brian Falkenhainer and Kenneth D. Forbus** "Compositional modeling: finding the right model for the job", *Artificial Intelligence* **51**, 1991

[GRF91] **The EQUATOR GRF team**, "ESPRIT II project #2409 EQUATOR The General Representation Formalism", D123-1 ,July 5 1991

[Kowalski86] **R.A. Kowalski and M.J. Sergot** "A Logic-Based Calculus of Events ", New Generation Computing, vol 4 , 1986

[Rayner91] **Manny Rayner** "On the applicability of nonmonotonic logic to formal reasoning in continous time", Artificial Intelligence, vol 49, May 1991